VULREAD: Knowledge-Graph–guided Software Vulnerability Reasoning and Detection

Anonymous ACL submission

Abstract

Software vulnerability detection (SVD) remains a critical challenge in modern systems. While LLMs offer natural-language explanations alongside predictions, most work focuses on binary evaluation, and the resulting explanations often lack semantic consistency with Common Weakness Enumeration (CWE) categories. To address this gap, we propose **VULREAD**, a knowledge-graph–guided approach for vulnerability reasoning and detection that moves beyond binary classification toward CWE-level reasoning. VULREAD uses a security knowledge graph (KG) as a semantic backbone and a strong teacher LLM to generate CWE-consistent contrastive reasoning supervision, enabling student model training without manual annotation. Student models are fine-tuned with Odds Ratio Preference Optimization to encourage taxonomy-aligned reasoning while suppressing unsupported explanations. Across three real-world datasets, VULREAD improves binary F1 by 8–10% and multi-class classification by approximately 30% Macro-F1 and 18% Micro-F1 over state-of-the-art baselines, confirming that KG-guided reasoning significantly improves CWE coverage and detection accuracy in vulnerability analysis. Our source code and data are available at https://anonymous.4open.science/r/Vul-ReaD.

1 Introduction

Software vulnerabilities remain a persistent and critical threat to modern software systems, enabling attackers to compromise system integrity, leak sensitive information, and disrupt essential services. Despite decades of research in automated software vulnerability detection (SVD), vulnerabilities still appear at scale in real-world software, driving ongoing efforts to improve detection techniques.

Researchers have developed various SVD methods relying on pattern-based static analysis or deep learning (DL) models trained on historical vulnerability data (Chakraborty et al., 2021; Li et al., 2021, 2018), but these often struggle to generalize as they learn superficial patterns rather than meaningful vulnerability indicators (Ding et al., 2024). LLMs have since emerged as a promising alternative, leveraging strong code comprehension and generative capabilities for function-level vulnerability detection (Zibaeirad and Vieira, 2025; Hou et al., 2024; Shestov et al., 2025).

However, existing SVD evaluations are largely dominated by binary classification, which fails to capture whether a model truly understands a vulnerability's nature (Labruna et al., 2025). Effective mitigation requires identifying the underlying weakness – typically expressed as a CWE category – yet binary accuracy alone may substantially overestimate real-world utility. Studies exploring multi-class classification typically restrict evaluation to a limited CWE subset (Wen et al., 2024; Lekssays et al., 2025; Du et al., 2024; Cao et al., 2024; Yu et al., 2025), obscuring scalability challenges and leaving unclear how well models perform across a wider CWE space (Deng et al., 2024; Jiang et al., 2024; Weyssow et al., 2025). Moreover, while recent LLM-based methods increasingly generate explanations alongside predictions (Khare et al., 2025), their semantic alignment with correct CWE categories has not been rigorously examined (Weyssow et al., 2025) – leaving open whether enhanced reasoning reflects genuine vulnerability understanding or merely obscures misclassification errors.

In this work, we investigate whether LLM reasoning can improve multi-class vulnerability detection across diverse CWE categories, evaluating not only classification accuracy but also the semantic alignment of model reasoning with ground-truth vulnerability types. This involves two key challenges: (i) CWE distributions are highly imbalanced, with a long tail of rare vulnerability types; and (ii) LLM reasoning that is often fluent but inconsistently aligned with the correct CWE.

Our approach addresses both issues with a structured security knowledge graph (KG) that organizes CWEs through semantic abstractions and grounding model reasoning in explicit CWE relations rather than free-form intuition.

To realize this, we propose **VULREAD**, a novel knowledge-graph–guided approach for vulnerability reasoning and detection, inspired by the success of KGs in knowledge-intensive tasks (Wu et al., 2025a; Wang et al., 2025, 2024b; Labruna et al., 2025; Feng et al., 2024; Zhang et al., 2023; Simoni et al., 2025). Our design is motivated by two technical insights. First, preference optimization has emerged as an effective post-training paradigm for aligning LLMs with downstream task objectives, enabling models to favor structured, taxonomy-consistent reasoning over fluent but unsupported explanations. Second, a strong teacher LLM can automatically generate preference pairs grounded in KG semantics, providing scalable supervision without costly manual annotation, a critical advantage given the scarcity of expert-labeled vulnerability reasoning data (Croft et al., 2023; Wen et al., 2023).

Building on these insights, VULREAD explicitly targets CWE-level reasoning, where correct classification depends on understanding vulnerability semantics rather than surface code patterns. At its core, VULREAD employs a security KG that organizes vulnerability knowledge by relating CWEs through common code entities and conceptual security abstraction classes (Li et al., 2025), capturing hierarchical and semantic relationships that serve as a unifying structure across dataset construction, training, and inference. We augment real-world benchmarks with KG-derived attributes and use a strong teacher LLM to generate contrastive reasoning pairs per example: a CWE-aligned reasoning highlighting faulty constructs and vulnerability mechanisms, and a deliberately flawed counterpart produced via label flipping. Student LLMs are then adapted using parameter-efficient fine-tuning with Odds Ratio Preference Optimization (ORPO) (Hong et al., 2024), a preference optimization method that jointly combines a supervised fine-tuning objective on valid rationales with a contrastive odds-ratio term that favors CWE-consistent reasoning over flawed alternatives, suppressing hallucinated or misleading explanations.

We evaluate VULREAD through a comprehensive empirical study comparing DL models and LLM-based methods under binary and multi-class SVD settings. VULREAD (Qwen2.5-7B-Instruct) improves F1 by 4-11% over leading binary classifiers on PrimeVul (Ding et al., 2024), outperforms the reasoning-based baseline by 30% Macro-F1 and 18% Micro-F1 on average, and consistently surpasses all LLM baselines in CWE-level reasoning. Results confirm two key insights: (i) LLMs generally outperform DL models in binary SVD, and (ii) KG-guided reasoning substantially enhances CWE coverage and interpretability in multi-class detection. This paper makes the following contributions:

- **CWE-level reasoning and detection.** VULREAD moves beyond binary detection toward CWE-level diagnosis over a long-tailed vulnerability space.
- **Knowledge-guided dataset distillation.** We augment existing SVD benchmarks with structured KG attributes and paired CWE-consistent/inconsistent rationales, enabling more consistent and interpretable vulnerability reasoning without manual annotation.
- **Preference optimization for reasoning.** We combine parameter-efficient fine-tuning with ORPO to improve factual grounding and CWE consistency of LLM-generated explanations.
- **Empirical study.** We systematically evaluate baselines under binary and multi-class settings, demonstrating that VULREAD achieves strong binary detection while substantially improving CWE-level reasoning accuracy.

2 Motivation and Challenges

Recent LLM-based SVD systems show promising performance (Steenhoek et al., 2024a), yet even when correctly flagging vulnerable code, explanations are often poorly aligned with CWE categories. Figure 2 (Appendix A) illustrates this: without external knowledge guidance, a baseline LLM correctly detects a vulnerability but assigns CWE-763 instead of ground-truth CWE-401. This persists under ORPO training – preference optimization improves surface coherence but not CWE attribution accuracy. VULREAD correctly identifies CWE-401 for the same example, with Figure 3 (Appendix A) quantifying consistently low CWE-level reasoning coverage across baselines.

This highlights a critical gap: most SVD approaches optimize for binary correctness while neglecting explanation semantics, motivating CWE-

centered knowledge integration for more trustworthy vulnerability analysis.

Key challenges include: *semantic ambiguity* (code may map to multiple CWEs), *long-tailed label space* (many underrepresented categories), *unconstrained reasoning* (LLM explanations prone to hallucination), and *detection-explanation misalignment* (binary accuracy without correct attribution). These motivate structured security knowledge integration in training and inference.

Background on software vulnerability detection, vulnerability taxonomies, and knowledge graph representations is provided in Appendix B.

3 Problem Formulation

Task Definition We study the problem of SVD with *semantically correct* vulnerability reasoning. Let x denote a function-level code snippet (a sequence of tokens or lines). Given x, many existing SVD systems predict a binary label $y \in \{0, 1\}$, where $y = 1$ indicates that x contains at least one vulnerability.

