Research Article

Evaluating Adversarial Robustness of Knowledge-Level RAG For Vulnerability Detection With Open-Weight Llms

Roshani Vasant Borgaonkar[1] & Ajay Sirsat[2]

[1,2] St. John College of Engineering and Management, Mumbai University (Autonomous)

Corresponding Author: Roshani Vasant Borgaonkar Email: 125roshani1007@sjcem.edu.in

Abstract

Knowledge-level Retrieval-Augmented Generation (RAG) has recently been shown to substantially improve LLM-based vulnerability detection by grounding predictions in distilled, historical vulnerability and fix knowledge, a framework introduced as Vul-RAG, and follow-up work has confirmed that these gains reproduce on open-weight, rather than only proprietary, LLMs. Independently, work on the robustness of retrieval-augmented language models (RALMs) has shown that imperfect retrieval, whether unanswerable, adversarial, or conflicting retrieved documents, severely degrades RALM performance in open-domain question answering. These two lines of work have not previously been brought together, since no existing study measures how a knowledge-level vulnerability RAG pipeline, built and evaluated on open-weight LLMs, behaves when its retrieved vulnerability-knowledge items are themselves adversarial, contradictory, or irrelevant. This paper unifies the two topics into a single study, formalizes the resulting research gap, and reports a pilot run of a Colab-reproducible implementation on the real public Vul-RAG Linux-kernel CVE benchmark released by Du et al. Using Qwen2.5-7B-Instruct as both the knowledge-distillation and detection model, the pilot finds that the knowledge-level RAG detector's accuracy on held-out CVEs is dominated by a severe bias toward predicting code as not vulnerable, with zero percent recall on the vulnerable class under clean retrieval against 93.3 percent recall on the patched class. This bias produces a floor effect in which aggregate accuracy barely moves under GenADV-style adversarial injection, under conflicting-knowledge injection, or under degraded retrieval; the Matthews correlation coefficient for the vulnerable class is in fact negative under clean, adversarial, and conflicting retrieval, indicating worse-than-chance detection rather than mere insensitivity to the attacks. The paper argues that this null robustness signal is itself an informative finding, since at pilot scale aggregate accuracy alone cannot distinguish a robust detector from a degenerate one, and future evaluation of this threat model must report class-conditional recall and confusion-matrix-derived metrics rather than accuracy alone. A second open-weight model, Llama-3.2-3B-Instruct, could not be evaluated in this run because of a Hugging Face gated-repository authentication failure, which is reported here as an execution limitation rather than as a result.

Keywords: adversarial robustness, knowledge-level retrieval-augmented generation, open-weight large language models, vulnerability detection.

1 Introduction

Large language models have shown strong code-comprehension abilities [1, 2, 3], but prompt-only vulnerability detection remains unreliable [4, 5], particularly at distinguishing vulnerable code from textually similar, already-patched code [5, 6]. Vul-RAG [7] addressed this limitation by introducing knowledge-level RAG [8, 9]. Rather than retrieving raw code snippets, the framework distills abstracted, multi-dimensional vulnerability knowledge, consisting of a semantic cause and a fixing solution, from historical CVEs, retrieves the knowledge items most relevant to a query function, and reasons over them iteratively. A subsequent reproducibility study extended this evaluation to open-weight LLMs, since the original results relied heavily on proprietary models such as GPT-4, and the practical, cost-sensitive deployment of such a system depends on whether the reported gains transfer to open-weight models that developers can self-host [10].

In a separate line of work, Park and Lee studied retrieval-augmented language models for open-domain question answering and showed that such models are inherently fragile to imperfect retrieval [11, 12, 13]. When the retrieved document set is unanswerable, adversarially crafted, or internally conflicting, retrieval-augmented language models frequently hallucinate rather than recognizing the problem [14]. Their study introduces a generative adversarial-document attack, referred to as GenADV, and a Robustness under Additional Document metric, referred to as RAD, to quantify this fragility.

These two literatures share a common object of study, namely retrieval-augmented generation built on an open-weight LLM, yet they have been examined in disjoint application settings, source-code vulnerability detection on one hand and open-domain question answering on the other, and under disjoint threat models, clean retrieval on one hand and adversarial retrieval on the other. Security-relevant RAG systems such as knowledge-level vulnerability detectors are an especially consequential setting in which this gap remains open, because an attacker able to influence a shared or crowd-sourced vulnerability knowledge base, for instance through a poisoned CVE write-up or a manipulated fix rationale in a public advisory feed [15, 16, 17], could

potentially suppress true-positive detections without ever touching the target code itself. This paper combines the two topics into one unified study of adversarially robust, open-weight knowledge-level RAG for vulnerability detection, implements the resulting pipeline end to end, and reports results from a real pilot run.

