

ReVul-CoT: Towards Effective Software Vulnerability Assessment with Retrieval-Augmented Generation and Chain-of-Thought Prompting

Zhijie Chen[✉]^a, Xiang Chen[✉]^{a,*}, Ziming Li[✉]^a, Jiacheng Xue[✉]^a, Chaoyang Gao[✉]^a

^a*School of Artificial Intelligence and Computer Science, Nantong University, Nantong, China*

Abstract

Context: Software Vulnerability Assessment (SVA) plays a vital role in evaluating and ranking vulnerabilities in software systems to ensure their security and reliability.

Objective: Although Large Language Models (LLMs) have recently shown remarkable potential in SVA, they still face two major limitations. First, most LLMs are trained on general-purpose corpora and thus lack domain-specific knowledge essential for effective SVA. Second, they tend to rely on shallow pattern matching instead of deep contextual reasoning, making it challenging to fully comprehend complex code semantics and their security implications.

Method: To alleviate these limitations, we propose a novel framework ReVul-CoT that integrates Retrieval-Augmented Generation (RAG) with Chain-of-Thought (CoT) prompting. In ReVul-CoT, the RAG module dynamically retrieves contextually relevant information from a constructed local knowledge base that consolidates vulnerability data from authoritative sources (such as NVD and CWE), along with corresponding code snippets and descriptive information. Building on DeepSeek-V3.1, CoT prompting guides the LLM to perform step-by-step reasoning over exploitability, impact scope, and related factors.

Results: We evaluate ReVul-CoT on a dataset of 12,070 vulnerabilities. Experimental results show that ReVul-CoT outperforms state-of-the-art SVA baselines by 16.50%–42.26% in terms of MCC, and outperforms the best baseline by 10.43%, 15.86%, and 16.50% in Accuracy, F1-score, and MCC, respectively. Our ablation studies further validate the contributions of considering dynamic retrieval, knowledge integration, and CoT-based reasoning.

Conclusion: Our results demonstrate that combining RAG with CoT prompting significantly enhances LLM-based SVA and points out promising directions for future research.

Keywords: Software Vulnerability Assessment; Large Language Model; Retrieval Augmented Generation; Chain-of-Thought.

1. Introduction

Software vulnerabilities undermine the security of computer systems, making them susceptible to instability and external exploitation. Once exploited by adversaries, these weaknesses may lead to issues such as illicit system access, information leakage, economic damage, operational interruptions, and potential legal liabilities [1]. Such vulnerabilities often stem from programming faults, architectural weaknesses, or misconfigurations, and their potential risks are amplified by the increasing sophistication of modern cyberattacks. Therefore, timely detecting and addressing such issues is crucial to maintaining software security. Software Vulnerability Assessment (SVA) [2, 3] has been widely adopted as a systematic approach to identify, evaluate, and prioritize security flaws, enabling organizations to mitigate critical vulnerabilities in advance of potential exploitation.

In recent years, most previous SVA approaches rely on the analysis of vulnerable source code or textual vulnerability de-

scriptions. For instance, Liu et al. [4] leveraged contextual information from vulnerable functions to predict severity levels, while Shahid et al. [5] extracted semantic features from vulnerability descriptions to infer CVSS scores. Although these approaches improve assessment to some extent, their reliance on a single information source limits both accuracy and interpretability. More recently, Gao et al. [6] explored the use of in-context learning (ICL) to integrate both code and descriptions for severity assessment, demonstrating the potential of multi-source information fusion. Meanwhile, Large Language Models (LLMs) have demonstrated remarkable capabilities across different software engineering tasks [7, 8, 9, 10, 11], their strengths in large-scale training, contextual understanding, and pattern recognition make them promising candidates for advancing SVA. However, applying LLMs to this task still faces two key challenges. **First, insufficient domain knowledge coverage**, as LLMs alone cannot keep pace with the rapidly evolving vulnerability landscape and require the incorporation of external knowledge sources; **Second, limited contextual reasoning capability**, since severity assessment involves reasoning about exploitability, impact scope, and related factors, yet existing approaches often map inputs directly to outputs without a structured reasoning process, leading to inaccurate results.

*Corresponding author

Email addresses: chengzhi3333333@gmail.com (Zhijie Chen[✉]), xchencs@ntu.edu.cn (Xiang Chen[✉]), dzycd53@gmail.com (Ziming Li[✉]), xjc202603@gmail.com (Jiacheng Xue[✉]), gcyol@outlook.com (Chaoyang Gao[✉])

To alleviate the aforementioned limitations, we propose a novel framework ReVul-CoT that integrates Retrieval-Augmented Generation (RAG) [12, 13, 14] with Chain-of-Thought (CoT) prompting [15]. Different from previous approaches that depend on a fixed set of static exemplars, ReVul-CoT incorporates a dynamic retrieval mechanism to obtain contextually relevant information from a local knowledge base. This RAG knowledge base combines vulnerability information crawled from authoritative sources such as the National Vulnerability Database (NVD) [16] and the Common Weakness Enumeration (CWE) [17] with the original vulnerability code and descriptions, preserving technical details while enriching them with higher-level contextual knowledge related to vulnerability types, exploitability, and potential impacts. This design allows the model to analyze vulnerabilities by considering both code-level features and broader semantic information. In addition, we adopt CoT prompting to guide the LLM through a step-by-step reasoning process. For instance, the model first examines retrieved examples, then analyzes the target vulnerability in terms of its context and characteristics, and finally derives its severity level by reasoning through exploitability, impact scope, and related factors. This structured reasoning process makes the decision path more transparent and provides a traceable basis for severity assessments.

We evaluate the effectiveness of ReVul-CoT on a dataset containing 12,070 vulnerability records. By combining this dataset with vulnerability information crawled from NVD and CWE, we constructed a local knowledge base that provides richer input features for the model. For severity assessment, we adopt the CVSS v3 standard, as it provides a more detailed and adaptable scoring system than CVSS v2, and aligns better with modern security assessment practices. For the backbone LLM, we employ DeepSeek-V3.1 [18], which has demonstrated strong performance in handling complex programming language tasks. DeepSeek-V3.1 not only shows enhanced capability in parsing source code semantics but also provides substantial improvements in context window length and reasoning ability over Popular LLMs, such as the Qwen series. These characteristics enable it to effectively process long inputs that combine extensive vulnerability descriptions with retrieved knowledge, establishing a solid foundation for SVA.

Experimental results demonstrate that ReVul-CoT outperforms baseline approaches across multiple evaluation metrics. In particular, ReVul-CoT achieves 87.50% Accuracy, 83.75% F1-score, and 79.51% MCC, representing improvements of 10.43, 15.86, and 16.5 percentage points, respectively, over the best baseline. Our ablation studies further validate the contribution of our framework components, including the incorporation of the local knowledge base, the retrieval strategy configuration, and the application of CoT prompting. These findings indicate that the integration of dynamic retrieval with reasoning chains can enhance the accuracy and consistency of SVA. Finally, comparisons with other popular LLMs, such as Qwen3-Coder and GLM-4.5, show that the DeepSeek-V3.1-based implementation remains highly competitive.

Our findings suggest that combining RAG with CoT reasoning can substantially improve SVA performance and open

new directions for future research in SVA and related tasks.

To our best knowledge, the main contributions of our study can be summarized as follows:

- **Perspective.** We integrate RAG with CoT prompting for software vulnerability assessment and then introduce the novel framework ReVul-CoT.
- **Approach.** Our proposed ReVul-CoT dynamically retrieves relevant external knowledge from a local knowledge base, constructed with data crawled from authoritative sources such as NVD and CWE, and combines it with source code and vulnerability descriptions. CoT prompting is employed to guide the model through step-by-step reasoning on exploitability, impact scope, and other related factors. This design improves both consistency and interpretability in SVA.
- **Dataset and Knowledge Base.** We evaluate our proposed framework on a dataset containing 12,070 vulnerability records and construct a local knowledge base using vulnerability descriptions from authoritative sources such as NVD and CWE. The dataset provides detailed vulnerability descriptions and corresponding source code, while the knowledge base integrates external domain knowledge, including vulnerability types, severity metrics, and exploit references. For severity assessment, we adopt the CVSS v3 standard to ensure a standardized evaluation framework.
- **Practical Effectiveness.** Experimental results show that ReVul-CoT can outperform baselines in terms of different performance metrics. Ablation studies further confirm the effectiveness of key components, including knowledge retrieval and reasoning chains.

Open Science. To enable verification and subsequent research, we share our dataset, source code and detailed results on GitHub (<https://github.com/chengzhi333/ReVul-CoT>).

2. Research Background and Research Motivation

2.1. Software Vulnerability Assessment

Software vulnerability assessment plays a vital role in software security by evaluating and prioritizing vulnerabilities based on their potential impact and exploitability. Through the calculation of risk scores according to predefined scoring criteria [2, 19, 20, 21, 22], SVA determines vulnerability severity and supports prioritizing remediation efforts. However, despite its importance, accurately assessing vulnerabilities remains a significant challenge. As software vulnerabilities (SV) become more complex and widespread, they introduce substantial security risks to modern software systems [2]. Inaccurate assessments may result in the underestimation of high risk vulnerabilities or the over-prioritization of low risk ones, causing sub-optimal allocation of security resources. Due to the increasing volume and complexity of vulnerabilities, manual evaluation has become increasingly impractical. It demands considerable

expertise and time and is still prone to inconsistencies and errors, owing to the dynamic nature of security threats. Therefore, automated and data-driven approaches have become pivotal in improving SVA and prioritization [3].

The Common Vulnerability Scoring System (CVSS) [23] offers a unified and systematic method to assessing software vulnerability severity. It assigns a numerical score from 0 to 10, where higher values indicate more critical vulnerabilities, based on multiple dimensions such as impact and exploitability. Early studies in SVA predominantly relied on CVSS v2 [24, 25]; however, CVSS v3.0 [26] has become the predominant version due to its more detailed metrics, improved assessment of exploitability, and enhanced measures of impact. Furthermore, CVSS v3.0 translates the numerical score into four qualitative severity categories—Low (0.1–3.9), Medium (4.0–6.9), High (7.0–8.9), and Critical (9.0–10.0), providing a clearer framework for prioritizing vulnerabilities.

Specifically, CVSS evaluates vulnerabilities across three key dimensions:

1. **Exploitability:** This dimension assesses the characteristics of a vulnerability that could lead to a successful attack. It is evaluated using four key metrics:

- **Attack Vector (AV):** Reflects the method by which a vulnerability can be exploited, such as through a network, adjacent network, or local system access.
- **Attack Complexity (AC):** Describes conditions that are beyond the attacker's control, such as the need for specialized knowledge or additional conditions for a successful exploit.
- **Privileges Required (PR):** Evaluates the level of privileges or access rights the attacker must acquire to exploit the vulnerability.
- **User Interaction (UI):** Indicates whether the vulnerability requires user interaction to trigger the exploit, distinguishing between vulnerabilities that can be exploited by an attacker alone or those requiring user participation.

2. **Scope:** The Scope metric determines whether exploiting a vulnerability impacts resources beyond its immediate security context, such as affecting other components within a system or causing cascading breaches. It helps to understand whether an exploit could compromise other parts of the system beyond the vulnerable component.

3. **Impact:** The Impact dimension evaluates the potential consequences of successfully exploiting the vulnerability. It measures the effect on the **Confidentiality, Integrity, and Availability** of information:

- **Confidentiality (C):** Refers to preventing unauthorized access to sensitive information.
- **Integrity (I):** Assesses the trustworthiness and accuracy of the information, ensuring that data remains reliable.
- **Availability (A):** Represents the accessibility of critical resources (e.g., memory or processing power) that could be disrupted during an attack.