Beyond binary detection, we consider *vulnerability cause identification*. Let $\mathcal{C}$ denote the set of CWE categories organized as a taxonomy. For a vulnerable function ($y = 1$), the SVD system is expected to predict one or more CWE IDs $c \in \mathcal{C}$ that characterize the underlying weakness.

LLM-Based Vulnerability Reasoning Recent approaches employ LLMs to analyze a code snippet x and generate both a vulnerability prediction y and a natural-language explanation r. Ideally, r should capture the root cause of the vulnerability and align with the predicted CWE-ID c. In practice, however, explanations often mischaracterize the cause or assign CWE labels only loosely related to the actual weakness (Figure 2). This mismatch underscores the need to explicitly model and evaluate the relationship between predictions, explanations, and standardized vulnerability semantics.

KG-Based Vulnerability Reasoning We assume access to a security knowledge graph $G = \{V, E\}$, where nodes V denote entities such as CWE categories, abstractions, and descriptive concepts, and edges E encode semantic relations (e.g., hierarchical or associative). Each CWE node is enriched with textual descriptions and metadata, providing a structured semantic space in which vulnerability concepts can be queried and reasoned about.

Problem Statement Given a function-level code snippet x, an LLM M, and a security KG G centered on CWE semantics, our objective is to design a detection approach that (i) predicts whether x is vulnerable, (ii) identifies the most appropriate CWE category, and (iii) produces explanations semantically anchored in G. We emphasize not only prediction accuracy but also the faithfulness and interpretability of reasoning, as a correct prediction accompanied by an incorrect or misleading explanation is considered a failure.

4 VULREAD

To address these challenges, we propose VULREAD, a knowledge-graph–guided approach for vulnerability reasoning and detection that integrates structured security knowledge into both LLM training and inference for explainable SVD. Figure 1 provides an overview of the VULREAD workflow. Given a labeled vulnerability dataset, we first construct a structured preference dataset by enriching each function with valid and flawed reasoning generated by a teacher LLM based on the custom KG. Therefore, unlike prior work that focuses on free-form natural language explanations (Weyssow et al., 2025), we extract explicit entities from functions mapped to security abstract classes.We then fine-tune a student model using ORPO to prefer structurally consistent reasoning over flawed alternatives. During inference, the KG serves as an external semantic guide, enabling CWE-consistent and security-aware predictions.

4.1 Security KG Construction

Inspired by (Li et al., 2025), we first construct a taxonomy of common software vulnerabilities by introducing conceptual vulnerability abstract classes $\mathcal{A}$ (e.g., File and Path Handling, Access Control, Memory Management) and function entities $\mathcal{B}$ (e.g., file paths, APIs, parameters).

- Vulnerability abstract classes define 13 conceptual categories of security flaws, forming an interpretable abstraction layer over hundreds of fine-grained CWE labels and ensuring full coverage of the official CWE corpus.
- Function entities are concrete programmatic elements whose presence or absence within a function serves as soft evidence for vulnerability mapping, as certain entities are strongly associated with specific flaw types while others signal their mitigation.

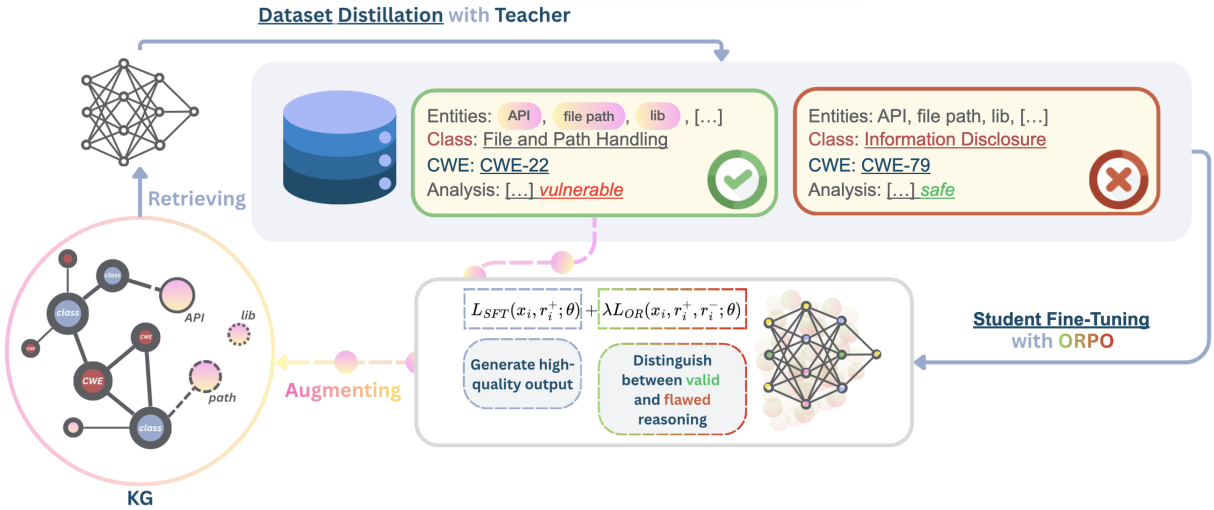

Figure 1: Overview of VULREAD. A security KG provides structured vulnerability semantics (entities, abstraction classes, CWE relations) to enrich each training example. A teacher LLM generates contrastive rationale pairs – CWE-consistent (valid) and CWE-inconsistent (flawed) – and a student is fine-tuned with ORPO, combining L_{SFT} on valid rationales with an odds-ratio term L_{OR} to favor CWE-aligned reasoning over flawed alternatives.

The main process of KG construction is to establish connections among vulnerability abstract classes $\mathcal{A}$, function entities $\mathcal{B}$, and CWEs $\mathcal{C}$. The relationship between $\mathcal{B}$ and $\mathcal{C}$ is directly inherited from the original data pairs $\{x, c\}$. Specifically, $b \in \mathcal{B}$ is usually extracted from x, and is associated with its corresponding CWE c.

To associate CWEs with abstract classes, we employ a hybrid mapping strategy that combines symbolic and semantic signals. In *Keyword-based Mapping*, each abstract class $a \in \mathcal{A}$ is represented by a curated set of security-related keywords K_a, which characterize its underlying vulnerability mechanism. Given a CWE description D_c, we compute the keyword matching score as the fraction of keywords in K_a that appear in D_c:

$$s_{\text{key}}(a, c) = \frac{1}{|K_a|} \sum_{k \in K_a} \mathbb{I}(k \in D_c). \quad (1)$$

In *Embedding-based Mapping*, we further capture semantic similarity beyond exact keyword matches. Specifically, we encode the abstract class description D_a and the CWE description D_c using a SentenceTransformer model F[1]. The embedding-based score is then computed by cosine similarity:

$$s_{\text{emb}}(a, c) = \frac{F(D_a)^\top F(D_c)}{\|F(D_a)\|_2 \, \|F(D_c)\|_2}. \quad (2)$$

We combine the two criteria to obtain the final matching score:

$$s = w s_{\text{key}} + (1 - w) s_{\text{emb}}. \quad (3)$$

[1] https://www.sbert.net/

When $s \geq 0.5$, the CWE c will be linked to the class a. Because many vulnerabilities exhibit overlapping characteristics, CWEs are allowed to map to multiple abstract classes. Our design preserves the multidimensional nature of security weaknesses rather than forcing overly coarse categorization.

The association between $\mathcal{A}$ and $\mathcal{B}$ is established through the intermediate linkage $\mathcal{A} \leftrightarrow \mathcal{C} \leftrightarrow \mathcal{B}$. This process results in the KG in Figure 1.

4.2 Reasoning Distillation

We employ Qwen2.5-32B-Instruct model as a teacher to generate structured vulnerability analyses for both vulnerable and non-vulnerable functions. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ denote a labeled dataset of x_i with binary vulnerability labels $y_i \in \{0, 1\}$. Following prior work on reasoning distillation for vulnerability detection (Weyssow et al., 2025; Wen et al., 2025), we employ a teacher LLM to generate contrastive reasoning pairs for each data sample. Specifically, the teacher LLM leverages the KG context together with the true label y_i to generate a valid reasoning path r_i^+, while a flawed counterpart r_i^- is constructed by label swapping $\tilde{y}_i$. Namely,

$$r_i^+ = \text{LLM}(x_i, y_i, G), r_i^- = \text{LLM}(x_i, \tilde{y}_i, G). \quad (4)$$

Then, we have the preference dataset $\mathcal{D}' = \{(x_i, y_i, r_i^+, r_i^-)\}_{i=1}^N$.