The contributions of this paper are as follows. First, it unifies knowledge-level vulnerability RAG as introduced by Vul-RAG [7] with the imperfect-retrieval robustness framework for retrieval-augmented language models introduced by Park and Lee [11], producing a single threat-aware evaluation pipeline. Second, it states explicitly the research gap that sits at the intersection of the two source papers, developed fully in Section 3. Third, it presents a Colab-reproducible implementation, described in Section 5, that was run on the real public Vul-RAG Linux-kernel CVE benchmark and that covers knowledge-base construction, retrieval, iterative knowledge-augmented detection, three adversarial-retrieval attack scenarios, and the associated robustness metrics. Fourth, it reports real experimental results in Section 6 from Qwen2.5-7B-Instruct on fifteen LLM-distilled knowledge items and thirty held-out evaluation queries, together with an honest account of a second-model run that failed for an infrastructure reason, namely a gated Hugging Face repository, and a discussion of what the observed floor effect implies for how this threat model should be evaluated at larger scale.

2 Related Work

2.1 Knowledge-Level RAG for Vulnerability Detection

Learning-based vulnerability detection has progressed from token- and slice-based neural models such as VulDeePecker [18], through graph-based models such as Devign [19] and ReVeal [6], to pre-trained transformer models such as CodeBERT [20] and its fine-tuned line-level variant LineVul [21], with dataflow-inspired models such as DeepDFA offering further efficiency gains [22]. Large vulnerability datasets including Big-Vul [23], CVEfixes [24], and DiverseVul [25] support this line of work, although studies of realistic evaluation settings report that performance drops sharply once duplicated and mislabeled data are controlled for [5, 6]. Evaluations of general-purpose and code-specialized LLMs on this task similarly find that prompt-only detection remains unreliable [4, 5], and a literature review of LLM-based vulnerability detection and repair documents this broader trend [26]. Related LLM-based security work includes studies of insecure code generation [27] and the combination of LLMs with program analysis [28].

Vul-RAG constructs an offline knowledge base by using an LLM to distill multi-dimensional vulnerability knowledge, consisting of an abstracted cause and a fixing solution, from historical CVE vulnerability and patch pairs [7]. At inference time the framework retrieves the knowledge items whose cause most closely matches the functional semantics of a query function and reasons over them iteratively. For each retrieved item it asks whether the code matches the cause, and if so, whether the corresponding fix is already applied, terminating with a vulnerable verdict on the first unmatched fix. On the PairVul benchmark, Vul-RAG reported a 12.96 percent accuracy improvement and a 110 percent pairwise-accuracy improvement over representative baselines, including LLMAO [29], LineVul [21], DeepDFA [22], and Cppcheck, and a follow-up ACM Transactions on Software Engineering and Methodology version reports accuracy gains of 16 to 24 percent [7]. A reproducibility and replicability study subsequently re-ran the Vul-RAG pipeline while substituting open-weight LLMs for the original proprietary backbone [10], which directly motivates this paper's framing of bridging the open-weight LLM gap: production and research deployments increasingly require self-hostable models for reasons of cost, data governance, and reproducibility, so it matters whether knowledge-level RAG's benefits persist when the reasoning model itself is open-weight and thus generally weaker at instruction-following than frontier proprietary models.

2.2 Adversarial Robustness of Retrieval-Augmented LLMs

Retrieval-augmented generation combines a parametric language model with a non-parametric retrieval component [8, 30, 31, 32], has been extended to code tasks [33, 34, 35], and is surveyed in [9]. Several studies show that such systems are sensitive to the quality of the retrieved context: language models can be distracted by irrelevant context [12, 13], the composition of the retrieved set, including the placement of relevant passages and the presence of unrelated ones, influences accuracy [36], and models differ in how they resolve conflicts between parametric and retrieved knowledge [37]. Benchmarks that probe noise robustness and counterfactual robustness of retrieval-augmented generation reach similar conclusions [38], and adaptive methods such as Self-RAG attempt to mitigate the problem [39]. A parallel security literature shows that retrieval corpora can be deliberately poisoned [15, 16, 17], that LLM-integrated applications are vulnerable to indirect prompt injection [40, 41], and, from earlier NLP and code-model research, that adversarial inputs and poisoned training data can mislead neural models [42, 43, 44, 45, 46].