For SVA, obtaining comprehensive and reliable data sources is crucial for driving progress in both academic studies and practical implementations. The National Vulnerability Database [4, 16] provides valuable data on vulnerabilities, affected systems, and mitigation strategies, significantly enhancing the capabilities of SVA systems.

2.2. Retrieval-Augmented Generation

Retrieval-Augmented Generation [12, 13, 14] has emerged as an advanced framework that significantly enhances the performance of LLMs by integrating real-time, domain-specific external knowledge. This integration addresses a key limitation of LLMs, namely hallucinations, by enabling the retrieval and incorporation of up-to-date information, thereby improving the relevance and accuracy of generated outputs. The RAG framework operates in three steps [27]: (1) External data is first transformed into vector representations and stored in a vector database. (2) When a query is issued, the system retrieves the most relevant records through similarity matching. (3) The retrieved information is then integrated with the query and supplied as input to the LLM. This mechanism makes RAG particularly well-suited for a wide range of applications, including SVA.

In the rapidly evolving domain of software security, the ability of RAG to incorporate authoritative external knowledge, such as CWE and NVD, distinguishes it from traditional methods that primarily rely on source code or textual descriptions. By enriching the model's context by incorporating high-level, generalizable knowledge from trusted sources, RAG enables LLMs to predict vulnerability severity with greater accuracy and effectiveness. Prior research introduced the Vul-RAG framework [28], which leverages high-level knowledge to enhance vulnerability detection, extending beyond traditional code-level methods.

RAG addresses the challenge of outdated or incomplete information in LLMs, which is particularly crucial in software vulnerability management, where thousands of CVEs are published annually [29], RAG's capability to integrate the latest data ensures that decision-making is grounded in up-to-date information. Traditional LLMs struggle to include this rapidly growing security data in their training, often leading to inaccuracies. The RAG framework mitigates this by enabling models to access external information, ensuring more accurate assessments and providing a scalable solution to the evolving security landscape.

2.3. Chain-of-Thought Prompting

Chain-of-Thought [15] Prompting is a technique in prompt engineering that encourages the LLM to reason step by step by breaking down a problem into intermediate stages. This approach generates a natural language reasoning chain that guides the model toward the final answer. This approach has demonstrated substantial effectiveness in tasks involving arithmetic and symbolic reasoning [15].

Generally, Chain-of-Thought prompting can be categorized into two main forms: few-shot and zero-shot. The zero-shot

CoT has proven effective in certain tasks by directly presenting the reasoning steps in the query without requiring exemplars [30], thus enhancing task-solving capabilities. In contrast, the few-shot CoT setting provides the model with several exemplars containing complete reasoning processes [15, 31]. These examples are static and predefined, illustrating how the model should perform step-by-step reasoning. By learning from these demonstrated reasoning patterns, the model can generate coherent and logical reasoning chains for new inputs. In the context of SVA, CoT can be leveraged to enhance the model’s ability to predict the severity of software vulnerabilities by providing a coherent reasoning chain that mimics human thought processes. Figure 1 illustrates that Chain-of-Thought prompting helps the model accurately determine the severity of a target vulnerability, while standard single-step prompting leads to misclassification.

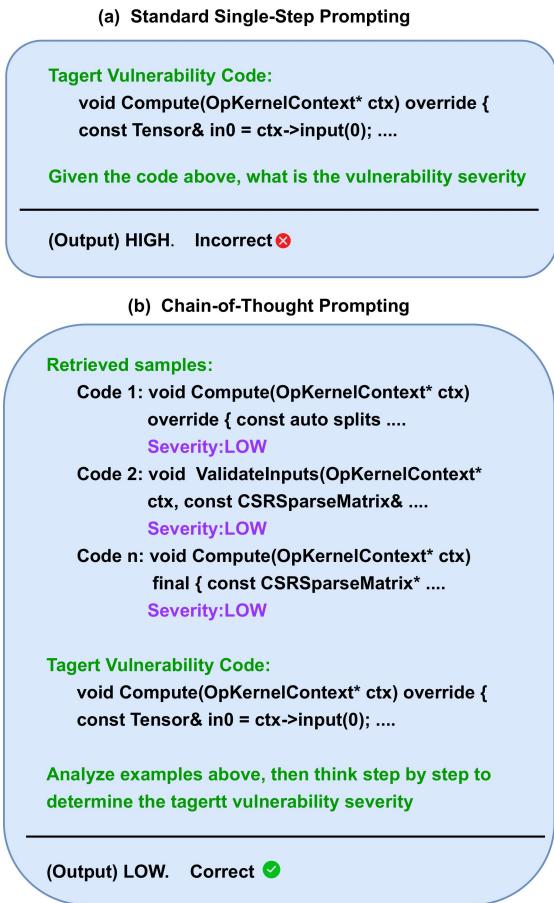


Figure 1: Comparison between standard single-step prompting and Chain-of-Thought prompting in software vulnerability severity assessment.

In SVA, CoT can enhance the model’s ability to predict vulnerability severity by providing a coherent reasoning chain that mimics human thought processes. This is particularly valuable in vulnerability severity assessment, where decision-making often requires a series of logical steps and domain knowledge. CoT enables the model to break down complex problems into more manageable steps. By including reasoning steps within the prompt, CoT allows the model to better understand the issue

at hand, considering multiple facets of the vulnerability, such as code context, severity indicators, and security implications. Prior studies [32] have demonstrated that this stepwise reasoning paradigm leads to improved reasoning accuracy across complex decision-making tasks.

2.4. Research Motivation

Motivation 1: Limitations of Existing SVA Methods. Software vulnerabilities pose severe risks to system security, and accurate severity assessment is critical for prioritizing mitigation efforts. However, existing SVA techniques still suffer from fundamental limitations. Traditional approaches [4, 24] mainly rely on a single information source, such as source code or textual vulnerability descriptions, which restricts their ability to capture the multifaceted characteristics of vulnerabilities. This unimodal perspective fails to represent the complex relationships among vulnerability causes, exploitability, and impact scope. Although recent studies [6, 33] attempt to combine code and textual information, they remain limited by shallow integration. These methods typically ignore domain knowledge from authoritative vulnerability sources (such as CWE taxonomies, NVD metrics, and relational dependencies) that contain essential semantic cues for accurate SVA.

Motivation 2: Challenges of LLM-based SVA. LLMs have demonstrated impressive capabilities in various software engineering tasks due to their strong contextual understanding and reasoning abilities. However, when applied to SVA, they face some limitations. The first is the lack of domain-specific expertise [34]. Pretrained LLMs are primarily trained on general-purpose corpora with static parameters, they lack an understanding of vulnerability assessment standards and decision rules—such as why a particular vulnerability is assigned a specific severity level. This deficiency in domain knowledge often leads to biases and inaccuracies when performing SVA. The second is limited reasoning depth. Existing LLM-based approaches typically treat SVA as a single-stage classification task, directly mapping inputs to outputs without engaging in step-by-step analytical reasoning. Consequently, these models are prone to hallucinations or inconsistent predictions, especially when dealing with complex vulnerability code and descriptions. Therefore, there is an urgent need for a knowledge-enhanced and reasoning-driven framework to address the domain knowledge gap and improve reasoning consistency, enabling more accurate and interpretable SVA.

Motivation 3: Bridging Knowledge and Reasoning through RAG and CoT. To overcome the above limitations, an effective way is to integrate knowledge enhancement and structured reasoning by combining Retrieval-Augmented Generation and Chain-of-Thought. Specifically, RAG retrieves domain-relevant information from authoritative vulnerability sources to construct and enrich a knowledge base, thereby enhancing the model’s contextual understanding and factual grounding. Meanwhile, CoT prompting guides the model to perform structured and step-by-step reasoning when analyzing vulnerability exploitability and severity, improving the logical consistency and stability of the reasoning process. By integrating these two complementary mechanisms, knowledge retrieval and reasoning guidance,

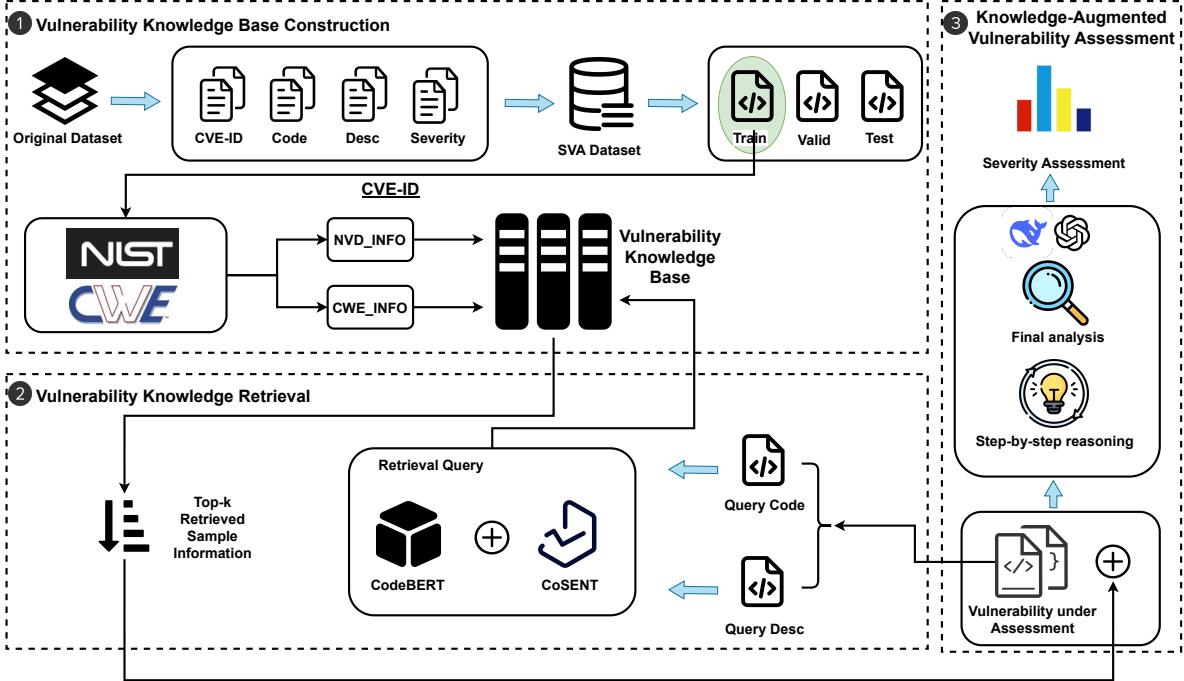


Figure 2: Framework of our proposed approach ReVul-CoT

the SVA framework can effectively bridge the gap between domain knowledge coverage and reasoning depth.

3. Our Proposed Framework ReVul-CoT

3.1. Overview

Figure 2 illustrates the framework of our proposed approach ReVul-CoT. The framework consists of three main phases. In the **vulnerability knowledge base construction phase**, ReVul-CoT collects and organizes vulnerability information from multiple authoritative sources to build a comprehensive knowledge base. In the **knowledge embedding and retrieval phase**, both the source code and textual descriptions of the target vulnerability are encoded into a shared embedding space. Similarity computation is then performed to retrieve the top- k semantically related vulnerability samples from the knowledge base. Finally, in the **knowledge-augmented severity assessment with CoT prompting phase**, the retrieved knowledge, along with the target vulnerability, is incorporated into prompting templates. The LLM then leverages this information to perform step-by-step reasoning and generate the final prediction of the corresponding vulnerability severity level. In the following subsections, we provide a detailed description of each phase.