The teacher extracts function entities (e.g., file paths, APIs, parameters), associates them with vulnerability abstraction classes, and maps reason-

ing to CWE – encoding explicit *entity–class–CWE* alignments grounded in structured security knowledge rather than free-form intuition. KG guidance is incorporated by retrieving CWE nodes and associated abstraction classes from G based on entity matches within x_i, and injecting them into the teacher prompt as structured context. The teacher is explicitly prompted to follow the KG ontology rather than producing free-form reasoning, ensuring that generated rationales are grounded in KG-defined vulnerability semantics.

4.3 KG-guided Preference Optimization

We fine-tune a student LLM using ORPO over preference pairs $(x_i; r_i^+ \succ r_i^-)$, combining a supervised fine-tuning (SFT) objective on r_i^+ with a contrastive odds-ratio loss:

$$\mathcal{L}_{\text{ORPO}} = \mathcal{L}_{\text{SFT}}(x_i, r_i^+; \theta) + \lambda\, \mathcal{L}_{\text{OR}}(x_i, r_i^+, r_i^-; \theta), \tag{5}$$

$$\mathcal{L}_{\text{SFT}}(x_i, r_i^+; \theta) = -\frac{1}{n}\sum_{i=1}^{n} \log p_\theta(r_i \mid x_i), \tag{6}$$

$$\mathcal{L}_{\text{OR}} = -\frac{1}{n}\sum_{i=1}^{n} \log \sigma\left(\log \frac{\text{OR}_\theta(r_i^+ \mid x_i)}{\text{OR}_\theta(r_i^- \mid x_i)}\right), \tag{7}$$

where $\text{OR}_\theta(r \mid x) = \frac{\pi_\theta(r|x)}{1-\pi_\theta(r|x)}$. The odds-ratio term encourages entity–class–CWE-consistent reasoning while suppressing structurally incorrect but superficially plausible explanations, reinforcing causal reasoning over pattern memorization and reducing hallucinated security claims.

4.4 Knowledge Graph Augmentation

The KG contains two primary concept types: abstraction classes and CWE nodes. We iteratively enrich it by mining function-related entities from teacher-generated rationales – including function and variable names, API calls, libraries, and vulnerability indicators – linking them to existing classes and CWEs via co-occurrence patterns and semantic similarity. This integrates real-world code patterns into the KG, strengthening connections between abstract vulnerability definitions and concrete implementation artifacts.

5 Experimental Setup

We evaluate VULREAD across three research questions: **RQ1** examines binary vulnerability detection performance against DL and LLM-based baselines; **RQ2** assesses CWE-consistent reasoning quality and its translation into accurate multi-class classification across long-tailed CWE spaces; and **RQ3** measures the contribution of KG grounding – through CWE abstraction and entity relations – to both detection accuracy and reasoning reliability. Additionally, we verify that VULREAD gains stem from the training procedure in Section 6.3.

5.1 Datasets

Dataset Overview We evaluate VULREAD on three recent large-scale datasets: **DiverseVul** (Chen et al., 2023), **PrimeVul** (Ding et al., 2024), and **R2Vul** (Weyssow et al., 2025). All samples are derived from real-world GitHub security-fix commits, ensuring evaluated vulnerabilities reflect genuine development scenarios rather than synthetic patterns. Released between 2023 and 2025, the datasets collectively span a broad CWE space: DiverseVul covers 150 CWEs, PrimeVul 140, and R2Vul 267, with the latter introducing a long-tail distribution particularly challenging for fine-grained CWE classification.

Preprocessing and Dataset Balancing We align all datasets to 18,000 samples, matching the R2Vul corpus size. Given severe class imbalance in DiverseVul and PrimeVul (vulnerable-to-non-vulnerable ratios of 1:17 and 1:33), we downsample the majority class while preserving CWE diversity. Samples lacking explicit CWE identifiers are excluded to enable consistent CWE-level evaluation. All datasets are enriched with KG-guided reasoning pairs generated by the teacher Qwen2.5-32B-Instruct (Section 4.2) and split 8:1:1 for training, validation, and testing, applied consistently across SFT, ORPO training, and evaluation. For subset-aligned comparisons (Section 5.2), we use PrimeVul as the shared benchmark, as both LLMxCPG (Lekssays et al., 2025) and ReVD (Wen et al., 2025) derive their training data from PrimeVul and do not provide compatible splits for other datasets. Reconstructing their pipelines across additional benchmarks is beyond the scope of this study; evaluating on PrimeVul ensures a controlled comparison under identical test conditions.

5.2 Baselines

Traditional Deep Learning Baselines We include three established DL baselines representing sequence- (BiLSTM (Farasat and Posegga, 2024)), semantic- (SySeVR (Li et al., 2021)), and graph-centric (Devign (Zhou et al., 2019)), and GNN-based (LIVABLE (Wen et al., 2024)) paradigms, re-

Table 1: Binary performance comparison with Baselines. Underlined values indicate the best performance within each model family. **Bold** values indicate the best performance across all models for a given dataset and metric.

Baselines		DiverseVul			R2Vul			PrimeVul		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
	BiLSTM	0.50	0.70	0.59	0.55	0.72	0.62	0.45	0.68	0.55
	SySeVR	0.15	0.25	0.19	0.08	0.15	0.11	0.12	0.20	0.15
	Devign	0.20	0.30	0.24	0.22	0.32	0.26	0.14	0.22	0.17
	LIVABLE	0.37	0.46	0.41	0.42	0.51	0.46	0.34	0.43	0.38
LLaMA	Zero-Shot	0.22	0.55	0.31	0.28	0.64	0.39	0.21	0.44	0.28
	ICL	0.30	0.64	0.41	0.39	0.61	0.48	0.33	0.50	0.41
	COT	0.36	0.76	0.49	0.41	0.73	0.53	0.39	0.51	0.44
	SFT	0.45	0.63	0.53	0.52	0.68	0.59	0.40	0.58	0.48
	VULREAD	0.66	0.68	0.67	0.67	0.69	0.68	0.57	0.59	0.58
DeepSeek	Zero-Shot	0.15	0.66	0.24	0.18	0.70	0.29	0.20	0.69	0.31
	ICL	0.21	0.75	0.39	0.26	0.75	0.39	0.26	0.74	0.38
	COT	0.23	0.79	0.36	0.32	0.74	0.48	0.28	0.80	0.41
	SFT	0.42	0.63	0.51	0.44	0.63	0.52	0.41	0.64	0.51
	VULREAD	0.64	0.66	0.65	**0.69**	0.71	0.70	0.59	0.61	0.60
Qwen	Zero-Shot	0.25	0.50	0.33	0.21	0.63	0.32	0.30	0.72	0.42
	ICL	0.37	0.67	0.48	0.41	0.72	0.52	0.35	**0.77**	0.48
	COT	0.38	0.60	0.47	0.46	0.75	0.57	0.36	0.75	0.48
	SFT	0.50	0.73	0.60	0.58	0.79	0.67	0.40	0.62	0.49
	R2Vul	0.56	0.72	0.63	0.56	**0.90**	0.69	0.50	0.68	0.58
	VULREAD	**0.78**	**0.80**	**0.79**	0.68	0.88	**0.77**	**0.69**	0.67	**0.68**

tained as standard reference points consistent with prior SVD benchmarking practice.

LLM-Based Models We evaluate three instruction-tuned code LLMs of comparable scale: Qwen2.5-7B-Instruct, LLaMA-3.1-8B-Instruct, and DeepSeek-Coder-7B-Instruct, each under four adaptation strategies: zero-shot, in-context learning (ICL), chain-of-thought (CoT), and QLoRA-based SFT. Prompting templates and fine-tuning configurations follow related work (Bi et al., 2023). Reasoning-oriented LLMs are excluded as fine-tuned backbones, as they are optimized for inference-time generation rather than supervised fine-tuning.