Park and Lee systematically study the robustness of retrieval-augmented language models to imperfect retrieval in open-domain question answering, identifying three problematic scenarios: unanswerable document sets, in which no retrieved document actually answers the query; adversarial document sets, in which a crafted document is designed to mislead the model; and conflicting document sets, in which retrieved documents disagree with each other [11]. They introduce GenADV, a generative-model-based adversarial attack that synthesizes a plausible but misleading document, and RAD, a metric for the resulting performance drop. They find that retrieval-augmented language models frequently fail to recognize unanswerable or contradictory contexts and hallucinate instead, and that adversarial documents cause the largest degradation, with the worst case occurring when adversarial and unanswerable conditions co-occur. Related security-RAG work in adjacent domains includes SecureRAG-RTL for hardware vulnerability detection [47] and IntelGuard for malicious-package detection [48], both of which explore knowledge-level RAG but, like Vul-RAG, evaluate only under clean retrieval.

2.3 Gap at the Intersection

No existing study evaluates a knowledge-level vulnerability-detection RAG pipeline under the GenADV or RAD-style imperfect-retrieval threat model, and no existing study of retrieval-augmented language model robustness targets the source-code vulnerability-detection setting or an open-weight-LLM knowledge-level reasoning loop of the kind used by Vul-RAG. Section 3 formalizes this gap.

3 Problem Formulation and Research Gap

Let K denote a vulnerability knowledge base of cause and fix items distilled from historical CVEs, let R denote a retriever that returns the top-k items of K most relevant to a query code snippet c, and let L denote an open-weight instruction-tuned LLM that iteratively reasons over the retrieved items to produce a verdict v ∈ VULNERABLE, NOT_VULNERABLE. Vul-RAG and its reproducibility study evaluate the accuracy of L over R(c, K) under the assumption that K and R behave as intended, that is, under a clean-retrieval assumption [7, 10]. The literature on retrieval-augmented language model robustness shows that this assumption is unsafe in general, since if an adversary can influence K, for instance through a poisoned advisory or a manipulated fix rationale, or can degrade R, for instance through retrieval noise or embedding-similarity collisions with off-topic items, a model's verdict can be flipped without altering c itself [11].

The research gap addressed by this paper is precisely the unevaluated intersection of these two settings. No prior work has measured how adversarially crafted or internally conflicting vulnerability-knowledge items, injected into R(c, K), affect the accuracy and pairwise accuracy of a Vul-RAG-style detector. No prior work has performed such a measurement across multiple open-weight LLM families, which are known to vary in instruction-following robustness relative to the proprietary models used in the original robustness study by Park and Lee. No prior work has adapted the RAD metric and an attack-success-rate metric to the code-vulnerability setting, in which the notion of ground truth is a binary security verdict rather than a question-answering answer string. Finally, no prior work has connected this threat model to a concrete deployment risk, namely knowledge-base poisoning of shared vulnerability-advisory data sources that feed a RAG-based detector. Closing this gap, and reporting what actually happens when the resulting pipeline is run end to end, is the contribution of the remainder of this paper.

4 Proposed Framework: Adversarially-Aware Knowledge-Level RAG

4.1 Knowledge Base Construction

Following Vul-RAG [7], each knowledge item is a tuple consisting of a CWE category, a cause, a fix, a vulnerable code sample, and a patched code sample. Each item is distilled by prompting an LLM with a held-out CVE's vulnerable code, patched code, CVE description, and diff, and asking the model to abstract a general cause and a general fixing strategy from that material.

4.2 Retrieval

Each knowledge item's CWE category and cause text is embedded with a sentence-embedding model, namely all-MiniLM-L6-v2 [49]. A query code snippet is embedded in the same way, and the top-k knowledge items by cosine similarity are retrieved, mirroring Vul-RAG's functional-semantics retrieval step [7].

4.3 Iterative Knowledge-Augmented Detection

For each retrieved item, the LLM is asked whether the query code matches the item's cause, and if it does, whether the code already applies the item's fix. The first cause match that lacks an applied fix yields a VULNERABLE verdict, and if the retrieved set is exhausted without such a match, the verdict is NOT_VULNERABLE. A prompt-only baseline that performs no retrieval is run for comparison.