3.2. Phase 1: Vulnerability Knowledge Base Construction

This phase aims to construct a comprehensive and structured knowledge base to support retrieval-augmented SVA. The primary objective is to enrich the dataset with authoritative domain knowledge and organize it into a format compatible with downstream retrieval and reasoning processes. By integrating

external vulnerability information and standardizing heterogeneous data sources, this phase establishes a unified knowledge foundation that enhances both contextual understanding and interpretability.

We select a subset of samples from the original dataset as knowledge base vulnerabilities, ensuring both representativeness and diversity. Each vulnerability sample includes its CVE-ID, vulnerable code snippet, detailed textual description, and the associated severity label. To augment the descriptive and semantic quality of each CVE instance, we collect vulnerability information from the NVD by querying authoritative databases and extract standardized CVSS metrics, which quantify the potential impact and exploitability of each vulnerability. While the dataset itself contains only basic code, description, and severity information, the integration of CVSS-related considerations during the knowledge processing stage ensures that each instance is encoded with relevant technical context. In addition, we collect data from the official CWE website, extract its definitions and descriptions, and integrate them into the dataset. By linking each CVE instance to the conceptual knowledge embedded in CWE, we improve the interpretability of the vulnerabilities while maintaining the original structure of the data. As a result, each CVE entry evolves into a structured unit of knowledge, combining concrete code, textual descriptions, and conceptual insights, forming a cohesive representation for later retrieval. Table 1 presents the original fields of our CVE dataset and the augmented external knowledge integrated via the local knowledge base.

Then, we normalize all the collected information, including source code, CVE descriptions, CVSS metrics, and CWE-derived knowledge, is normalized into structured JSON objects. This standardized representation ensures that each CVE instance

Table 1

Comparison of original dataset and augmented external knowledge.

Original Dataset	Augmented External Knowledge
CVE-ID	NVD_INFO: CVSS Metrics (v3.x), Impact Scores, Exploitability Scores, Affected CPEs (software/hardware configurations)
Code	
Description	CWE_INFO: CWE-ID, Descriptions of Weaknesses, Extended Descriptions, Common Consequences
Base Severity	

contains both concrete technical details (e.g., vulnerable code and CVSS scores) and abstract semantic information (e.g., CWE definitions and mitigation strategies). By associating each CVE instance with its enriched contextual knowledge, it can capture both specific implementation details (such as code behavior) and high-level conceptual insights (such as attack characteristics and mitigation strategies). The constructed knowledge base preserves the essential elements of each CVE instance while incorporating authoritative external information from NVD and CWE. This unified design maintains data integrity and provides a rich foundation for retrieval-augmented reasoning prediction. During inference, the interaction between raw data and enriched contextual knowledge enables the LLM not only to understand the underlying technical details of vulnerabilities but also to grasp their higher-level semantic relationships and security implications.

Through this structured integration of technical and semantic information, each CVE instance becomes part of a contextually rich knowledge framework. This multi-layered representation enhances the model’s ability to reason across different levels of abstraction and supports retrieval-augmented reasoning, ultimately improving the effectiveness of SVA.

3.3. Phase 2: Knowledge Embedding and Retrieval

In this phase, the embedding and retrieval process serves as a core component of the proposed framework, enabling efficient similarity computation and precise retrieval of semantically related vulnerability knowledge for severity assessment. This process consists of two parts: (1) **The Embedding Part:** The main objective of this part is to map vulnerability code and textual descriptions into a shared embedding space, allowing the model to represent and compare them within a unified semantic dimension. By embedding these two modalities into the same latent space, the model can capture both the structural characteristics of the code and the semantic meaning of the textual descriptions, thus providing a consistent and reliable foundation for similarity-based retrieval. (2) **The Retrieval Part:** Based on the obtained embeddings, this part calculates cosine similarities for two modalities—code-to-code and description-to-description—to generate two relevance scores for each entry

in the knowledge base. These two scores are then combined through a tunable weighting parameter to obtain an overall similarity, thereby achieving a dynamic balance between structural and semantic information. According to the aggregated similarity, all entries are ranked, and a specified number of the most relevant candidate samples are selected as retrieval results. Finally, these candidate samples are fed into a backbone LLM for further analysis and vulnerability severity assessment.

3.3.1. Embedding Part

In the Embedding Part, we focus on projecting both the vulnerability code and its corresponding description into a shared, continuous vector space. This enables meaningful similarity computations during the retrieval phase, which can efficiently match vulnerabilities based on both their structural and semantic features. To achieve this, we utilize CodeBERT [35] for encoding vulnerability code and CoSENT [36] for encoding vulnerability descriptions. These models were specifically selected due to their remarkable performance in capturing both the syntactic and semantic nuances of programming code and natural language text.

For code embedding, we utilize CodeBERT, a pre-trained bimodal model that is adept at handling various programming languages. CodeBERT [37, 38] processes tokenized vulnerability code and generates a fixed-dimensional vector $\mathbf{X}_A \in \mathbb{R}^D$, where D represents the dimensionality of the embedding space. The semantic representation of the code is derived by passing the code through CodeBERT, extracting the output from the final hidden layer, and averaging the hidden states to create a unified vector representation.

For the vulnerability descriptions, we leverage CoSENT [36], a model implemented based on Text2vec [39], which is specifically designed for processing textual data. CoSENT generates an embedding for each vulnerability description, capturing the underlying semantic meaning, and maps it into the shared vector space with the code embeddings. This allows for effective retrieval, as both the code and description are represented in the same vector space, enabling more coherent similarity computations.

3.3.2. Retrieval Part

The retrieval Part performs a similarity-based search within the vulnerability knowledge base, utilizing the embeddings generated during the embedding phase. Specifically, it calculates cosine similarity between the embedding of the target vulnerability’s source code and those of other code snippets in the knowledge base, as well as between the target vulnerability’s textual description and corresponding descriptions in the repository. On this basis, we then select the most relevant candidate vulnerabilities based on each similarity measure. Next, we apply a combined method to aggregate the code and description similarity scores into a weighted overall similarity, which reflects the total relevance of each candidate. This dual retrieval mechanism ensures that both the structural features of the vulnerability and its semantic content are fully considered, leading to a more comprehensive identification of the most relevant

samples and enhancing the accuracy and effectiveness of the retrieval process.

Code Similarity. To measure the similarity between the target vulnerability’s code and the retrieved code snippets, we employ cosine similarity. The formula for computing cosine similarity between the target code \mathbf{X}_A and a candidate code \mathbf{X}_B is as follows:

$$\text{CosSim}_{\text{code}}(\mathbf{X}_A, \mathbf{X}_B) = \frac{\mathbf{X}_A \cdot \mathbf{X}_B}{\|\mathbf{X}_A\| \|\mathbf{X}_B\|} \quad (1)$$

where \mathbf{X}_A and \mathbf{X}_B represent the vector embeddings of the target and retrieved vulnerability codes, respectively. This formula calculates the cosine of the angle between the two vectors, quantifying their semantic similarity in the embedded space.

Description Similarity. We compute the cosine similarity between the target vulnerability’s description and the descriptions in the knowledge base. The similarity score is computed as:

$$\text{CosSim}_{\text{desc}}(\mathbf{D}_A, \mathbf{D}_B) = \frac{\mathbf{D}_A \cdot \mathbf{D}_B}{\|\mathbf{D}_A\| \|\mathbf{D}_B\|} \quad (2)$$

where \mathbf{D}_A and \mathbf{D}_B represent the embeddings of the target and retrieved vulnerability descriptions, respectively. This computation assesses the semantic similarity between the textual descriptions.

Combined Similarity. To combine both code and description similarities, we compute the overall similarity between two vulnerabilities V_A and V_B using a weighted sum of the individual similarity scores:

$$\begin{aligned} \text{Sim}(V_A, V_B) &= \phi \times \text{CosSim}_{\text{code}}(\mathbf{X}_A, \mathbf{X}_B) \\ &\quad + (1 - \phi) \times \text{CosSim}_{\text{desc}}(\mathbf{D}_A, \mathbf{D}_B) \end{aligned} \quad (3)$$

in this equation, ϕ is a parameter that controls the relative weighting between the code and description similarities. This allows for flexible tuning of the retrieval process based on the context of the task at hand. By adjusting ϕ , we can prioritize either the structural aspects of the code or the semantic content of the description, depending on the requirements of the vulnerability evaluation. This combined similarity measure provides a comprehensive and flexible mechanism for retrieving the most relevant vulnerabilities, enhancing the overall retrieval accuracy.

After calculating the similarity scores, the top- k most relevant samples are retrieved from the knowledge base based on their high combined similarity to the target vulnerability. These top- k samples are selected for further processing, where a LLM is employed to make more informed predictions regarding the vulnerability’s severity.

3.4. Phase 3: Knowledge-Augmented Severity Assessment with CoT Prompting

After retrieving high-quality samples through the Embedding and Retrieval Mechanism, this phase leverages the retrieved

information for SVA. Specifically, we employ a large language model assessment framework that combines Retrieval-Augmented Generation with few-shot Chain-of-Thought prompting, where the retrieved samples are dynamically integrated into the CoT prompting process. By enriching the reasoning process with contextually relevant examples, the model is able to ground its analysis in concrete instances, thereby enhancing both the accuracy and interpretability of the predictions. Our method incorporates task-specific, dynamically retrieved samples from the knowledge base, ensuring that each prediction is supported by the most relevant and up-to-date contextual information.

Few-shot learning has demonstrated strong potential in enabling LLMs to perform domain-specific tasks with limited labeled data. In traditional few-shot CoT prompting [15, 31], the model is supplied with a predetermined set of exemplars and expected to reason through the target problem by imitating these demonstrations. While this strategy allows the LLM to decompose problems step by step, its reliance on static exemplars severely constrains adaptability in high-dimensional domains such as SVA. In these scenarios, vulnerabilities exhibit substantial diversity in code structure, semantic context, and security impact, rendering fixed exemplars insufficient for robust generalization. To address this limitation, our approach introduces a dynamic few-shot CoT mechanism supported by retrieval augmentation. The knowledge base, enriched with structured vulnerability information, serves as the source for selecting exemplars most relevant to the target case. At inference time, the top- k samples are retrieved based on similarity across both code embeddings and textual descriptions. These samples provide not only syntactic and semantic proximity but also contextual signals such as CWE categorizations and CVSS metrics. Once retrieved, the samples are embedded directly into the prompt, where they act as domain-specific demonstrations guiding the LLM’s reasoning. By dynamically updating exemplars for each prediction, this mechanism enables the model to adapt to evolving vulnerability patterns and ensures that the reasoning chain remains grounded in realistic, high-quality cases.

The core of our RAG-enhanced LLM prediction approach lies in the design of the prompt template. Drawing on insights from prior research in source code summarization and vulnerability assessment [6, 28], we have developed a CoT prompt template, the overall structure of which is shown in Figure 3, that effectively guides the LLM through the reasoning process by integrating both code semantics and vulnerability descriptions. The template is structured into three primary steps:

Step 1: Analyze Demonstration Samples. The LLM is provided with the top- k retrieved samples, which supply rich contextual information including vulnerability code, textual descriptions, and associated severity levels. The model examines these exemplars according to a set of criteria that encompass functional semantics, vulnerability causes, and exploitability, together with extended dimensions such as potential attack vectors, user interaction requirements, privilege escalation conditions, and the broader security environment in which the vulnerability may arise. The analysis further considers the likelihood of cascading impacts across components, consistency with CWE categorizations, and the quantitative severity met-

CoT Prompt:

You are an expert in code vulnerability assessment, and you will rate the vulnerabilities based on the following scoring criteria:
0.1-3.9: LOW, 4.0-6.9: MEDIUM, 7.0-8.9: HIGH, 9.0-10.0: CRITICAL.