State-of-the-art SVD Baselines We compare against three recent LLM-based SVD methods: R2Vul (Weyssow et al., 2025), ReVD (Wen et al., 2025), and LLMxCPG (Lekssays et al., 2025), representing state-of-the-art approaches via structured reasoning, curriculum-based preference optimization, and code property graph guidance respectively. LLMxCPG and ReVD are evaluated under subset-aligned settings for RQ1 only, as their custom preprocessing pipelines are incompatible with multi-CWE evaluation. R2Vul is included in both RQ1 and RQ2 due to its reasoning-based nature.

Evaluation Metrics For RQ1, we report Precision, Recall, and F1-score. For RQ2, we report Micro- and Macro-F1/Precision/Recall. Micro-F1 aggregates across all classes and is dominated by frequent CWEs, while Macro-F1 averages equally across all classes regardless of frequency – making it particularly important for long-tail CWE assessment. For LLM-based models, CWE predictions are extracted from generated reasoning text – reliable by design, as ORPO fine-tuning trains the student to consistently embed a CWE identifier whenever a vulnerability is detected, ensuring extracted IDs directly reflect classification decisions. DL baselines are evaluated solely on classifier outputs. RQ3 uses the same metrics for with/without-KG comparisons.

6 Results and Discussion

6.1 RQ1: Overall Performance

Table 1 summarizes binary detection performance. DL baselines achieve substantially lower F1-scores (0.11–0.26 on R2Vul, 0.15–0.24 on PrimeVul), reflecting limited generalization beyond surface patterns. General-purpose LLMs without task-specific training already improve markedly (F1: 0.33–0.48 zero-shot), with SFT pushing further to 0.67. VULREAD achieves the strongest results across all settings: integrated with Qwen, it attains F1-scores of 0.79, 0.77, and 0.68 on DiverseVul, R2Vul, and PrimeVul, respectively, surpassing both SFT and R2Vul. Consistent gains across LLaMA and DeepSeek backbones confirm that KG guidance transfers across student architectures.

Table 2: Binary performance under restricted CWE subsets with LLMxCPG and ReVD.

Model	Precision	Recall	F1
LLMxCPG	**86%**	55%	67%
VULREAD	83%	**63%**	**71%**
Model	**Precision**	**Recall**	**F1**
ReVD	58%	64%	53%
VULREAD	**65%**	**66%**	**64%**

For subset-aligned comparisons, in Table 2, VULREAD achieves F1 of 71.4% vs. 67.2% for LLMxCPG, and 64% vs. 53% for ReVD on PrimeVul, with consistent improvements in both precision and recall, confirming superiority under constrained evaluation settings. **RQ1 Summary:** VULREAD consistently achieves the highest binary F1-scores across datasets and models, outperforming both DL and LLM-based SVD baselines.

6.2 RQ2: CWE-level Reasoning

Reasoning-based CWE Classification Table 5 reports Macro- and Micro-F1 for LLM-based models, where CWE predictions are extracted from generated reasoning text. VULREAD consistently achieves the highest scores across all datasets: 34.4%/23.4% on DiverseVul, 29.9%/25.2% on R2Vul, and 35.6%/22.3% on PrimeVul, outperforming all LLM baselines by a clear margin and demonstrating more reliable CWE identification through reasoning.

DL Multi-Class Performance DL baselines (BiLSTM, SySeVR, Devign, LIVABLE) are evaluated solely on classifier outputs and perform near-zero (Macro-F1 <0.0005, Micro-F1 <0.001) across all datasets, while LIVABLE achieves marginally higher scores (Macro-F1 ≈ 0.02, Micro-F1 ≈ 0.03), remaining far below LLM-based approaches. BiLSTM achieves the best results and is the only DL baseline reported in Table 7 (Appendix D), as others prevent meaningful comparison. Their failure to capture semantic and structural diversity underscores the advantage of reasoning-driven LLM approaches. **RQ2 Summary:** VULREAD achieves the highest Macro- and Micro-F1 across all datasets, while DL baselines perform near-zero in CWE-level classification.

6.3 RQ3: KG-Guidance Contribution.

Table 3 compares VULREAD with and without KG on R2Vul, using the original prompt design from (Weyssow et al., 2025) as the KG-free baseline. KG integration improves binary detection (F1: 0.64 $\rightarrow$ 0.77; precision: 0.52 $\rightarrow$ 0.67; recall: 0.84 $\rightarrow$ 0.88) with more pronounced gains in CWE-level reasoning (Macro-F1: 0.05 $\rightarrow$ 0.30; Micro-F1: 0.07 $\rightarrow$ 0.25), confirming that structured knowledge guidance drives both more accurate vulnerability detection and reliable CWE identification. **RQ3 Summary:** KG integration contributes the most to performance, with Macro-F1 improving from 0.05 to 0.30.

Table 3: Impact of KG.

	Precision	Recall	F1-score	Macro-F1	Micro-F1
W/o KG	0.52	0.84	0.64	0.05	0.07
With KG	**0.67**	**0.88**	**0.77**	**0.30**	**0.25**

Ablation: Distillation vs. KG Retrieval A natural concern is whether VULREAD's gains arise from training or simply from access to KG context at inference. To address this, we compare against the teacher model prompted with KG context at test time. Under matched KG conditions, VULREAD consistently outperforms the teacher across all datasets, demonstrating that the distillation process contributes signal beyond retrieval alone. On R2Vul, the teacher achieves an F1 of 0.52, a Micro-F1 of 0.13, and a Macro-F1 of 0.06, whereas VULREAD attains 0.77, 0.25, and 0.30 respectively (Table 4). More detailed results for PrimeVul and DiverseVul are provided in Appendix C.1, where the gap remains consistent. These findings directly demonstrate that the training pipeline contributes signal beyond inference-time retrieval.

Table 4: Teacher Model vs. VULREAD on R2Vul

	F1-score	Macro F1	Micro F1
Qwen-32B (Teacher + KG)	0.52	0.06	0.13
VULREAD	**0.77**	**0.30**	**0.25**

We discuss other DL baseline behavior, KG retrieval impact, knowledge guidance components, and future backbone directions in Appendix D.

7 Related Work

Vulnerability Detection DL-based approaches achieve reasonable binary classification but operate

Table 5: Multi-Class CWE Classification.

Baseline		DiverseVul		R2Vul		PrimeVul	
		Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
LLaMA	SFT	0.021	0.043	0.032	0.044	0.023	0.050
	VULREAD	**0.313**	**0.192**	**0.208**	**0.187**	**0.337**	**0.207**
DeepSeek	SFT	0.033	0.018	0.020	0.064	0.026	0.057
	VULREAD	**0.331**	**0.188**	**0.230**	**0.219**	**0.359**	**0.213**
Qwen	SFT	0.019	0.057	0.028	0.047	0.031	0.049
	R2Vul	0.030	0.063	0.028	0.067	0.036	0.059
	VULREAD	**0.344**	**0.234**	**0.299**	**0.252**	**0.356**	**0.223**

as black-box predictors with no insight into vulnerability causes (Zhou et al., 2019; Wen et al., 2024; Li et al., 2018; Hin et al., 2022; Wartschinski et al., 2022; Farasat and Posegga, 2024; Rahman et al., 2024; Cao et al., 2024). LLMs have since been explored via prompting and fine-tuning, enabling textual rationales and fix suggestions alongside detection (Sheng et al., 2024; Zhang et al., 2024; Ding et al., 2024; Yang et al., 2025; Gonçalves et al., 2024; Nong et al., 2022; Tamberg and Bahsi, 2025; Wen et al., 2025), with some works augmenting SVD with external knowledge yet still evaluating primarily on binary classification (Nguyen et al., 2025; Lekssays et al., 2025; Du et al., 2024; Zheng et al., 2025; Sun et al., 2024; Wang et al., 2024a; Deng et al., 2024). Attempts at CWE-type classification cover only limited subsets (Wen et al., 2024; Gao et al., 2023; Mirsky et al., 2023), and explanation-generating approaches remain vague or detached from vulnerability categories (Ullah et al., 2024; Yin et al., 2024b; Mao et al., 2024; Ni et al., 2023; Steenhoek et al., 2024b; Zibaeirad and Vieira, 2025; Li et al., 2024; Jararweh et al., 2025). R2Vul (Weyssow et al., 2025) is the closest prior work, introducing CWE-aligned reasoning via preference optimization, yet its reasoning is frequently incorrect and scope is limited to binary classification. VULREAD addresses this gap by grounding detection and reasoning in a structured security KG for reliable CWE attribution across a long-tailed vulnerability space.