4.4 Threat Model: Imperfect-Retrieval Attack Scenarios

Adapting the taxonomy of Park and Lee [11] to this setting, three attack scenarios are evaluated alongside the clean-retrieval baseline, each constructed from whatever the retriever actually surfaces as the top item for a given query. This constraint reflects a realistic setting, since evaluation queries are held-out CVEs whose own knowledge item is never present in the knowledge base. In the adversarial scenario, modeled on the GenADV attack, the LLM itself is prompted to rewrite the top-retrieved item's cause and fix into a plausible-sounding but incorrect pair that argues the vulnerable pattern is safe, and this rewritten item is spliced in alongside the genuinely retrieved items. In the conflicting scenario, a second, contradictory knowledge item is derived from the top-retrieved item, asserting that it is in fact a benign false positive, and this item is added alongside the genuinely retrieved items. In the degraded-retrieval scenario, modeled on the unanswerable condition studied by Park and Lee, the genuine top-k items are dropped entirely and replaced with k knowledge items drawn from elsewhere in the knowledge base.

4.5 Robustness Metrics

Four metrics are used to evaluate the pipeline. Accuracy is the fraction of individually correct verdicts. Pairwise accuracy is the fraction of vulnerable and patched code pairs that are classified correctly on both sides, following Vul-RAG's emphasis on distinguishing near-identical vulnerable and patched code [7]. The RAD-style robustness score is defined as clean accuracy minus scenario accuracy for each attack scenario, adapted from the RAD metric of Park and Lee [11]. Attack success rate is the fraction of queries that are correctly classified under clean retrieval but that flip to incorrect classification under a given attack scenario.

5 Experimental Setup

The full pipeline was implemented as a self-contained Google Colab notebook and run on the real, public benchmark released by the Vul-RAG authors, hosted as KnowledgeRAG4LLMVulD on GitHub [7, 23, 24], which contains 4,667 real vulnerable and patched C function pairs distilled from 2,174 Linux-kernel CVEs across the ten most common CWE categories [7, 50, 51]. Model weights were loaded through the Hugging Face Transformers library [52] in four-bit precision using bitsandbytes with the NF4 quantization scheme [53, 54]. The pilot configuration sampled five CWE categories, namely CWE-416, CWE-476, CWE-362, CWE-20, and CWE-787, with three CVEs per category used for knowledge-base construction, yielding fifteen knowledge items in total, and three disjoint, held-out CVEs per category used for evaluation, yielding thirty test queries consisting of fifteen vulnerable and fifteen patched functions. Qwen2.5-7B-Instruct [55] was used as both the knowledge-distillation model and the detection model, and was evaluated under all five conditions, namely baseline, clean, adversarial, conflicting, and degraded retrieval, with no errored queries recorded across all one hundred and fifty rows. Llama-3.2-3B-Instruct [56, 57] was configured as a second model, but its weight download failed with a 401 GatedRepoError from Hugging Face, since the model requires license acceptance and an authenticated token. No Llama results were therefore produced in this run, and this outcome is reported here as an execution limitation rather than omitted silently.

6 Results

Table I reports the real, measured results for Qwen2.5-7B-Instruct across all five conditions on the thirty-query held-out evaluation set.

TABLE I. Qwen2.5-7B-Instruct on the real Vul-RAG Linux-kernel benchmark, using five CWE categories, fifteen knowledge-base items, and thirty held-out queries. The RAD score and attack success rate (ASR) are computed relative to the clean scenario.

Scenario	Accuracy	Pairwise Acc.	RAD score	ASR
baseline (no RAG)	0.500	0.000	n/a	n/a
clean	0.467	0.000	n/a	n/a
adversarial	0.467	0.000	0.000	0.000
conflicting	0.467	0.000	0.000	0.000
degraded retrieval	0.500	0.000	−0.033	0.000

Table II decomposes accuracy by true label, a decomposition that is essential to interpreting Table I correctly.

TABLE II. Class-conditional recall by scenario, for Qwen2.5-7B-Instruct, with Wilson 95% confidence intervals (n=15 per class per scenario). The wide intervals reflect the small pilot sample size and are discussed in Section 6.

Scenario	Recall (VULN)	95% CI (VULN)	Recall (NOT_VULN)	95% CI (NOT_VULN)
baseline	0.067	[0.012, 0.298]	0.933	[0.702, 0.988]
clean	0.000	[0.000, 0.204]	0.933	[0.702, 0.988]
adversarial	0.000	[0.000, 0.204]	0.933	[0.702, 0.988]
conflicting	0.000	[0.000, 0.204]	0.933	[0.702, 0.988]
degraded retrieval	0.000	[0.000, 0.204]	1.000	[0.796, 1.000]

Because sample size and predicted-class counts are known exactly (fifteen vulnerable and fifteen patched queries per scenario), the underlying confusion matrix and several secondary classification metrics for the vulnerable class can be reconstructed exactly from Tables I and II rather than re-estimated; these are reported in Tables III and IV to make the floor effect visually and quantitatively unambiguous, as suggested by a reviewer of an earlier draft of this paper.