Step 1: Carefully analyze the following retrieved vulnerability samples. For each sample, perform a structured analysis covering the following dimensions:

- Functional semantics of the code
- Root causes of the vulnerability
- Fixing or mitigation strategies
- Impact scope
- Exploitability factors
- Impact type
- Security context
- Severity mapping clues
- Official severity level

Top-k Samples Knowledge includes:
CVE-ID, CWE-IDs, Base Score, Base Severity, Code Snippet, Description, NVD Knowledge, CWE Knowledge

Step 2: Using the reasoning patterns and knowledge distilled from Step 1, analyze the target vulnerability in detail. Construct a structured explanatory representation that addresses the following aspects before making a judgment:

1. Functional Semantics
2. Vulnerability Causes
3. Fixing Solutions
4. Impact Scope
5. Exploitability
6. Impact Type
7. Security Context
8. Severity Mapping Clues

Step 3: Based on your reasoning in Step 1 and Step 2, output the severity level of the target vulnerability.

Figure 3: The CoT prompt template utilized by ReVul-CoT.

rics defined by CVSS.

Step 2: Target Vulnerability Analysis. Building on the knowledge distilled from Step 1, the target vulnerability, consisting of its source code and textual description, is presented to the LLM for detailed examination. Guided by the reasoning patterns extracted from the exemplars, the model performs a structured analysis of the target instance across multiple dimensions. It first evaluates the impact scope by contrasting the behavioral characteristics of the vulnerability with those observed in the retrieved samples, distinguishing whether the issue remains confined to a local component or has the potential to propagate across module boundaries. Exploitability is then assessed by aligning the target conditions with attack vectors, privilege requirements, and user interaction patterns identified in Step 1. Finally, the vulnerability is interpreted within a broader security framework, integrating CWE categorizations, CVSS metrics, and the cascading effects highlighted in prior cases, thereby constructing a coherent representation.

Step 3: Severity Prediction. In the final stage, the LLM synthesizes the structured knowledge generated in Step 2 and produces a severity assessment for the target vulnerability. To accomplish this, the API interface of the selected LLM is in-

voked, and the concrete prompt constructed according to the designed template and contextual information is provided as input. The model then processes this input and generates the prediction, which is not a direct classification but the result of a reasoning process that integrates multiple dimensions, including the impact scope, exploitability factors, and the broader security context established in the previous steps. The model evaluates how the identified characteristics align with standardized assessment frameworks and translates them into a severity level. Following the scoring criteria, the predicted severity is categorized into one of four levels: LOW, MEDIUM, HIGH, or CRITICAL. Each predicted level reflects a gradation of potential risk, where LOW denotes minimal security implications, MEDIUM indicates moderate but non-critical threats, HIGH corresponds to vulnerabilities with substantial risks requiring timely remediation, and CRITICAL highlights issues with severe security consequences demanding immediate mitigation. This reasoning-driven prediction ensures that the final severity level is both technically grounded and contextually justified.

4. Experimental Setup

In this section, we first elaborate on the research questions and the underlying design motivations. Subsequently, we provide a comprehensive overview of the experimental subjects, baselines, performance evaluation metrics, and implementation details.

4.1. Research Questions

To demonstrate the effectiveness and competitiveness of ReVul-CoT, we design the following five research questions (RQs) in this study.

RQ1: How does our proposed method ReVul-CoT perform in software vulnerability assessment compared to state-of-the-art baselines?

Motivation. RQ1 aims to investigate whether ReVul-CoT can surpass the performance of current state-of-the-art SVA baselines. To ensure a fair and comprehensive evaluation, we compare it against three categories of representative baselines, including description-based, code-based, and bimodal approaches. In particular, representative baselines include Func_{RF} and Func_{LGBM} [4] for source code-based assessment, CWM_{NB} , CWM_{SVM} , and CWM_{LR} [24] for description-based models, and advanced models such as Cvss-bert [5], SPSV [40], SVA-CL [41], MTLM [33], and SVA-ICL [6]. To objectively assess ReVul-CoT in comparison with the baseline methods, we adopt three commonly used evaluation metrics: Accuracy, Macro-F1, and Matthews Correlation Coefficient (MCC).

RQ2: Whether considering both source code and vulnerability description with information can improve the performance of ReVul-CoT?

Motivation. Two modalities contain complementary information (source code reflects syntactic and structural patterns, while vulnerability descriptions convey semantic and contextual knowledge). In this study, we hypothesize that integrating these two modalities can enhance the retrieval relevance and

reasoning quality of ReVul-CoT. Therefore, RQ2 investigates whether incorporating both code and text leads to better model performance, and further explore the most effective modality fusion ratio for ReVul-CoT in this RQ.

RQ3: How does the number of similar samples in RAG retrieval affect the performance of ReVul-CoT?

Motivation. In retrieval-augmented frameworks, the number of retrieved samples (Top- k) directly influences the contextual richness and relevance of the retrieved knowledge. Too few samples may result in limited context, while too many may introduce irrelevant or noisy information that harms reasoning quality. Moreover, LLMs such as DeepSeek-V3.1 have inherent token-length constraints, which necessitate a balance between retrieval quantity and reasoning efficiency. Hence, we design RQ3 to analyze how varying the number of retrieved samples affects the performance of ReVul-CoT.

RQ4: How does the use of Chain-of-Thought prompting affect the performance of ReVul-CoT?

Motivation. While retrieval provides ReVul-CoT with relevant external knowledge, it does not inherently enable structured reasoning or interpretability. Chain-of-Thought prompting helps LLMs perform step-by-step logical inference, improving their reasoning transparency and consistency. Therefore, we design RQ4 to investigate whether incorporating CoT reasoning improves ReVul-CoT’s ability to assess vulnerabilities more accurately.

RQ5: How does external knowledge enhancement impact the performance of ReVul-CoT?

Motivation. Although LLMs possess high generalization capabilities, their internal knowledge may be incomplete or outdated. To address this limitation, ReVul-CoT integrates structured domain knowledge from authoritative sources into its retrieval process. This external knowledge enrichment enables the model to better understand vulnerability. We design RQ5 to examine how such knowledge enhancement influences the overall reasoning and classification performance of ReVul-CoT.

4.2. Experimental Subject

We constructed our experimental subject based on the MegaVul dataset released by Ni et al. [42]. The latest version of MegaVul (updated on April 14, 2024) is derived from the Common Vulnerabilities and Exposures (CVE) repository [29] and contains 17,975 labeled vulnerability samples written in C and C++. In total, it covers 176 distinct vulnerability types across 1,062 open-source software projects reported between 2006 and 2024. Compared with earlier datasets such as SARD [43], Devign [44], and Big-Vul [45], MegaVul provides broader project coverage, richer vulnerability diversity, and higher data integrity through advanced code validation tools.

However, we found that the severity ratings in the MegaVul dataset were not fully standardized according to the CVSS v3 framework. To address this, we re-collected the CVSS v3 severity scores, vector information, and vulnerability descriptions for all entries, and removed records that lacked complete CVSS v3 information. After this processing, we ultimately obtained 12,070 vulnerability entries that fully comply with the CVSS

v3 standard. Despite the enhancements and novel concepts introduced in CVSS v4, practical datasets annotated with CVSS v4 scores are still extremely limited [6]. Therefore, this study primarily conducts analysis based on CVSS v3. For the 12,070 valid vulnerability entries, we retained only the CVE-ID, vulnerability description, source code snippet, and severity level, so that the dataset can be more effectively integrated with our knowledge base in subsequent experiments.

Consistent with prior work [6, 24], we applied stratified sampling to partition the dataset into a knowledge set (80%), a validation set (10%), and a test set (10%), maintaining the same distribution of the four severity levels across all subsets. To prevent information leakage, the retrieval knowledge base for the RAG module was constructed based solely on the knowledge set (80%). During validation and testing, the model retrieves information only from this fixed knowledge base and does not access any samples from the validation or test sets, thereby ensuring strict isolation between data partitions.

4.3. Performance Measures

To thoroughly assess the effectiveness of the proposed framework ReVul-CoT, we utilize three commonly used evaluation metrics: Accuracy, F1-score, and MCC. These metrics offer a comprehensive view of model’s performance, enabling a deeper assessment of its effectiveness. Below are the detailed definitions and formulas for calculating these performance metrics.

Accuracy: Accuracy is the proportion of correctly predicted samples over the total number of samples. In the context of SVA, it measures the overall correctness of the model’s classification across all severity categories. The formula for Accuracy is given by:

$$\text{Accuracy} = \frac{\sum_i TPI}{\sum_i (TPI + FPI + FNi + TNi)} \quad (4)$$

where TPI , FPI , FNi , and TNi represent the true positives, false positives, false negatives, and true negatives for the i -th severity class, respectively. Given the multi-class nature of the SVA task and the presence of class imbalance, we assess the performance of the model ReVul-CoT using the macro-averaged F1 score and MCC [46].

F1-score: The F1-score is the harmonic mean of Precision and Recall. It balances the trade-off between the ability of the model to correctly identify positive instances (precision) and its ability to identify all relevant instances (recall). For multi-class classification tasks like ours, the macro-average F1-score [47] is used, which is the average of the F1-scores across all categories. The F1-score and its macro-average are defined as follows:

$$F1\text{-score}_i = \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \quad (5)$$

$$F1\text{-score}_{\text{macro}} = \frac{1}{N} \sum_{i=1}^N F1\text{-score}_i \quad (6)$$

where $\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}$ and $\text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$. Here, N is the number of severity categories.

MCC: As a comprehensive metric incorporating true positives, false positives, false negatives, and true negatives, MCC is particularly effective for imbalanced datasets. While traditionally used for binary classification, it can be generalized to multi-class problems. The MCC for each severity class and its macro-average [48] are defined as:

$$\text{MCC}_i = \frac{TP_i \times TN_i - FP_i \times FN_i}{\sqrt{(TP_i + FP_i)(TP_i + FN_i)(TN_i + FP_i)(TN_i + FN_i)}} \quad (7)$$

$$\text{MCC}_{\text{macro}} = \frac{1}{N} \sum_{i=1}^N \text{MCC}_i \quad (8)$$

A higher MCC value indicates better performance, with a range from -1 (worst) to 1 (best).

4.4. Baselines

To evaluate the effectiveness of our proposed method, ReVulCoT, we compared it with seven state-of-the-art baselines from the field of SVA. These baselines were chosen based on their relevance to SVA and the well-established methodologies they utilize. We divided the baselines into three categories: (1) source code-based baselines, (2) description-based baselines, and (3) Bimodal data-based baselines.

Source Code-Based Baselines: Le et al. [4] proposed two source code-based SVA approaches, FunRF and FunLGBM. These methods incorporate contextual information from vulnerable code during model construction, thereby effectively enhancing the performance of vulnerability assessment.

- **Func_{RF}:** This baseline adopts the Random Forest (RF) [49] algorithm as its classifier. RF is an ensemble learning technique that generates multiple decision trees and integrates their outputs to improve predictive stability and accuracy. It demonstrates strong performance in classification tasks due to its ability to manage large-scale datasets and capture intricate feature interactions. The main hyperparameters include the number of estimators, the maximum depth, and the number of leaf nodes. In this baseline, the tuning process considers the following candidate values: the number of estimators is varied among 100, 200, 300, 400, 500; the maximum depth is examined at levels 3, 5, 7, 9, unlimited; and the number of leaf nodes is set to 100, 200, 300, unlimited.