Knowledge Graphs in the Vulnerability Domain Vulnerability KGs have been applied to retrieval and similarity search (Sun et al., 2023; Dong et al., 2024), entity alignment (Yan et al., 2023), risk analysis (Yin et al., 2024a), and link-prediction modeling (Liu et al., 2025a), demonstrating advantages in representing relational dependencies that flat feature representations cannot capture (Wu et al., 2025a; Wang et al., 2025). Despite this, existing KGs are rarely integrated into SVD pipelines (Wang et al., 2024c) leaving the role of explicit semantic relations such as CWE hierarchies largely unexplored. VULREAD addresses this gap by using the KG as a semantic backbone across dataset construction, training, and inference.

8 Conclusion

This paper introduces VULREAD, a knowledge-graph–guided approach to software vulnerability reasoning and detection that injects structured security knowledge into dataset construction, training, and inference for CWE-level reasoning over long-tailed vulnerability distributions. Through KG-informed distillation and ORPO-based preference optimization, VULREAD improves the factual accuracy and CWE consistency of LLM-generated explanations – demonstrating that LLMs outperform DL baselines in binary SVD, and that KG guidance is critical for multi-class CWE reasoning, substantially improving semantic consistency over SFT-only and R2Vul baselines. The backbone-agnostic nature of VULREAD's gains suggests that applying KG-guided preference optimization to reasoning-oriented architectures represents a promising direction for future work, alongside refinement of CWE taxonomies, improved KG retrieval, and extension to broader security tasks.

Limitations

CWE classification Our KG organizes CWEs into 13 abstract classes inspired by MITRE documentation and prior work (Li et al., 2025). However, this taxonomy is not fixed and alternative

groupings may be more precise or less granular. The assignment of CWEs to these classes is based on keyword and behavioral cues, which introduces the possibility of misclassification. As a result, the reported improvements in reasoning may depend on a chosen abstraction scheme.

Retrieved KG Context At inference time, KG retrieval occasionally returns extensive contextual information, including multiple CWE candidates associated with extracted entities. In such cases, the model may mention several CWEs and conclude that the function could relate to any of them. To mitigate this, we incorporated confidence levels based on entity frequency, but this approach remains limited. Future work will explore more refined retrieval strategies to reduce noise and improve precision.

Dual-use risk VulReaD is designed to assist developers in identifying and understanding software vulnerabilities for remediation purposes. However, as with any vulnerability detection system, the generated explanations and CWE attributions could potentially be misused to facilitate exploitation rather than defense. We encourage responsible deployment in controlled development environments and advocate for access restrictions in adversarial settings.

References

Sean Barnum. 2012. Standardizing cyber threat intelligence information with the structured threat information expression (stix). *Mitre Corporation*, 11(2012):1–22.

Yingzhou Bi, Jiangtao Huang, Penghui Liu, and Lianmei Wang. 2023. Benchmarking software vulnerability detection techniques: A survey. *arXiv preprint arXiv:2303.16362*.

Sicong Cao, Xiaobing Sun, Xiaoxue Wu, David Lo, Lili Bo, Bin Li, Xiaolei Liu, Xingwei Lin, and Wei Liu. 2024. Snopy: Bridging sample denoising with causal graph learning for effective vulnerability detection. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 606–618.

Saikat Chakraborty, Rahul Krishna, Yangruibo Ding, and Baishakhi Ray. 2021. Deep learning based vulnerability detection: Are we there yet? *IEEE Transactions on Software Engineering*, 48(9):3280–3296.

Yizheng Chen, Zhoujie Ding, Lamya Alowain, Xinyun Chen, and David Wagner. 2023. Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 654–668.

Roland Croft, M Ali Babar, and M Mehdi Kholoosi. 2023. Data quality for software vulnerability datasets. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 121–133. IEEE.

Xiao Deng, Fuyao Duan, Rui Xie, Wei Ye, and Shikun Zhang. 2024. Improving long-tail vulnerability detection through data augmentation based on large language models. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 262–274. IEEE.

Yangruibo Ding, Yanjun Fu, Omniyyah Ibrahim, Chawin Sitawarin, Xinyun Chen, Basel Alomair, David Wagner, Baishakhi Ray, and Yizheng Chen. 2024. Vulnerability detection with code language models: How far are we? *arXiv preprint arXiv:2403.18624*.

Junzhe Dong, Shuo Geng, and Xiong Zhang. 2024. An intelligent retrieval system for similar information system vulnerabilities based on knowledge graph. In *2024 IEEE 9th International Conference on Data Science in Cyberspace (DSC)*, pages 171–176. IEEE.

Xueying Du, Geng Zheng, Kaixin Wang, Yi Zou, Yujia Wang, Wentai Deng, Jiayi Feng, Mingwei Liu, Bihuan Chen, Xin Peng, and 1 others. 2024. Vul-rag: Enhancing llm-based vulnerability detection via knowledge-level rag. *arXiv preprint arXiv:2406.11147*.

Talaya Farasat and Joachim Posegga. 2024. Machine learning techniques for python source code vulnerability detection. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy*, pages 151–153.

Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, and Yulia Tsvetkov. 2024. Don't hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. *arXiv preprint arXiv:2402.00367*.

Stefan Fenz and Andreas Ekelhart. 2009. Formalizing information security knowledge. In *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*, pages 183–194.

Zeyu Gao, Hao Wang, Yuchen Zhou, Wenyu Zhu, and Chao Zhang. 2023. How far have we gone in vulnerability detection using large language models. *arXiv preprint arXiv:2311.12420*.

José Gonçalves, Tiago Dias, Eva Maia, and Isabel Praça. 2024. Scope: Evaluating llms for software vulnerability detection. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 34–43. Springer.

Sabaat Haroon, Ahmad Faraz Khan, Ahmad Humayun, Waris Gill, Abdul Haddi Amjad, Ali R Butt, Mohammad Taha Khan, and Muhammad Ali Gulzar. 2025. How accurately do large language models understand code? *arXiv preprint arXiv:2504.04372*.

David Hin, Andrey Kan, Huaming Chen, and M Ali Babar. 2022. Linevd: Statement-level vulnerability detection using graph neural networks. In *Proceedings of the 19th international conference on mining software repositories*, pages 596–607.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189.

Anders Høst, Pierre Lison, and Leon Moonen. 2023. Constructing a knowledge graph from textual descriptions of software vulnerabilities in the national vulnerability database. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 386–391.

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8):1–79.

Ala Jararweh, Michael Adams, Avinash Sahu, Abdullah Mueen, and Afsah Anwar. 2025. Llavul: A multimodal llm for interpretable vulnerability reasoning about source code. In *2025 5th Intelligent Cybersecurity Conference (ICSC)*, pages 232–241. IEEE.

Xuefeng Jiang, Lvhua Wu, Sheng Sun, Jia Li, Jingjing Xue, Yuwei Wang, Tingting Wu, and Min Liu. 2024. Investigating large language models for code vulnerability detection: An experimental study. *arXiv preprint arXiv:2412.18260*.

Avishree Khare, Saikat Dutta, Ziyang Li, Alaia Solko-Breslin, Rajeev Alur, and Mayur Naik. 2025. Understanding the effectiveness of large language models in detecting security vulnerabilities. In *2025 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 103–114. IEEE.

Tiziano Labruna, Jon Ander Campos, and Gorka Azkune. 2025. When to retrieve: Teaching llms to utilize information retrieval effectively. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing-Natural Language Processing in the Generative AI Era*, pages 623–632.

Ahmed Lekssays, Hamza Mouhcine, Khang Tran, Ting Yu, and Issa Khalil. 2025. {LLMxCPG}:{Context-Aware} vulnerability detection through code property {Graph-Guided} large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 489–507.