TABLE III. Confusion matrix by scenario, for Qwen2.5-7B-Instruct (VULNERABLE is the positive class; n=15 vulnerable, n=15 patched, per scenario).

Scenario	TP	FN	TN	FP
baseline	1	14	14	1
clean	0	15	14	1
adversarial	0	15	14	1
conflicting	0	15	14	1
degraded retrieval	0	15	15	0

TABLE IV. Precision, recall, F1, and Matthews correlation coefficient (MCC) for the vulnerable class, by scenario, for Qwen2.5-7B-Instruct. Values with a zero denominator (degraded retrieval makes zero positive predictions) are reported as 0.000 by the standard scikit-learn convention [58].

Scenario	Precision	Recall	F1	MCC
baseline	0.500	0.067	0.118	0.000
clean	0.000	0.000	0.000	−0.186
adversarial	0.000	0.000	0.000	−0.186
conflicting	0.000	0.000	0.000	−0.186
degraded retrieval	0.000	0.000	0.000	0.000

Table IV sharpens the picture Table I already suggests. MCC [59, 60], which unlike accuracy is not inflated by class imbalance, is negative for clean, adversarial, and conflicting retrieval (−0.186), indicating that the detector performs worse than chance on the vulnerable class under those three conditions, not merely poorly. MCC returns to 0.000 under degraded retrieval and at baseline only because a zero row or column in the confusion matrix makes the coefficient degenerate, not because detection improved. The 95% confidence intervals in Table II, computed with the Wilson score method [61, 62], are wide (of order ±0.15 to ±0.20) precisely because n=15 per class is small, and they visibly overlap across scenarios; this is the paper's own evidence, stated plainly, that the pilot is underpowered to make a fine-grained robustness ranking among the three attack scenarios, even though the qualitative floor-effect finding itself is clear and robust to this uncertainty.

Two observations follow directly from the raw predictions, all one hundred and fifty of which parsed successfully with no failures. First, the detector almost never predicts VULNERABLE. Across the one hundred and twenty RAG-scenario queries, that is, four scenarios of thirty queries each, only three total VULNERABLE predictions were made, and only one of those had a non-null matched knowledge item per scenario, meaning that in each of the clean, adversarial, and conflicting scenarios, exactly one of the fifteen vulnerable queries triggered the iterative cause-then-fix match that Vul-RAG's algorithm requires for a VULNERABLE verdict, while degraded retrieval triggered zero such matches. Second, because clean-retrieval recall on the vulnerable class is already zero percent, none of the three attack scenarios can drive it any lower. Mechanically, the RAD score and attack success rate collapse to approximately zero not because the detector is robust, but because there is no remaining vulnerable-class signal for an attack to destroy. The one directional shift that did occur, in the degraded-retrieval scenario, where accuracy rose from 46.7 percent to 50.0 percent, producing a RAD score of −0.033, reflects the NOT_VULNERABLE recall rising from 93.3 percent to 100 percent when the retrieved

knowledge items are entirely off topic. With no relevant knowledge at all, the detector defaults even more strongly to NOT_VULNERABLE, which happens to be correct more often given the class balance of the evaluation set, rather than because the detector reasoned more robustly.

7 Discussion

The pilot run produced a floor effect rather than the graded robustness degradation that the motivating hypotheses were designed to detect, namely that clean-retrieval accuracy should exceed the prompt-only baseline in the manner reported by Vul-RAG, that harm should increase in the order adversarial, conflicting, degraded, and that robustness should vary by model family. At fifteen knowledge-base items and thirty evaluation queries, the binary cause-check prompt, which asks whether the code under review matches a given vulnerability cause and requires a yes or no answer, appears heavily biased toward a negative answer for Qwen2.5-7B-Instruct in this configuration, so the detector rarely enters the fix-check branch that could produce a VULNERABLE verdict at all. This finding is consistent with, and arguably a sharper version of, the finding in the Vul-RAG reproducibility study that open-weight models can differ substantially from proprietary ones in how reliably they follow the iterative knowledge-augmented reasoning protocol [10].

The practical implication of this result is methodological before it is substantive. At this scale, accuracy and RAD score alone cannot distinguish a robust knowledge-level RAG detector from a degenerate detector that always predicts the code to be safe. Any future run of this pipeline, including a rerun with Llama-3.2-3B-Instruct once the gating issue is resolved, and any run conducted at full benchmark scale, should report class-conditional recall as shown in Table II alongside aggregate accuracy, and should treat a near-zero RAD score as inconclusive rather than as evidence of robustness whenever clean-retrieval recall on the vulnerable class is itself near zero.