- **Func_{LGBM}:** This baseline adopts the Light Gradient Boosting Machine (LGBM) algorithm as the classifier. LGBM [50] is a gradient boosting decision tree method designed for high efficiency and scalability, making it particularly suitable for handling large-scale datasets. It achieves competitive performance by constructing trees in a leaf-wise manner, which improves accuracy while maintaining computational speed. The primary hyperparameters, similar

to those in RF, include the number of estimators, the maximum depth, and the number of leaves.

Description-Based Baselines: Le et al. [24] proposed three approaches based on vulnerability descriptions, each employing different text processing techniques and classifiers to predict the severity of software vulnerabilities from textual data. These approaches were included in our comparison to evaluate their effectiveness against the proposed approach.

- **CWM_{NB}:** This baseline adopts the Naive Bayes (NB) [51] classifier, a probabilistic model grounded in Bayes' theorem and built upon the assumption of conditional independence among features. Owing to its simplicity and computational efficiency, NB performs effectively on large-scale datasets and offers rapid training and implementation. However, its predictive performance may decline when the independence assumption is violated. In this baseline, no hyperparameter tuning was conducted during the validation process.
- **CWM_{SVM}:** This baseline adopts the Support Vector Machine (SVM) [52] classifier, a supervised learning model that separates data points by constructing an optimal hyperplane within a transformed feature space. SVM is particularly effective in handling high-dimensional data and performs well in both binary and multi-class classification settings. To regulate model complexity, the regularization parameter is tuned using candidate values from 0.01, 0.1, 1, 10, 100.
- **CWM_{LR}:** This baseline adopts the Logistic Regression (LR) [53] classifier, a statistical learning model primarily designed for binary classification tasks. It can be extended to multi-class scenarios through the one-vs-rest strategy. LR is valued for its efficiency and interpretability, though its performance may depend on the chosen text representation (e.g., TF or TF-IDF). The regularization parameter is tuned from the candidate set 0.01, 0.1, 1, 10, 100, with 0.1 selected for TF features and 10 for TF-IDF features.

In addition to traditional machine learning baselines, we also included deep learning-based approaches for comparison. Specifically, two BERT-based methods, both leveraging pre-trained language models to capture semantic information from vulnerability descriptions.

- **Cvss-bert:** This baseline adopts a BERT-based framework proposed by Shahid et al. [5], in which individual classifiers are trained for each CVSS vector component to infer both the CVSS vector and the corresponding severity level from textual vulnerability descriptions. To enhance interpretability, the approach applies a gradient-based saliency analysis, allowing clearer insights into the model's decision process.
- **SPSV:** This baseline adopts the SPSV model proposed by Babalau et al. [40], which employs a multi-task learning

framework built upon a pre-trained BERT architecture to generate contextual representations from vulnerability descriptions. The model simultaneously predicts severity scores and related vulnerability metrics using only textual input available at the time of disclosure.

Bimodal Data-Based Baselines: Some recent approaches combine both source code and vulnerability descriptions to enhance prediction accuracy:

- **SVA-CL:** This baseline adopts the SVA-CL framework proposed by Xue et al. [41], which integrates prompt tuning with continual learning for vulnerability severity assessment. The method constructs hybrid prompts by combining information from both source code and vulnerability descriptions to fine-tune the CodeT5 model. To enhance model stability and performance, it incorporates confidence-based replay together with regularization strategies.
- **MTLM:** This baseline adopts the MTLM framework proposed by Du et al. [33], which integrates multimodal information from both source code and vulnerability descriptions. The model employs GraphCodeBERT to extract semantic and structural representations, followed by a Bi-GRU network with an attention mechanism for deeper feature refinement. A multi-task learning strategy with shared parameters is then applied to enhance the model’s generalization across related prediction tasks.
- **SVA-ICL:** This baseline adopts the SVA-ICL approach proposed by Gao et al. [6], which integrates vulnerability source code with textual descriptions for severity prediction. The method utilizes pre-trained language and code models to compute similarity scores across both modalities and retrieves the most relevant samples. These retrieved examples are incorporated as in-context demonstrations to guide the large language model in making severity predictions.

4.5. Implementation Details and Running Platform

In our SVA framework, vulnerability-related knowledge is retrieved by evaluating the similarity between source code and textual vulnerability descriptions, with a practical weighting of 60% for code and 40% for descriptions. This choice is informed by prior studies [6], which have shown that such a balance enhances the retrieval of relevant vulnerability information by leveraging both structural characteristics from source code and semantic cues from textual descriptions.

After retrieving similar samples using RAG, we apply few-shot CoT prompting for vulnerability severity assessment, the CoT samples are drawn from the knowledge retrieved by RAG. In terms of the hyperparameters for LLM, we adhere to the settings used in previous studies [54, 55, 56], specifically setting the temperature parameter to 0. This setting ensures that the model generates the most probable predictions, resulting in higher certainty in the generated text.

All experiments were conducted on a server equipped with an Intel(R) Core(TM) i7-13600K processor and a GeForce RTX 4090 GPU with 24GB of graphics memory. The operating system used was Windows 10. This configuration provides the computational power necessary to efficiently process large-scale vulnerability datasets, while also ensuring the resources required for effective handling and analysis of these datasets.

5. Results

5.1. RQ1: How does our proposed method ReVul-CoT perform in software vulnerability assessment compared to state-of-the-art baselines?

Approach. To validate the effectiveness of our proposed framework ReVul-CoT for the software vulnerability assessment task, we compare it against state-of-the-art baselines, including Func_{RF} , Func_{LGBM} [4], CWM_{NB} , CWM_{SVM} , CWM_{LR} [24], Cvss-bert [5], SPSV [40], SVA-CL [41], MTLM [33] and SVA-ICL [6]. The experimental settings for ReVul-CoT follow those described in Section 4.5, and the performance of all approaches is evaluated using the metrics introduced in Section 4.3.

Result. Table 2 presents the comparative results between our proposed ReVul-CoT and the baseline methods. For each metric, the best and second-best results are highlighted in bold and underline, respectively. As can be seen, ReVul-CoT consistently outperforms the competing approaches on Accuracy, F1-score, and MCC, with the most remarkable advantage observed on MCC. In particular, ReVul-CoT achieves 87.50% Accuracy, 83.75% F1-score, and 79.51% MCC. When compared with the strongest baseline models, these results indicate improvements of at least 10.43, 15.86, and 16.5 percentage points in Accuracy, F1-score, and MCC, respectively. Compared with traditional machine-learning approaches such as Func_{RF} and Func_{LGBM} , which rely solely on static feature extraction, ReVul-CoT demonstrates superior adaptability in capturing semantic and contextual dependencies between vulnerabilities. Likewise, compared to text-only models (e.g., CWM_{NB} , CWM_{SVM}) and bimodal fusion methods (e.g., MTLM, SVA-CL), ReVul-CoT benefits from its retrieval-augmented reasoning mechanism, which dynamically integrates external contextual knowledge during inference rather than depending on fixed exemplars.

In addition to performance improvements, ReVul-CoT also demonstrates excellent computational efficiency. The overhead of vectorization and retrieval is significantly optimized through an offline embedding and indexing mechanism. Specifically, before inference, code snippets and vulnerability descriptions in the knowledge base are converted into semantic embeddings, which are then stored in a PostgreSQL database. A FAISS index is further built to enable efficient retrieval and similarity computation during runtime. Although the embedding and indexing processes require some initial computational resources, this cost occurs only once. In later vulnerability assessments, the mechanism can retrieve relevant samples and compute similarities in real time with minimal latency, thereby achieving a balance between accuracy and efficiency.

Table 2: Comparison results of ReVul-CoT and baselines across three evaluation metrics.

Approach	Accuracy (%)	F1-score (%)	MCC (%)
Func _{RF}	62.97	43.69	37.25
Func _{LGBM}	69.39	62.48	48.51
CWM _{NB}	62.44	49.26	37.98
CWM _{SVM}	68.16	58.53	46.74
CWM _{LR}	69.32	62.55	49.42
Cvss-bert	70.31	61.26	50.54
SPSV	72.42	61.76	53.68
SVA-CL	72.55	62.43	56.13
MTLM	74.54	64.85	59.52
SVA-ICL	77.07	67.89	63.01
ReVul-CoT	87.50	83.75	79.51

Summary for RQ1: The experimental results demonstrate that ReVul-CoT outperforms state-of-the-art baselines in the software vulnerability assessment task. Compared with the second-best baseline, ReVul-CoT achieves improvements of 10.43, 15.86, and 16.50 percentage points in Accuracy, F1-score, and MCC, respectively.

5.2. RQ2: Whether considering both source code and vulnerability description with information can improve the performance of ReVul-CoT?

Approach. For RQ2, we investigate the impact of source code and vulnerability description information on the performance of ReVul-CoT. We separately compute the similarity of source code and vulnerability descriptions, and then combine these similarity scores using different weighting ratios according to Equation (3) to evaluate the contribution of each modality to the model’s retrieval and generation performance. In our experiments, we explore several weighting schemes for integrating the two types of information, including using only vulnerability descriptions (0%:100%), equal weighting of both modalities (50%:50%), and using only source code (100%:0%).

Result. The performance comparison results of ReVul-CoT under different fusion ratios between code similarity and text similarity are presented in Table 3. The results show a clear trend: when the contribution of both modalities is balanced, the model achieves superior and more stable performance across all evaluation metrics. In particular, when the code-to-text similarity ratio is set to 60%:40%, ReVul-CoT achieves its best results—87.50% Accuracy, 83.75% F1-score, and 79.51% MCC, demonstrating that moderate weighting between the two modalities yields the optimal integration effect. When the ratio is biased toward a single modality, performance degrades notably. When only textual similarity (0%:100%) is considered, ReVul-CoT performs poorly across all three metrics. This is because vulnerability descriptions alone may contain redundant or ambiguous natural language content, which causes the model to misinterpret contextual severity cues during reasoning. Conversely, when only code similarity (100%:0%) is used, performance also declines, as code features provide structural but

not semantic information about exploitability and impact scope. These observations indicate that source code and textual descriptions offer complementary information: code captures syntactic and functional structure, whereas descriptions convey high-level semantic context.

Table 3: Experimental results of ReVul-CoT with varying fusion ratios between description similarity ($\text{CosSim}_{\text{desc}}$) and vulnerability source code similarity ($\text{CosSim}_{\text{code}}$).

CosSim_{code}	CosSim_{desc}	Accuracy (%)	F1-score (%)	MCC (%)
100%	0%	72.52	63.96	54.34
90%	10%	82.78	78.34	71.72
80%	20%	85.10	82.54	75.50
70%	30%	86.59	82.97	78.03
60%	40%	87.50	83.75	79.51
50%	50%	86.84	83.19	78.41
40%	60%	86.67	82.67	78.16
30%	70%	<u>86.92</u>	84.24	<u>78.60</u>
20%	80%	86.75	84.03	78.35
10%	90%	86.42	<u>84.20</u>	77.73
0%	100%	86.51	82.96	77.87

In addition to the quantitative evaluation, we also conduct a qualitative analysis using two representative cases, as illustrated in Figure 4. These examples further verify how different modality ratios influence the reasoning accuracy of ReVul-CoT. In Case 1, ReVul-CoT produces inaccurate predictions when relying solely on source code or exclusively on vulnerability descriptions. More precisely, under the code-to-text ratio configurations of 100%:0% or 0%:100%, the model predicts the severity level as High, which does not match the ground-truth label (i.e., Medium). However, when both modalities are considered simultaneously with a balanced ratio of 60%:40%, ReVul-CoT makes the correct assessment (i.e., Medium). This demonstrates that combining code and description information enables the model capture both structural and semantic cues, thereby improving its judgment accuracy in SVA. In Case 2, when the source code similarity is excluded (i.e., ratio is set to 0%:100%), ReVul-CoT returns an incorrect evaluation since the ground truth is Critical, whereas the model predicts High. Further analysis suggests that this error is caused by redundant information within the vulnerability description, which hinder the LLM from accurately assessing the severity level. However, when source code similarity is included, ReVul-CoT provides the correct Critical assessment. This again confirms that fusing both source code and vulnerability description information allows the model to reason more effectively in SVA.