Xinghang Li, Jingzhe Ding, Chao Peng, Bing Zhao, Xiang Gao, Hongwan Gao, and Xinchen Gu. 2025. Safegenbench: A benchmark framework for security vulnerability detection in llm-generated code. *arXiv preprint arXiv:2506.05692*.

Zhen Li, Deqing Zou, Shouhuai Xu, Hai Jin, Yawei Zhu, and Zhaoxuan Chen. 2021. Sysevr: A framework for using deep learning to detect software vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2244–2258.

Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. 2018. Vuldeepecker: A deep learning-based system for vulnerability detection. *arXiv preprint arXiv:1801.01681*.

Ziyang Li, Saikat Dutta, and Mayur Naik. 2024. Iris: Llm-assisted static analysis for detecting security vulnerabilities. *arXiv preprint arXiv:2405.17238*.

Ruitong Liu, Yaxuan Xie, Zexu Dang, Jinyi Hao, Xiaowen Quan, Yongcai Xiao, and Chunlei Peng. 2025a. Dynamic vulnerability knowledge graph construction via multi-source data fusion and large language model reasoning. *Electronics*, 14(12):2334.

Xiaojian Liu, Xinwei Guo, and Wen Gu. 2025b. Seckg2vec: A novel security knowledge graph relational reasoning method based on semantic and structural fusion embedding. *Computers & Security*, 149:104192.

Qiheng Mao, Zhenhao Li, Xing Hu, Kui Liu, Xin Xia, and Jianling Sun. 2024. Towards effectively detecting and explaining vulnerabilities using large language models. *arXiv e-prints*, pages arXiv–2406.

Yisroel Mirsky, George Macon, Michael Brown, Carter Yagemann, Matthew Pruett, Evan Downing, Sukarno Mertoguno, and Wenke Lee. 2023. {VulChecker}: Graph-based vulnerability localization in source code. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6557–6574.

Inoussa Mouiche and Sherif Saad. 2025. Entity and relation extractions for threat intelligence knowledge graphs. *Computers & Security*, 148:104120.

Van Nguyen, Surya Nepal, Xingliang Yuan, Tingmin Wu, and Carsten Rudolph. 2025. Safe: A novel approach for software vulnerability detection from enhancing the capability of large language models. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, pages 392–406.

Chao Ni, Xin Yin, Kaiwen Yang, Dehai Zhao, Zhenchang Xing, and Xin Xia. 2023. Distinguishing look-alike innocent and vulnerable code by subtle semantic representation learning and explanation. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1611–1622.

Yu Nong, Rainy Sharma, Abdelwahab Hamou-Lhadj, Xiapu Luo, and Haipeng Cai. 2022. Open science in software engineering: A study on deep learning-based vulnerability detection. *IEEE Transactions on Software Engineering*, 49(4):1983–2005.

Md Mahbubur Rahman, Ira Ceka, Chengzhi Mao, Saikat Chakraborty, Baishakhi Ray, and Wei Le. 2024. Towards causal deep learning for vulnerability detection. In *Proceedings of the IEEE/ACM 46th international conference on software engineering*, pages 1–11.

Rebecca Russell, Louis Kim, Lei Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul Ellingwood, and Marc McConley. 2018. Automated vulnerability detection in source code using deep representation learning. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 757–762. IEEE.

Ze Sheng, Fenghua Wu, Xiangwu Zuo, Chao Li, Yuxin Qiao, and Lei Hang. 2024. Lprotector: An llm-driven vulnerability detection system. *arXiv preprint arXiv:2411.06493*.

Aleksei Shestov, Rodion Levichev, Ravil Mussabayev, Evgeny Maslov, Pavel Zadorozhny, Anton Cheshkov, Rustam Mussabayev, Alymzhan Toleu, Gulmira Tolegen, and Alexander Krassovitskiy. 2025. Finetuning large language models for vulnerability detection. *IEEE Access*.

Zhenpeng Shi, Nikolay Matyunin, Kalman Graffi, and David Starobinski. 2024. Uncovering cwe-cve-cpe relations with threat knowledge graphs. *ACM Transactions on Privacy and Security*, 27(1):1–26.

Marco Simoni, Andrea Saracino, Vinod P, and Mauro Conti. 2025. Morse: Bridging the gap in cybersecurity expertise with retrieval augmented generation. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, pages 1213–1222.

Benjamin Steenhoek, Md Mahbubur Rahman, Monoshi Kumar Roy, Mirza Sanjida Alam, Earl T Barr, and Wei Le. 2024a. A comprehensive study of the capabilities of large language models for vulnerability detection. *CoRR*.

Benjamin Steenhoek, Md Mahbubur Rahman, Monoshi Kumar Roy, Mirza Sanjida Alam, Hengbo Tong, Swarna Das, Earl T Barr, and Wei Le. 2024b. To err is machine: Vulnerability detection challenges llm reasoning. *arXiv preprint arXiv:2403.17218*.

Shaznin Sultana, Sadia Afreen, and Nasir U Eisty. 2024. Code vulnerability detection: A comparative analysis of emerging large language models. *arXiv preprint arXiv:2409.10490*.

Jiamou Sun, Zhenchang Xing, Qinghua Lu, Xiwei Xu, and Liming Zhu. 2023. A multi-faceted vulnerability searching website powered by aspect-level vulnerability knowledge graph. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 60–63. IEEE.

Yuqiang Sun, Daoyuan Wu, Yue Xue, Han Liu, Wei Ma, Lyuye Zhang, Yang Liu, and Yingjiu Li. 2024. Llm4vuln: A unified evaluation framework for decoupling and enhancing llms' vulnerability reasoning. *arXiv preprint arXiv:2401.16185*.

Zareen Syed, Ankur Padia, M Lisa Mathews, Tim Finin, Anupam Joshi, and 1 others. 2016. Uco: A unified cybersecurity ontology. In *Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security*, pages 195–202.

Karl Tamberg and Hayretdin Bahsi. 2025. Harnessing large language models for software vulnerability detection: A comprehensive benchmarking study. *IEEE Access*.

Saad Ullah, Mingji Han, Saurabh Pujar, Hammond Pearce, Ayse Coskun, and Gianluca Stringhini. 2024. Llms cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks. In *2024 IEEE symposium on security and privacy (SP)*, pages 862–880. IEEE.

Chengpeng Wang, Wuqi Zhang, Zian Su, Xiangzhe Xu, Xiaoheng Xie, and Xiangyu Zhang. 2024a. Llmdfa: analyzing dataflow in code with large language models. *Advances in Neural Information Processing Systems*, 37:131545–131574.

Junjie Wang, Mingyang Chen, Binbin Hu, Dan Yang, Ziqi Liu, Yue Shen, Peng Wei, Zhiqiang Zhang, Jinjie Gu, Jun Zhou, and 1 others. 2024b. Learning to plan for retrieval-augmented large language models from knowledge graphs. *arXiv preprint arXiv:2406.14282*.

Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge graph retrieval-augmented generation for llm-based recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27152–27168.

Yechao Wang, Wuqiang Shen, Chaosheng Yao, and Zitong Zhang. 2024c. Research on software vulnerability information knowledge graph construction method based on text mining. In *2024 6th International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 57–65. IEEE.

Laura Wartschinski, Yannic Noller, Thomas Vogel, Timo Kehrer, and Lars Grunske. 2022. Vudenc: vulnerability detection with deep learning on a natural codebase for python. *Information and Software Technology*, 144:106809.

Xin-Cheng Wen, Cuiyun Gao, Feng Luo, Haoyu Wang, Ge Li, and Qing Liao. 2024. Livable: exploring long-tailed classification of software vulnerability types. *IEEE Transactions on Software Engineering*, 50(6):1325–1339.

Xin-Cheng Wen, Xinchen Wang, Cuiyun Gao, Shaohua Wang, Yang Liu, and Zhaoquan Gu. 2023. When less is enough: Positive and unlabeled learning model for vulnerability detection. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 345–357. IEEE.

Xin-Cheng Wen, Yijun Yang, Cuiyun Gao, Yang Xiao, and Deheng Ye. 2025. Boosting vulnerability detection of llms via curriculum preference optimization with synthetic reasoning data. *arXiv preprint arXiv:2506.07390*.