This finding also suggests two concrete next steps rather than defenses in the strict sense. The first is to diagnose why the cause-check prompt under-triggers for this particular model and knowledge-base combination, since possible causes include the knowledge items' causes being too abstracted or vague relative to the concrete kernel code under review, or the yes or no single-token framing itself biasing the model toward whatever majority class it favored during instruction tuning. The second is to recognize that only once clean-retrieval recall is nontrivial does it become meaningful to interpret RAD score and attack success rate as evidence about adversarial robustness specifically, as opposed to evidence about detector calibration.

Once clean-retrieval recall has been confirmed to be nontrivial, two further changes to the experimental design would materially strengthen the robustness claim beyond what this pilot can support. The first is scale: a follow-up run should cover ten to twenty CWE categories with twenty to fifty CVEs each for both knowledge-base construction and evaluation, rather than five categories and fifteen items, and should evaluate three to five open-weight model families side by side, for example Qwen2.5-7B, Mistral-7B, Llama-3.2-3B, and a comparable Gemma model [63], so that model-dependence of robustness (untested here because only one model completed a run) can actually be measured. The second is the attacker-detector relationship: in this pilot the adversarial knowledge item is generated by the same model family being evaluated, a white-box-adjacent assumption acknowledged in Section 5. A stronger design would deliberately separate attacker and detector, for instance generating the adversarial item with Qwen2.5 or Mistral-7B while the detector under test is Llama-3.2, and then reversing the pairing, to measure whether adversarial knowledge items transfer across model families or are only effective against the model that generated them.

8 Limitations

Several limitations qualify these results. Llama-3.2-3B-Instruct could not be evaluated in this run because of a gated Hugging Face repository that returned a 401 GatedRepoError, so all reported numbers are for Qwen2.5-7B-Instruct only, and the hypothesis that robustness varies by model family remains untested here. The pilot scale of fifteen knowledge items and thirty queries is sufficient to validate the pipeline end to end, but it is too small, and evidently too imbalanced in the predictions it triggered, to support a statistically meaningful robustness comparison, and the floor effect documented in Section 6 is itself a symptom of this limitation. The near-zero RAD scores are consistent with either a genuinely robust detector or a detector that is simply insensitive to its knowledge base in this configuration, and while Table II shows the data to be much more consistent with the latter explanation, disambiguating the two possibilities definitively requires a configuration in which clean-retrieval recall on the vulnerable class is substantially above zero. Adversarial item generation in this pipeline uses the same LLM family that is being evaluated, which is a white-box-adjacent assumption, and a fully black-box GenADV attacker of the kind used in the original study by Park and Lee [11] may behave differently, although given the floor effect observed here this variable was evidently not the limiting factor in

this particular run. Finally, the framework as implemented targets function-level vulnerability detection rather than the broader security-incident reconstruction task named in the original project brief, and this framing choice has been carried over unchanged from the pilot design.

9 Conclusion and Future Work

This paper combined two previously disjoint research threads, namely knowledge-level RAG for open-weight vulnerability detection as developed by Vul-RAG and its open-weight reproducibility study [7, 10], and adversarial robustness of retrieval-augmented language models as developed by Park and Lee [11], into a single, explicitly stated research gap together with a Colab implementation that was actually run on the real Vul-RAG Linux-kernel CVE benchmark. The pilot run with Qwen2.5-7B-Instruct produced a null robustness result that is best explained by a detector-calibration floor effect rather than by genuine adversarial robustness: the Matthews correlation coefficient for the vulnerable class is negative under clean, adversarial, and conflicting retrieval alike, showing the detector performs worse than chance on that class regardless of the retrieval condition, and a second model could not be evaluated because of a Hugging Face access-gating failure. Both outcomes are reported directly rather than smoothed over, since they are informative in their own right: they show that evaluating this threat model meaningfully requires first verifying that a knowledge-level detector's clean-retrieval recall is nontrivial, and second ensuring that more than one open-weight model family actually completes a run. The immediate next steps, detailed in Section 6, are to resolve the Llama access issue by accepting its license and authenticating, or to substitute an ungated model of comparable size such as Mistral-7B [64]; to diagnose and, if necessary, re-prompt the cause-check step so as to reduce its bias toward NOT_VULNERABLE; and to rerun the pipeline at ten to twenty CWE categories, twenty to fifty CVEs per category, and three to five open-weight model families, with attacker and detector deliberately drawn from different model families to test transferability, once clean-retrieval recall has been confirmed to be nontrivial, so that the adversarial, conflicting, and degraded-retrieval scenarios have a genuine signal available to degrade.