Summary for RQ2: Compared with considering a single modality, integrating source code and vulnerability description information leads to significantly better performance in ReVul-CoT. The results demonstrate that a balanced weighting of 60%:40% between code and text similarities achieves optimal performance.

Case1:	Code: int kvm_read_guest_virt(struct kvm_vcpu *vcpu, gva_t addr, void *val, unsigned int bytes, struct x86_exception *exception) { u32 access; access = (kvm_x86_ops->get_cpl(vcpu) == 3) ? PFERR_USER_MASK : 0; return kvm_read_guest_virt_helper(addr, val, bytes, vcpu, access, exception); }	Description: The KVM implementation in the Linux kernel through 4.20.5 has an Information Leak.	×
	ReVul-CoT (CosSim_{code}=100%, CosSim_{desc}=0%) : HIGH		×
	ReVul-CoT (CosSim_{code}=0%, CosSim_{desc}=100%) : HIGH		×
	ReVul-CoT (CosSim_{code}=60%, CosSim_{desc}=40%) : MEDIUM		✓
	Ground Truth: MEDIUM		
Case2:	Code: inline void StringData::setSize(int len) { assertx(!isImmutable() && !hasMultipleRefs()); assertx(len >= 0 && len <= capacity()); mutableData()[len] = 0; m_lenAndHash = len; assertx(m_hash == 0); assertx(checkSane()); }	Description: xbuf_format_converter, used as part of exif_read_data, was appending a terminating null character to the generated string, but was not using its standard append char function. As a result, if the buffer was full, it would result in an out-of-bounds write. This issue affects HHVM versions prior to 4.56.3, all versions between 4.57.0 and 4.80.1, all versions between 4.81.0 and 4.93.1, and versions 4.94.0, 4.95.0, 4.96.0, 4.97.0, 4.98.0.	×
	ReVul-CoT (CosSim_{code}=100%, CosSim_{desc}=0%) : HIGH		×
	ReVul-CoT (CosSim_{code}=0%, CosSim_{desc}=100%) : CRITICAL		✓
	ReVul-CoT (CosSim_{code}=60%, CosSim_{desc}=40%) : CRITICAL		✓
	Ground Truth: CRITICAL		

Figure 4: Two representative cases of the base severity predicted by our proposed ReVul-CoT under different similarity settings (i.e., considering only source code, only vulnerability descriptions, and both with the best ratio).

5.3. RQ3: How does the number of similar samples in RAG retrieval affect the performance of ReVul-CoT?

Approach. In RQ3, we investigate the impact of the number of retrieved samples on the performance of ReVul-CoT. Specifically, during the retrieval stage, the retrieval mechanism selects the top- k most similar samples to the target vulnerability, where k is set to 0, 3, 5, and 7, corresponding to the zero-shot (without retrieval), 3-shot, 5-shot, and 7-shot configurations, respectively. According to the findings from RQ2, the weighting code-to-text ratio is fixed at 60%:40% in this RQ. To ensure a fair comparison, the same prompt template is used across all experiments with different numbers of retrieved samples.

Result. We show the performance of ReVul-CoT with different numbers of retrieved samples in Table 4. As illustrated in the table, the number of retrieved samples has a significant

Table 4: The performance variation of ReVul-CoT under different numbers of retrieved samples.

Setting	Accuracy (%)	F1-score (%)	MCC (%)
without RAG(zero-shot)	41.14	31.45	12.27
3-shot	83.11	77.76	72.49
5-shot	87.50	83.75	79.51
7-shot	79.88	73.02	67.24

impact on model performance. When no retrieval is performed (i.e., RAG is not enabled), the model performs poorly, indicating that the lack of contextual examples prevents the LLM from effectively understanding and reasoning about vulnerability severity. With an increasing number of retrieved samples, ReVul-CoT exhibits noticeable performance gains, indicating that integrating multiple contextually related examples enables the model to more effectively grasp semantic structures and contextual relationships. Setting the number of retrieved samples to five yields the best overall performance, striking an optimal balance between contextual diversity and reasoning coherence. However, when too many samples are retrieved, the model’s performance begins to degrade. This decline is primarily caused by information redundancy and interference, which may lead the model to ignore key details and introduce noise during the reasoning process. Therefore, retrieving five similar samples (5-shot) enables ReVul-CoT to achieve the most stable and accurate prediction results.

Summary for RQ3: Increasing the number of retrieved samples can improve the performance of ReVul-CoT to a certain extent. However, when the number of samples exceeds five, performance declines due to information redundancy and interference. Our study shows that using 5-shot achieves the optimal balance between contextual diversity and reasoning stability.

5.4. RQ4: How does the use of Chain-of-Thought prompting affect the performance of ReVul-CoT?

Approach. To evaluate the effect of introducing CoT into the ReVul-CoT framework, we incorporate a CoT prompting mechanism into the model to guide the LLM in performing intermediate reasoning steps before generating the final prediction, thereby forming a more structured decision-making process. In our experimental design, we establish two comparative settings: one using CoT reasoning and the other without it. Both settings are kept identical in all other aspects to ensure that any performance differences can be only attributed to the introduction of CoT.

Result. The experimental results, as shown in Table 5, indicate that enabling CoT reasoning leads to consistent improvements in Accuracy, F1-score, and MCC. This enhancement can be primarily attributed to the structured reasoning process introduced by CoT prompting. By explicitly decomposing the decision-making process into intermediate steps, the model avoids shallow pattern matching and instead focuses on the causal relationships among vulnerability attributes, thereby improving prediction accuracy.

Table 5: Experimental results of ReVul-CoT with and without Chain-of-Thought prompting.

Setting	Accuracy (%)	F1-score (%)	MCC (%)
without CoT	85.19	81.05	75.67
with CoT	87.50	83.75	79.51

Furthermore, qualitative observations reveal that the model with CoT reasoning demonstrates more stable and consistent reasoning behavior when handling vulnerabilities with ambiguous descriptions or overlapping severity features. In contrast, the model without CoT is more prone to abrupt or contradictory predictions, indicating that its reasoning relies more on surface-level statistical correlations rather than logical analysis. In addition, CoT helps reduce hallucination phenomena and improves confidence calibration. Because the reasoning process includes explicit intermediate justifications, the model can self-correct inconsistencies before producing the final output, resulting in more reliable vulnerability severity assessments. This finding is consistent with previous studies [57, 58] on reasoning-enhanced LLMs, which demonstrate that structured reasoning processes can improve both task generalization and interpretability .

Summary for RQ4: Introducing CoT prompting for reasoning effectively enhances the overall performance of ReVul-CoT. By guiding the model to perform structured and step-by-step reasoning, CoT improves logical consistency, reduces hallucinations, and enables more accurate and effective SVA.

5.5. RQ5: How does external knowledge enhancement affect the performance of ReVul-CoT?

Approach. To explore the impact of external knowledge enhancement on the performance of ReVul-CoT, we incorporate domain-specific vulnerability knowledge into the model during both the knowledge base construction and reasoning processes. This external knowledge is derived from public vulnerability databases and contains relevant information for individual vulnerabilities. During inference, the model retrieves vulnerabilities that are semantically similar to the target one, each of which carries external knowledge, and integrates this information into the reasoning process. This allows the model to combine internal semantic representations with external factual knowledge, enabling a more comprehensive severity assessment. A control experiment without external knowledge is also conducted to evaluate the performance improvement brought by external knowledge enhancement.

Result. The results in Table 6 show that ReVul-CoT achieves better performance when external knowledge enhancement is applied. The introduction of external knowledge enables the model to perform more accurate and reliable vulnerability severity assessments. This performance improvement mainly results from the model’s access to richer factual and contextual information. By integrating vulnerability-related knowledge, ReVul-CoT can better capture subtle relationships between code

semantics and descriptive text, thereby reducing reasoning ambiguity—particularly in cases where vulnerability descriptions are incomplete or semantically unclear. Furthermore, the incorporation of external knowledge enhances the completeness and consistency of the reasoning process, allowing the model to make judgments based on authoritative domain knowledge rather than relying solely on linguistic patterns. Specifically, the incorporation of external knowledge enhances the completeness and consistency of the reasoning process, allowing the model to make judgments based on authoritative domain knowledge rather than relying solely on linguistic patterns.

Table 6: Comparative performance of ReVul-CoT with and without external knowledge enhancement.

Setting	Accuracy (%)	F1-score (%)	MCC (%)
without external knowledge	81.54	76.07	69.93
with external knowledge	87.50	83.75	79.51

Summary for RQ5: Incorporating external knowledge enhances the performance of ReVul-CoT. The results demonstrate that external knowledge enrichment effectively complements the model’s semantic understanding and strengthens its reasoning capability in SVA.

6. Discussions

6.1. Comparison with Other Popular LLMs

In this section, we selected DeepSeek-V3.1 as the backbone LLM for the ReVul-CoT framework. To further validate its effectiveness in SVA, we also compared it with three popular LLMs, including GLM-4.5, Gemini-2.5-Pro, and Qwen3-Coder-30B-A3B-Instruct. These models have been widely adopted in various software engineering tasks, and thus provide valuable benchmarks for comparison under the same experimental configurations.

Table 7: Performance comparison between DeepSeek-V3.1 and other popular LLMs.

Setting	Accuracy (%)	F1-score (%)	MCC (%)
GLM-4.5	81.37	73.97	69.80
Gemini-2.5-Pro	<u>83.69</u>	<u>78.60</u>	<u>73.42</u>
Qwen3-Coder-30B-A3B-Instruct	80.55	69.08	68.43
DeepSeek-V3.1	87.50	83.75	79.51

As shown in Table 7, DeepSeek-V3.1 achieves the best overall performance across all evaluation metrics. In comparison, Gemini-2.5-Pro ranks second, demonstrating relatively strong results due to its larger context window and enhanced reasoning capabilities. GLM-4.5 performs moderately well but exhibits certain limitations in reasoning depth and contextual fusion. Meanwhile, Qwen3-Coder-30B-A3B-Instruct, although competitive in code comprehension, performs less effectively in integrating heterogeneous information such as source code and textual descriptions, primarily due to its relatively smaller parameter size.

The excellence of DeepSeek-V3.1 results from the combined influence of multiple key factors: its extended context window, stronger code analysis and language understanding capabilities, and enhanced reasoning ability, which make it highly suitable for integration with CoT prompting. These comparative results confirm the competitiveness of DeepSeek-V3.1 within the ReVul-CoT framework and demonstrate that the proposed approach achieves a balanced trade-off among reasoning quality, contextual comprehension, and computational efficiency across different LLM architectures.

6.2. Evaluation of Token Usage

Token usage is an important factor when evaluating the efficiency and practicality of LLM-based frameworks [12]. In this study, we conducted a systematic analysis of token consumption in ReVul-CoT, using DeepSeek-V3.1 as the backbone model. The evaluation is based on the optimal experimental configuration in our study, where the similarity fusion ratio between code and text is set to 6:4, the retrieval parameter is fixed at Top- $k = 5$, and Chain-of-Thought prompting is enabled to guide the model in performing hierarchical and structured reasoning.