Martin Weyssow, Chengran Yang, Junkai Chen, Ratnadira Widyasari, Ting Zhang, Huihui Huang, Huu Hung Nguyen, Yan Naing Tun, Tan Bui, Yikun Li, and 1 others. 2025. R2vul: Learning to reason about software vulnerabilities with reinforcement learning and structured reasoning distillation. *arXiv preprint arXiv:2504.04699*.

Wenjie Wu, Yongcheng Jing, Yingjie Wang, Wenbin Hu, and Dacheng Tao. 2025a. *arXiv preprint arXiv:2503.01642*.

Xiaoxue Wu, Shiyu Weng, Bin Zheng, Wei Zheng, Xiang Chen, and Xiaobin Sun. 2025b. Ng_mderank: A software vulnerability feature knowledge extraction method based on n-gram similarity. *Journal of Software: Evolution and Process*, 37(1):e2727.

Qi Yan, Mohan Li, and Yanbin Sun. 2023. Neighborhood matching entity alignment model for vulnerability knowledge graphs. In *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 38–45. IEEE.

Yanjing Yang, Xin Zhou, Runfeng Mao, Jinwei Xu, Lanxin Yang, Yu Zhang, Haifeng Shen, and He Zhang. 2025. Dlap: A deep learning augmented large language model prompting framework for software vulnerability detection. *Journal of Systems and Software*, 219:112234.

Jiao Yin, Wei Hong, Hua Wang, Jinli Cao, Yuan Miao, and Yanchun Zhang. 2024a. A compact vulnerability knowledge graph for risk assessment. *ACM Transactions on Knowledge Discovery from Data*, 18(8):1–17.

Xin Yin, Chao Ni, and Shaohua Wang. 2024b. Multitask-based evaluation of open-source llm on software vulnerability. *IEEE Transactions on Software Engineering*.

Junji Yu, Honglin Shu, Michael Fu, Dong Wang, Chakkrit Tantithamthavorn, Yasutaka Kamei, and Junjie Chen. 2025. A preliminary study of large language models for multilingual vulnerability detection. In *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 161–168.

Chenyuan Zhang, Hao Liu, Jiutian Zeng, Kejing Yang, Yuhong Li, and Hui Li. 2024. Prompt-enhanced software vulnerability detection using chatgpt. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pages 276–277.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve anything to augment large language models. *arXiv preprint arXiv:2310.07554*.

Xinran Zheng, Xingzhi Qian, Huichi Zhou, Shuo Yang, Yiling He, Suman Jana, and Lorenzo Cavallaro. 2025. Learning to focus: Context extraction for efficient code vulnerability detection with language models. *arXiv preprint arXiv:2505.17460*.

Xin Zhou, Duc-Manh Tran, Thanh Le-Cong, Ting Zhang, Ivana Clairine Irsan, Joshua Sumarlin, Bach Le, and David Lo. 2024. Comparison of static application security testing tools and large language models for repo-level vulnerability detection. *arXiv preprint arXiv:2407.16235*.

Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Advances in neural information processing systems*, 32.

Arastoo Zibaeirad and Marco Vieira. 2025. Reasoning with llms for zero-shot vulnerability detection. *arXiv preprint arXiv:2503.17885*.

A Motivation

Figure 2 illustrates a failure case of baseline R2Vul on CWE-level reasoning. Given a function labeled CWE-401 (Memory Leak) (top), R2Vul generates a fluent rationale that incorrectly attributes the issue to CWE-763 (Use-After-Free) and introduces an unrelated CVE reference (bottom), illustrating semantic misalignment between generated explanations and the target CWE.

B Background

B.1 Software Vulnerability Detection

SVD aims to identify exploitable defects in source code. Learning-based approaches predominantly operate on function- or statement-level representations and frame the task as binary classification (Chakraborty et al., 2021; Russell et al., 2018; Rahman et al., 2024). While competitive in detection performance, these models offer limited interpretability and do not explain underlying vulnerability causes (Ni et al., 2023; Zhou et al., 2024). LLMs pretrained on large-scale code corpora have

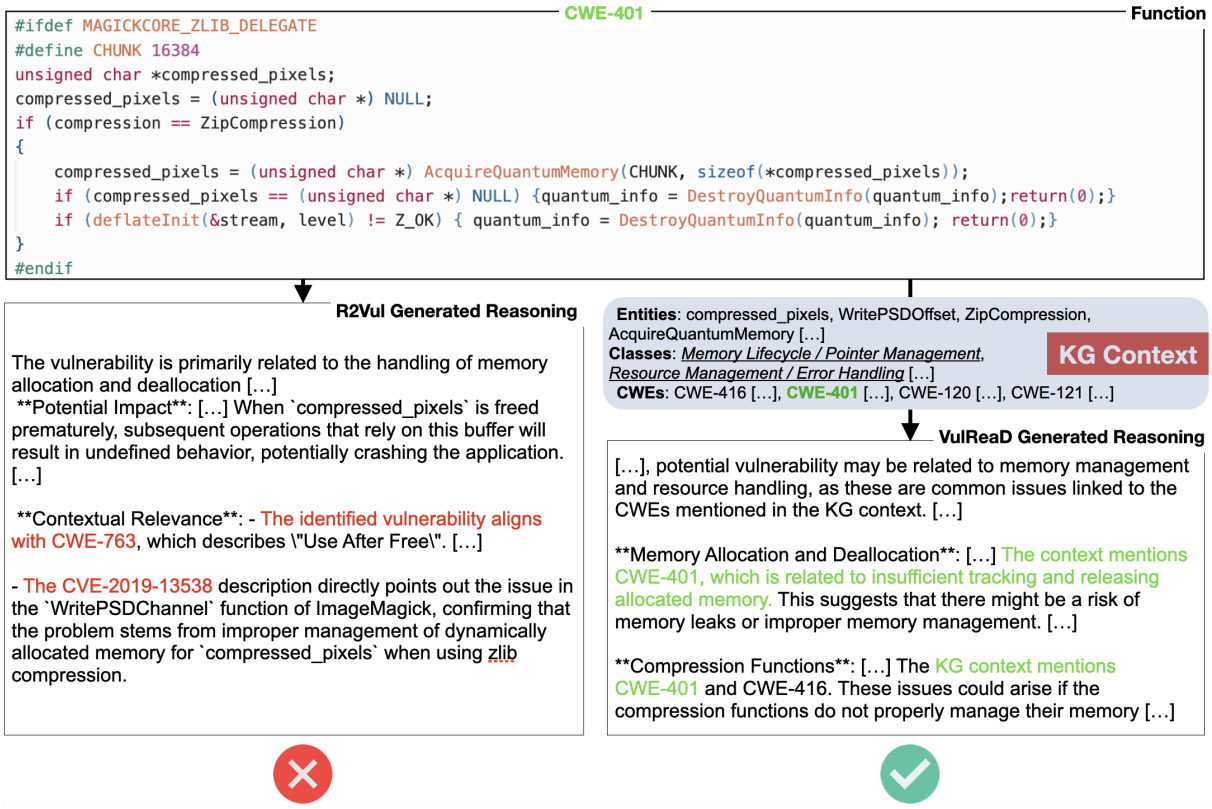

Figure 2

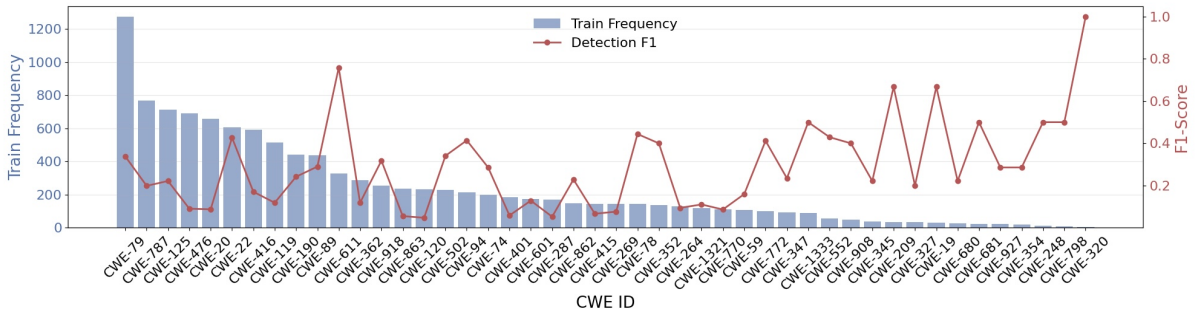

Figure 3: Training CWE Data Frequency vs. Reasoning-Based CWE Identification Accuracy.

more recently been explored for SVD, as their language modeling capabilities enable both code analysis and natural-language explanation generation (Wang et al., 2024a; Sun et al., 2024; Sultana et al., 2024). However, the correctness and reliability of such explanations remain an open challenge (Ullah et al., 2024; Haroon et al., 2025).