References

[1] M. Chen et al. Evaluating large language models trained on code. arXiv:2107.03374, 2021.

[2] B. Rozière et al. Code Llama: Open foundation models for code. arXiv:2308.12950, 2023.

[3] B. Hui et al. Qwen2.5-Coder technical report. arXiv:2409.12186, 2024.

[4] 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. Proceedings of the IEEE Symposium on Security and Privacy, 2024.

[5] Y. Ding, Y. Fu, O. Ibrahim, C. Sitawarin, X. Chen, B. Alomair, D. Wagner, B. Ray, and Y. Chen. Vulnerability detection with code language models: How far are we? arXiv:2403.18624, 2024.

[6] S. Chakraborty, R. Krishna, Y. Ding, and B. Ray. Deep learning based vulnerability detection: Are we there yet? IEEE Transactions on Software Engineering, 48(9), 2022.

[7] 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. ACM Transactions on Software Engineering and Methodology (TOSEM), 2026. https://doi.org/10.1145/3797277

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems (NeurIPS), 33:9459–9474, 2020.

[9] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey. arXiv:2312.10997, 2023.

[10] Anonymous. Revisiting Vul-RAG: Reproducibility and Replicability of RAG-based Vulnerability Detection with Open-Weight Models. arXiv:2606.04739, 2026.

[11] S.-I. Park and J.-Y. Lee. Toward Robust RALMs: Revealing the Impact of Imperfect Retrieval on Retrieval-Augmented Language Models. Transactions of the Association for Computational Linguistics (TACL), 12:1686–1702, 2024. https://doi.org/10.1162/tacl_a_00724

[12] O. Yoran, T. Wolfson, O. Ram, and J. Berant. Making retrieval-augmented language models robust to irrelevant context. Proceedings of ICLR, 2024.

[13] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou. Large language models can be easily distracted by irrelevant context. Proceedings of ICML, 2023.

[14] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 2023.

[15] W. Zou, R. Geng, B. Wang, and J. Jia. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. arXiv:2402.07867, 2024.
[16] Z. Zhong, Z. Huang, A. Wettig, and D. Chen. Poisoning retrieval corpora by injecting adversarial passages. Proceedings of EMNLP, 2023.
[17] J. Xue, M. Zheng, Y. Hu, F. Liu, X. Chen, and Q. Lou. BadRAG: Identifying vulnerabilities in retrieval augmented generation of large language models. arXiv:2406.00083, 2024.
[18] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, and Y. Zhong. VulDeePecker: A deep learning-based system for vulnerability detection. Proceedings of the Network and Distributed System Security Symposium (NDSS), 2018.
[19] Y. Zhou, S. Liu, J. Siow, X. Du, and Y. Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. Advances in Neural Information Processing Systems (NeurIPS), 32, 2019.
[20] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou. CodeBERT: A pre-trained model for programming and natural languages. Findings of EMNLP, 2020.
[21] M. Fu and C. Tantithamthavorn. LineVul: A transformer-based line-level vulnerability prediction. Proceedings of the IEEE/ACM International Conference on Mining Software Repositories (MSR), 2022.
[22] B. Steenhoek, H. Gao, and W. Le. Dataflow analysis-inspired deep learning for efficient vulnerability detection. Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE), 2024.
[23] J. Fan, Y. Li, S. Wang, and T. N. Nguyen. A C/C++ code vulnerability dataset with code changes and CVE summaries. Proceedings of MSR, 2020.
[24] G. Bhandari, A. Naseer, and L. Moonen. CVEfixes: Automated Collection of Vulnerabilities and Their Fixes from Open-Source Software. Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering, pages 30–39, 2021.
[25] Y. Chen, Z. Ding, L. Alowain, X. Chen, and D. Wagner. DiverseVul: A new vulnerable source code dataset for deep learning based vulnerability detection. Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID), 2023.
[26] X. Zhou, S. Cao, X. Sun, and D. Lo. Large language model for vulnerability detection and repair: Literature review and the road ahead. ACM Transactions on Software Engineering and Methodology, 2025.
[27] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri. Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. Proceedings of the IEEE Symposium on Security and Privacy, 2022.
[28] Y. Sun, D. Wu, Y. Xue, H. Liu, H. Wang, Z. Xu, X. Xie, and Y. Liu. GPTScan: Detecting logic vulnerabilities in smart contracts by combining GPT with program analysis. Proceedings of ICSE, 2024.
[29] A. Z. H. Yang, C. Le Goues, R. Martins, and V. Hellendoorn. Large language models for test-free fault localization. Proceedings of ICSE, 2024.
[30] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. REALM: Retrieval-augmented language model pre-training. Proceedings of the International Conference on Machine Learning (ICML), 2020.
[31] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. Proceedings of EMNLP, 2020.
[32] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. Proceedings of EACL, 2021.
[33] M. R. Parvez, W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang. Retrieval augmented code generation and summarization. Findings of EMNLP, 2021.
[34] F. Zhang, B. Chen, Y. Zhang, J. Keung, J. Liu, D. Zan, Y. Mao, J.-G. Lou, and W. Chen. RepoCoder: Repository-level code completion through iterative retrieval and generation. Proceedings of EMNLP, 2023.
[35] Z. Z. Wang, A. Asai, X. V. Yu, F. F. Xu, Y. Xie, G. Neubig, and D. Fried. CodeRAG-Bench: Can retrieval augment code generation? arXiv:2406.14497, 2024.
[36] F. Cuconasu, G. Trappolini, F. Siciliano, S. Filice, C. Campagnano, Y. Maarek, N. Tonellotto, and F. Silvestri. The power of noise: Redefining retrieval for RAG systems. Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, 2024.
[37] J. Xie, K. Zhang, J. Chen, R. Lou, and Y. Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. Proceedings of ICLR, 2024.
[38] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation. Proceedings of the AAAI Conference on Artificial Intelligence, 38, 2024.