In our study, the total input tokens refer to the complete input sequence fed into the LLM, which mainly consists of four components: (1) the textual description of the target vulnerability; (2) its corresponding source code snippet; (3) the contextual information of the Top-5 most similar samples retrieved from the knowledge base; and (4) the Chain-of-Thought prompting template used to guide reasoning. As illustrated in Figure 5, among these components, the retrieved samples and their contextual information are the primary source of token growth, while CoT prompts further extend the input length by introducing additional reasoning instructions. Under this configuration, each query’s average input includes the vulnerability description, source code, retrieved Top-5 sample contexts, and the CoT prompt. Experimental results show that the average token consumption per input is approximately 8,246 tokens, with a minimum of 2,505 and a maximum of 68,666 tokens. The retrieved context accounts for an average of 87% of the total token usage, while the remaining tokens come from the vulnerability description, source code, and CoT reasoning template. Although retrieval introduces additional overhead, it substantially enriches the contextual grounding of the model, thereby improving the accuracy of vulnerability severity assessment. Meanwhile, the CoT prompting also introduces a moderate increase in token usage [15], but this increase further enhances the transparency of reasoning and the stability of the results.

Through a well-designed experimental setup, ReVul-CoT achieves a balance between token usage efficiency and prediction performance. Although the introduction of retrieval and CoT prompting inevitably increases input length, this additional cost effectively enhances reasoning quality and classification accuracy. In this study, our analysis focuses solely on input tokens, i.e., the complete input sequence fed into the LLM. While output tokens are also influenced by CoT prompting (e.g., intermediate reasoning outputs), they are not included in this measurement, as our primary objective is to evaluate the efficiency

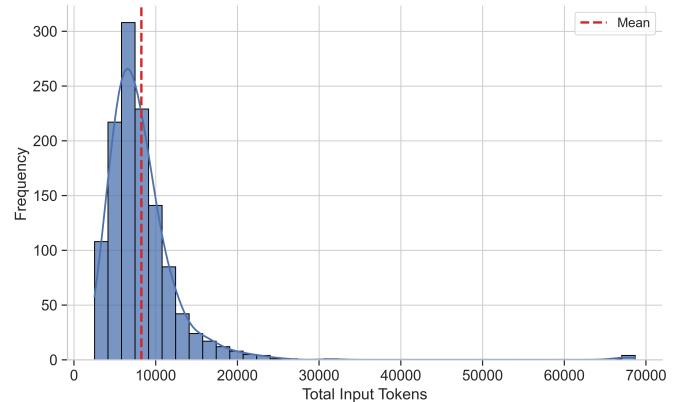


Figure 5: Distribution of total input tokens in the framework ReVul-CoT based on DeepSeek-V3.1, notice the histogram shows the total token consumption per input sample.

and practicality of prompt design within a retrieval-augmented and reasoning-driven framework.

6.3. Threats to Validity

In this subsection, we discuss the potential threats to the validity of our study.

Internal Threats. As LLMs are generally built upon extensive datasets derived from publicly accessible sources, they inherently carry a potential risk of unintended data exposure. In our experimental setup, this could occur if the model had previously encountered vulnerability samples resembling those in our test dataset, thereby recalling prior patterns rather than performing genuine reasoning. However, the observed weak zero-shot (without RAG) performance suggests that any potential data leakage had an extremely low impact on our experimental results.

External Threats. One major external threat stems from the dependence on the LLM API version. Since different versions or providers of LLMs may lead to variations in performance, it is difficult to ensure that all versions can reproduce the results obtained in this study. To enhance reproducibility, we clearly indicate the use of the DeepSeek-V3.1 API and release open-source resources for verification. Another potential threat originates from the dataset. Although our dataset integrates NVD and CWE knowledge to achieve wide coverage, it is still confined to publicly available CVE records. Consequently, unreported vulnerabilities or those tied to proprietary systems might be missing, which could limit generalization. For vulnerability types with few historical samples, performance may fluctuate. To alleviate this issue, our framework is continuously updated with new vulnerabilities and revised scoring standards (e.g., CVSS updates), ensuring long-term adaptability and improvement.

Construct Threats. Construct threats are related to how we measure and interpret model performance. In our evaluation, we employed widely used metrics—Accuracy, F1-score, and

MCC—which capture complementary perspectives on classification performance. Additionally, the construction of the CoT prompt introduces potential construct threats, since prompt design choices may shape the reasoning chain. We mitigated this risk by grounding the prompt in established CVSS criteria and CWE taxonomies, ensuring that the reasoning process aligns with domain standards.

Conclusion Threats. Conclusion threats concern the scope and the extent to which our findings can be generalized. Our dataset primarily focuses on vulnerabilities written in C and C++, as these languages dominate in security-critical systems. However, the proposed framework ReVul-CoT is not limited to these programming languages and is extendable to vulnerability analysis in other programming languages, including Java and Python. Moreover, although we compared ReVul-CoT against state-of-the-art baselines, some recent methods could not be included due to the unavailability of source code or proprietary implementations, which may introduce bias in comparative conclusions. To address this limitation, we selected representative baselines from three categories (code-based, description-based, and hybrid approaches) and conducted detailed ablation studies to reinforce the validity of our conclusions.

7. Related Work

Software vulnerability assessment seeks to evaluate the severity of software vulnerabilities by analyzing vulnerability-related data such as source code and textual descriptions, so that developers can prioritize high-risk vulnerabilities and mitigate potential security threats. Existing studies can generally be categorized into three groups: (1) description-based SVA, (2) source code-based SVA, and (3) Bimodal data-based SVA.

Description-Based SVA. These researches focused on utilizing textual vulnerability descriptions to estimate severity. Han et al. [59] leveraged word embeddings and convolutional neural networks (CNNs) to capture discriminative words and syntactic patterns for vulnerability severity classification. Spanos et al. [47] designed a multi-objective model that integrates text analysis with severity prediction. Liu et al. [60] introduced deep learning–driven classifiers to perform vulnerability text classification. Later, Le et al. integrated both character-level and word-level features to enhance text representation, while Babau et al. [40] adopted a multi-task framework built upon a pre-trained BERT model, enabling simultaneous prediction of vulnerability severity levels and associated metrics. Shahid et al. [5] further improved interpretability by introducing CVSS-BERT, which predicts CVSS vector metrics and severity scores using multiple fine-tuned BERT classifiers with gradient-based saliency analysis.

Source Code-Based SVA. Parallel to text-based methods, researchers also investigated source-code-driven SVA, which directly analyzes vulnerable functions or code snippets to estimate severity. Ganesh et al. [61] evaluated traditional machine learning algorithms for vulnerability prediction but found low generalization due to code diversity. Le et al. [4] introduced a function-level evaluation approach that leverages contextual features of vulnerable statements, demonstrating that

code granularity substantially affects severity prediction. Hao et al. [62] introduced a graph-based approach that constructs function call and vulnerability attribute graphs, using graph attention networks to model interactions between sensitive APIs and vulnerable components.

Bimodal Data-Based SVA. To overcome the limitations of unimodal approaches, recent works have proposed bimodal SVA frameworks that jointly utilize both code and textual descriptions. Xue et al. [41] introduced SVACL, a prompt-tuning-based continual learning framework that integrates hybrid prompts from source code and vulnerability descriptions using CodeT5. SVACL effectively mitigates catastrophic forgetting while adapting to continuously evolving vulnerability datasets. Similarly, Du et al. [33] proposed a multi-task learning model (MTLM) combining GraphCodeBERT and Bi-GRU with attention mechanisms to process bimodal inputs, demonstrating that integrating heterogeneous information improves robustness and accuracy. More recently, Gao et al. [6] proposed SVA-ICL, which applies in-context learning to large language models, thereby enhancing the effectiveness of SVA.

However, existing LLM-based methods suffer from knowledge limitations and lack of reasoning transparency. To address these issues, RAG [12, 13, 14] has emerged as an effective solution for enriching LLMs with domain-specific knowledge, while chain-of-thought prompting [15] enhances interpretability through explicit reasoning chains. Despite their success in other domains, the integration of RAG and CoT into SVA remains largely unexplored.

Different from previous SVA studies [28], to our best knowledge, our proposed framework ReVul-CoT is the first to integrate RAG with CoT prompting for SVA. ReVul-CoT dynamically retrieves contextually relevant information from a local knowledge base enriched with NVD and CWE data, combining both source code and vulnerability descriptions. The retrieved knowledge guides the LLM through a structured reasoning process to evaluate exploitability, impact scope, and contextual factors under standardized CVSS criteria. Experiments demonstrate that ReVul-CoT outperforms SVA baselines across multiple performance metrics, validating its effectiveness. Furthermore, ablation studies confirm that both the retrieval module and CoT prompting are indispensable.

8. Conclusion and Future Work

In our study, we proposed a novel framework ReVul-CoT that integrates Retrieval-Augmented Generation with Chain-of-Thought prompting for software vulnerability assessment. ReVul-CoT dynamically retrieves contextually relevant information from a local knowledge base enriched with NVD and CWE data, enabling the large language model DeepSeek-V3.1 to leverage knowledge more effectively under Chain-of-Thought prompting. It performs structured reasoning based on standardized CVSS criteria to conduct comprehensive software vulnerability assessment. Experimental evaluations on the dataset show that ReVul-CoT achieves superior performance compared to existing SVA baselines across various evaluation metrics. Furthermore, ablation studies confirm that both the RAG-based re-

trieval module and CoT reasoning mechanism are essential for enhancing the overall effectiveness of the model.

Despite the promising performance of ReVul-CoT in SVA, several directions merit further exploration. First, we want to further expand and enrich the knowledge base in terms of scale and sources. Second, we want to investigate more efficient retrieval strategies to further enhance the overall efficiency and accuracy of ReVul-CoT. Third, we want to develop more refined CoT prompting strategies, leveraging diversified reasoning chains and prompt templates to guide the large language model toward reasoning that aligns more closely with the standards of the software assessment domain. Finally, we aim to extend ReVul-CoT to other programming languages to validate its cross-language generalization and further apply it to broader software security tasks, such as vulnerability remediation, exploit prediction, and patch generation.

CRediT authorship contribution statement

Zhijie Chen: Conceptualization, Methodology, Software, Validation, Data Curation, Writing-Original Draft. **Xiang Chen:** Conceptualization, Methodology, Writing -review & editing, Supervision. **Ziming Li:** Data curation, Software, Validation. **Jiacheng Xue:** Data curation, Software, Validation. **Chaoyang Gao:** Data curation, Software, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Zhijie Chen and Xiang Chen have contributed equally to this work and are co-first authors. Xiang Chen is the corresponding author. This research was partially supported by the National Natural Science Foundation of China (Grant no. 61202006), the Open Project of State Key Laboratory for Novel Software Technology at Nanjing University under (Grant No. KFKT2024B21) and the Postgraduate Research & Practice Innovation Program of Jiangsu Province (Grant nos. SJCX25_2005).