B.2 Vulnerability Taxonomies

The security community has developed standardized taxonomies to support consistent vulnerability reporting and analysis (Barnum, 2012; Fenz and Ekelhart, 2009; Syed et al., 2016). CVE[2] assigns unique identifiers to publicly disclosed vulnerabilities in specific products, while CWE[3] provides a complementary hierarchical taxonomy of abstract software weakness types–such as improper input validation or insecure file handling–that characterize recurring root causes at the design or implementation level. Both are integrated in the NVD[4], which enriches CVE entries with severity metrics and descriptive metadata (Høst et al., 2023). We focus on CWE as the primary semantic abstraction, as its categories provide a principled basis for reasoning about vulnerability causes and supporting

[2]https://cve.mitre.org/

[3]https://cwe.mitre.org/

[4]https://nvd.nist.gov

remediation.

B.3 Knowledge Graphs for Security

A KG represents entities and their relationships as a graph, where nodes denote entities and edges encode semantic relations, capturing hierarchical structure and rich attributes. In the security domain, KGs provide a natural mechanism for organizing vulnerability knowledge– including CWE relationships, vulnerability descriptions, affected components, and mitigations–enabling structured querying and reasoning anchored in standardized taxonomies (Shi et al., 2024; Mouiche and Saad, 2025; Wu et al., 2025b; Høst et al., 2023; Liu et al., 2025b; Dong et al., 2024). In this work, we use Neo4j[5] as our graph database, leveraging its scalable storage and Cypher query language for efficient KG construction, querying, and maintenance.

C Implementation Details

All LLM-based experiments employ fine-tuning using QLoRA within the PEFT framework. Models are quantized to 4-bit precision, with a LoRA rank of 16 and scaling factor of 32. Optimization is performed using AdamW with a learning rate of 2×10^{-5} and a batch size of 64. Only adapter parameters are updated during training, while the base model weights remain frozen. We use a maximum input length of 4096 tokens and apply greedy decoding during inference with sampling disabled. All experiments are conducted on NVIDIA A100 GPUs with 80GB memory and RTX-class GPUs with 40GB memory, using mixed-precision training with FP16 enabled. Model training and evaluation are implemented using HuggingFace's Trainer API.

C.1 Ablation: Extended Results

Table 6 reports teacher-vs-VULREAD comparisons on PrimeVul and DiverseVul. On PrimeVul, the teacher achieves an F1 of 0.37 with Micro-F1 of 0.07 and Macro-F1 of 0.11, while VULREAD reaches 0.68, 0.22, and 0.36 respectively—roughly doubling performance across all three metrics. Results on DiverseVul follow the same trend: the teacher scores 0.41 F1, 0.05 Micro-F1, and 0.10 Macro-F1, compared to VULREAD's 0.79, 0.23, and 0.34. The consistent and substantial gap across both datasets reinforces the finding on R2Vul: the training pipeline internalizes vulnerability reasoning patterns that inference-time retrieval alone cannot replicate.

D Discussion

D.1 DL baselines

DL Baselines in Section 6.2 exhibited poor performance in RQ2, particularly in multi-class CWE classification. In this section we discuss additional experiment that show improvements on BiLSTM baseline. Our training corpus contains over 200 CWE types in R2Vul corpus, which introduced a severe imbalance across classes. We experimented with several standard strategies to address this issue, including downsampling, hybrid sampling, and class-weight adjustments. None of these approaches yielded meaningful improvements, and the model consistently collapsed to predicting a single CWE with negligible precision. To further investigate, we restricted the classification task to a smaller subset of the data. Specifically, we selected the first 20 CWE types from the MITRE Top-25 list and trained the BiLSTM on this reduced set. Under this setting, the model achieved measurable improvements, reaching a Macro Precision of 0.0502, Macro Recall of 0.1154, and Macro F1-score of 0.0641, compared to near-zero values in the full R2Vul setting as showed in Table 7 (Section 6.2). While these scores remain modest, they demonstrate that limiting the classification space to frequent and well-represented CWEs can partially mitigate the shortcomings of DL baselines.

D.2 Impact of KG retrieval

The ablation study in RQ3 highlights that KG guidance substantially improves both binary detection accuracy and CWE-level reasoning. This distinction reflects the nature of the task: binary detection relies primarily on surface-level vulnerability cues, whereas CWE attribution requires structured semantic alignment. By retrieving knowledge from the KG, we enable reasoning that is more precise and less prone to confusion between related categories. These findings confirm that KG integration is most critical for fine-grained classification rather than coarse detection.

D.3 Impact of Knowledge Guidance

We examine the effect of different information sources on model reasoning. In the original R2Vul setup, reasoning traces often contained CVE identifiers. Our analysis showed that these identifiers

[5]https://neo4j.com/

Table 6: Teacher Model (Inference-Time KG) vs. VULREAD.

Model	DiverseVul			PrimeVul		
	F1	Macro-F1	Micro-F1	F1	Macro-F1	Micro-F1
Qwen-32B (Teacher + KG)	0.41	0.10	0.05	0.37	0.11	0.07
VULREAD	**0.79**	**0.34**	**0.23**	**0.68**	**0.36**	**0.22**

Table 7: BiLSTM Performance under Different CWE Subset Training Settings.

Training Setting	Macro P	Macro R	Macro F1
BiLSTM (Full CWE subset)	0.0136	0.0002	0.0004
BiLSTM (Top-25 CWE subset)	**0.0502**	**0.1154**	**0.0641**

were consistently hallucinated and did not match ground-truth labels. When included, they negatively affected performance of VULREAD. For this reason, we masked CVE-IDs during training to prevent the model from relying on hallucinating signals. We also evaluated the role of entities and abstract class information derived from the KG. These features did not improve binary detection when used in isolation. However, they contributed to better CWE-level reasoning when combined with KG guidance. This indicates that such information is useful only in contexts where structured relational knowledge is available to support reasoning.

Threats to validity, including data distillation quality, CWE classification noise, and computational constraints, are discussed in Appendix E.

D.4 Reasoning-oriented LLMs as future backbones

The student models in VulReaD are open-weight code LLMs fine-tuned via ORPO, consistent with established practice in supervised SVD. Reasoning-oriented models such as DeepSeek-R1 or QwQ are primarily designed for inference-time chain-of-thought generation and have not been adopted as fine-tuning backbones in prior SVD work. Notably, the ablation results in Table 3 (Section 6.3) indicate that the observed gains are attributable to the KG framework rather than model capacity, suggesting that VulReaD's contributions are largely backbone-agnostic. Investigating whether KG-guided preference optimization further amplifies the native reasoning capabilities of reasoning-oriented architectures in CWE-level diagnosis represents a promising direction for future work.

E Threats to Validity

Computational constraints The number of hyperparameter trials for QLoRA configuration and the LLM generation method was relatively limited. Additionally, due to GPU constraints, we did not attempt to fine-tune larger models, such as the 34B version. Although these limitations may have resulted in alternative findings, they do not fundamentally undermine our primary objective: to explore the potential of conducting vulnerability detection and explanation tasks using LLMs.

Data Distillation The teacher model (Qwen2.5-32B-Instruct) is applied only to training splits; test sets are withheld from distillation entirely. However, as a large pretrained model, Qwen2.5-32B-Instruct may have encountered publicly available vulnerability datasets or CVE descriptions during pretraining. This risk is inherent to all LLM-based SVD approaches and is not unique to VulReaD. To mitigate its influence on our conclusions, we evaluate all models under identical conditions on held-out test sets with no test-time supervision and conduct ablation study.