[39] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. Proceedings of the International Conference on Learning Representations (ICLR), 2024.
[40] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec), 2023.
[41] F. Perez and I. Ribeiro. Ignore previous prompt: Attack techniques for language models. arXiv:2211.09527, 2022.
[42] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. Proceedings of ICLR, 2015.
[43] R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. Proceedings of EMNLP, 2017.
[44] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Universal adversarial triggers for attacking and analyzing NLP. Proceedings of EMNLP-IJCNLP, 2019.
[45] N. Yefet, U. Alon, and E. Yahav. Adversarial examples for models of code. Proceedings of the ACM on Programming Languages (OOPSLA), 4, 2020.
[46] R. Schuster, C. Song, E. Tromer, and V. Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion. Proceedings of the USENIX Security Symposium, 2021.
[47] SecureRAG-RTL: A Retrieval-Augmented, Multi-Agent, Zero-Shot LLM-Driven Framework for Hardware Vulnerability Detection. arXiv:2603.05689, 2026.
[48] W. Guo, S. Song, J. Guo, Z. Xu, C. Liu, H. Ou, M. Ge, and Y. Liu. Bridging Expert Reasoning and LLM Detection: A Knowledge-Driven Framework for Malicious Packages. Proceedings of the ACM Web Conference (WWW), 2026.
[49] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of EMNLP-IJCNLP, 2019.
[50] MITRE Corporation. Common Weakness Enumeration (CWE). https://cwe.mitre.org
[51] National Institute of Standards and Technology. National Vulnerability Database (NVD). https://nvd.nist.gov
[52] T. Wolf et al. Transformers: State-of-the-art natural language processing. Proceedings of EMNLP: System Demonstrations, 2020.
[53] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. Advances in Neural Information Processing Systems (NeurIPS), 36, 2023.
[54] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. Advances in Neural Information Processing Systems (NeurIPS), 35, 2022.
[55] Qwen Team, Alibaba Group. Qwen2.5 Technical Report. arXiv:2412.15115, 2024.
[56] AI at Meta. Llama 3.2: Revolutionizing Edge AI and Vision with Open, Customizable Models. Meta AI Blog, 2024.
[57] A. Dubey et al. The Llama 3 herd of models. arXiv:2407.21783, 2024.
[58] F. Pedregosa et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.
[59] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA) – Protein Structure, 405(2):442–451, 1975.
[60] D. Chicco and G. Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics, 21(1):6, 2020.
[61] E. B. Wilson. Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association, 22(158):209–212, 1927.
[62] L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. Statistical Science, 16(2):101–133, 2001.
[63] Gemma Team. Gemma: Open models based on Gemini research and technology. arXiv:2403.08295, 2024.
[64] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. Le Scao, T. Lavril, T. Wang, T. Lacroix, and W. El Sayed. Mistral 7B. arXiv:2310.06825, 2023.