References

- [1] R. Croft, M. A. Babar, M. M. Kholoosi, Data quality for software vulnerability datasets, in: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), IEEE, 2023, pp. 121–133.
- [2] T. H. Le, H. Chen, M. A. Babar, A survey on data-driven software vulnerability assessment and prioritization, ACM Computing Surveys 55 (5) (2022) 1–39.
- [3] S. Elder, M. R. Rahman, G. Fringer, K. Kapoor, L. Williams, A survey on software vulnerability exploitability assessment, ACM Computing Surveys 56 (8) (2024) 1–41.
- [4] T. H. M. Le, M. A. Babar, On the use of fine-grained vulnerable code statements for software vulnerability assessment models, in: Proceedings of the 19th International Conference on Mining Software Repositories, 2022, pp. 621–633.
- [5] M. R. Shahid, H. Debar, Cvss-bert: Explainable natural language processing to determine the severity of a computer security vulnerability from its description, in: 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2021, pp. 1600–1607.
- [6] C. Gao, X. Chen, G. Zhang, Sva-icl: Improving llm-based software vulnerability assessment via in-context learning and information fusion, Information and Software Technology (2025) 107803.
- [7] P. Vaithilingam, T. Zhang, E. L. Glassman, Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models, in: Chi conference on human factors in computing systems extended abstracts, 2022, pp. 1–7.
- [8] J. Liu, C. S. Xia, Y. Wang, L. Zhang, Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, Advances in Neural Information Processing Systems 36 (2023) 21558–21572.
- [9] M. D. Purba, A. Ghosh, B. J. Radford, B. Chu, Software vulnerability detection using large language models, in: 2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW), IEEE, 2023, pp. 112–119.
- [10] X. Zhou, T. Zhang, D. Lo, Large language model for vulnerability detection: Emerging results and future directions, in: Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results, 2024, pp. 47–51.
- [11] G. Lu, X. Ju, X. Chen, W. Pei, Z. Cai, Grace: Empowering llm-based software vulnerability detection with graph structure and in-context learning, Journal of Systems and Software 212 (2024) 112031.
- [12] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in neural information processing systems 33 (2020) 9459–9474.
- [13] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, W.-t. Yih, Replug: Retrieval-augmented black-box language models, arXiv preprint arXiv:2301.12652 (2023).

- [14] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, Y. Shoham, In-context retrieval-augmented language models, *Transactions of the Association for Computational Linguistics* 11 (2023) 1316–1331.
- [15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, *Advances in neural information processing systems* 35 (2022) 24824–24837.
- [16] National vulnerability database, <https://nvd.nist.gov/> (2024).
- [17] Common weakness enumeration, <https://cwe.mitre.org/> (2025).
- [18] deepseek, <https://www.deepseek.com/> (2025).
- [19] M. Humayun, N. Jhanjhi, M. F. Almufareh, M. I. Khalil, Security threat and vulnerability assessment and measurement in secure software development., *Computers, Materials & Continua* 71 (3) (2022).
- [20] Y. Cao, X. Ju, X. Chen, L. Gong, Mcl-vd: Multi-modal contrastive learning with lora-enhanced graphcodebert for effective vulnerability detection, *Automated Software Engineering* 32 (2) (2025) 67.
- [21] L. Wang, G. Lu, X. Chen, X. Dai, J. Qiu, Sift: enhance the performance of vulnerability detection by incorporating structural knowledge and multi-task learning, *Automated Software Engineering* 32 (2) (2025) 38.
- [22] Z. Cai, Y. Cai, X. Chen, G. Lu, W. Pei, J. Zhao, Csvd-tf: Cross-project software vulnerability detection with tradaboost by fusing expert metrics and semantic metrics, *Journal of Systems and Software* 213 (2024) 112038.
- [23] Common vulnerability scoring system, <https://www.first.org/cvss/> (2025).
- [24] T. H. M. Le, B. Sabir, M. A. Babar, Automated software vulnerability assessment with concept drift, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), IEEE, 2019, pp. 371–382.
- [25] T. H. M. Le, D. Hin, R. Croft, M. A. Babar, Deepcva: Automated commit-level vulnerability assessment with deep multi-task learning, in: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2021, pp. 717–729.
- [26] M. R. Nowak, M. Walkowski, S. Sujecki, Support for the vulnerability management process using conversion cvss base score 2.0 to 3. x, *Sensors* 23 (4) (2023) 1802.
- [27] C. Jiang, X. Pan, G. Hong, C. Bao, Y. Chen, M. Yang, Feedback-guided extraction of knowledge base from retrieval-augmented llm applications (2025). arXiv: 2411.14110.
- [28] X. Du, G. Zheng, K. Wang, Y. Zou, Y. Wang, W. Deng, J. Feng, M. Liu, B. Chen, X. Peng, et al., Vul-rag: Enhancing llm-based vulnerability detection via knowledge-level rag, arXiv preprint arXiv:2406.11147 (2024).
- [29] Common vulnerabilities and exposures, <https://www.cve.org/About/Metrics/> (2025).
- [30] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, *Advances in neural information processing systems* 35 (2022) 22199–22213.
- [31] S. Diao, P. Wang, Y. Lin, R. Pan, X. Liu, T. Zhang, Active prompting with chain-of-thought for large language models, arXiv preprint arXiv:2302.12246 (2023).
- [32] Y. Nong, M. Aldeen, L. Cheng, H. Hu, F. Chen, H. Cai, Chain-of-thought prompting of large language models for discovering and fixing software vulnerabilities, arXiv preprint arXiv:2402.17230 (2024).
- [33] X. Du, S. Zhang, Y. Zhou, H. Du, A vulnerability severity prediction method based on bimodal data and multi-task learning, *Journal of Systems and Software* 213 (2024) 112039.
- [34] Y. He, X. Zhu, D. Li, H. Wang, Enhancing large language models for specialized domains: A two-stage framework with parameter-sensitive lora fine-tuning and chain-of-thought rag, *Electronics* 14 (10) (2025) 1961.
- [35] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, et al., Codebert: A pre-trained model for programming and natural languages, arXiv preprint arXiv:2002.08155 (2020).
- [36] X. Huang, H. Peng, D. Zou, Z. Liu, J. Li, K. Liu, J. Wu, J. Su, P. S. Yu, Cosent: consistent sentence embedding via similarity ranking, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 32 (2024) 2800–2813.
- [37] J. Zhao, X. Chen, G. Yang, Y. Shen, Automatic smart contract comment generation via large language models and in-context learning, *Information and Software Technology* 168 (2024) 107405.
- [38] K. Liu, X. Chen, C. Chen, X. Xie, Z. Cui, Automated question title reformulation by mining modification logs from stack overflow, *IEEE Transactions on Software Engineering* 49 (9) (2023) 4390–4410.
- [39] X. Ming, Similarities: similarity calculation and semantic search toolkit, <https://github.com/shibing624/similarities> (2022).
- [40] I. Babalau, D. Corlatescu, O. Grigorescu, C. Sandescu, M. Dascalu, Severity prediction of software vulnerabilities based on their text description, in: 2021 23rd international symposium on symbolic and numeric algorithms for scientific computing (SYNASC), IEEE, 2021, pp. 171–177.

- [41] J. Xue, X. Chen, J. Wang, Z. Cui, Towards prompt tuning-based software vulnerability assessment with continual learning, *Computers & Security* 150 (2025) 104184.
- [42] C. Ni, L. Shen, X. Yang, Y. Zhu, S. Wang, Megavul: Ac/c++ vulnerability dataset with comprehensive code representations, in: Proceedings of the 21st International Conference on Mining Software Repositories, 2024, pp. 738–742.
- [43] S. SARD, Software assurance reference dataset project,(2020).
- [44] Y. Zhou, S. Liu, J. Siow, X. Du, Y. Liu, Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks, *Advances in neural information processing systems* 32 (2019).
- [45] J. Fan, Y. Li, S. Wang, T. N. Nguyen, Ac/c++ code vulnerability dataset with code changes and cve summaries, in: Proceedings of the 17th international conference on mining software repositories, 2020, pp. 508–512.
- [46] S. Boughorbel, F. Jarray, M. El-Anbari, Optimal classifier for imbalanced data using matthews correlation coefficient metric, *PloS one* 12 (6) (2017) e0177678.
- [47] G. Spanos, L. Angelis, A multi-target approach to estimate software vulnerability characteristics and severity scores, *Journal of Systems and Software* 146 (2018) 152–166.
- [48] Comparing two k-category assignments by a k-category correlation coefficient, *Computational biology and chemistry* 28 (5-6) (2004) 367–374.
- [49] T. K. Ho, et al., Proceedings of 3rd international conference on document analysis and recognition, in: Proceedings of 3rd international conference on document analysis and recognition, 1995.
- [50] G. K. LightGBM, A highly efficient gradient boosting decision tree, Garnett IGaUVLaSBaHWaRFaSVaR, editor (2017).
- [51] S. J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, 2016.
- [52] C. Cortes, V. Vapnik, Support-vector networks, *Machine learning* 20 (3) (1995) 273–297.
- [53] S. H. Walker, D. B. Duncan, Estimation of the probability of an event as a function of several independent variables, *Biometrika* 54 (1-2) (1967) 167–179.
- [54] Z. Cheng, T. Xie, P. Shi, C. Li, R. Nadkarni, Y. Hu, C. Xiong, D. Radev, M. Ostendorf, L. Zettlemoyer, et al., Binding language models in symbolic languages, arXiv preprint arXiv:2210.02875 (2022).
- [55] Y. Wang, W. Wang, S. Joty, S. C. Hoi, Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation, arXiv preprint arXiv:2109.00859 (2021).
- [56] N. Nashid, M. Sintaha, A. Mesbah, Retrieval-based prompt selection for code-related few-shot learning, in: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), IEEE, 2023, pp. 2450–2462.
- [57] S. A. Akbar, M. M. Hossain, T. Wood, S.-C. Chin, E. M. Salinas, V. Alvarez, E. Cornejo, Hallumeasure: Fine-grained hallucination measurement using chain-of-thought reasoning, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024, pp. 15020–15037.
- [58] A. Kumar, H. Kim, J. S. Nathani, N. Roy, Improving the reliability of llms: Combining cot, rag, self-consistency, and self-verification, arXiv preprint arXiv:2505.09031 (2025).
- [59] Z. Han, X. Li, Z. Xing, H. Liu, Z. Feng, Learning to predict severity of software vulnerability using only vulnerability description, in: 2017 IEEE International conference on software maintenance and evolution (ICSME), IEEE, 2017, pp. 125–136.
- [60] K. Liu, Y. Zhou, Q. Wang, X. Zhu, Vulnerability severity prediction with deep neural network, in: 2019 5th international conference on big data and information analytics (BigDIA), IEEE, 2019, pp. 114–119.
- [61] S. Ganesh, T. Ohlsson, F. Palma, Predicting security vulnerabilities using source code metrics, in: 2021 Swedish workshop on data science (SweDS), IEEE, 2021, pp. 1–7.
- [62] J. Hao, S. Luo, L. Pan, A novel vulnerability severity assessment method for source code based on a graph neural network, *Information and Software Technology* 161 (2023) 107247.

Zhijie Chen is currently pursuing the Master degree at the School of Artificial Intelligence and Computer Science, Nantong University. repository mining.

Xiang Chen received the B.Sc. degree in the school of management from Xi'an Jiaotong University, China in 2002. Then he received his M.Sc., and Ph.D. degrees in computer software and theory from Nanjing University, China in 2008 and 2011 respectively. He is currently an Associate Professor at the School of Artificial Intelligence and Computer Science, Nantong University. He has authored or co-authored more than 170 papers in refereed journals or conferences, such as IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering and Methodology, IEEE Transactions on Reliability, Empirical Software Engineering, Information and Software

Technology, Journal of Systems and Software, Software Testing, Verification and Reliability, Journal of Software: Evolution and Process, Automated Software Engineering, Software - Practice and Experience, Science of Computer Programming, Computer & Security, Knowledge-based Systems, Engineering Applications of Artificial Intelligence, International Conference on Software Engineering (ICSE), International Conference on the Foundations of Software Engineering (FSE), International Conference Automated Software Engineering (ASE), International Symposium on Software Testing and Analysis (IS-STA), International Conference on Software Maintenance and Evolution (ICSME), International Conference on Program Comprehension (ICPC), International Symposium on Software Reliability Engineering (ISSRE) and International Conference on Software Analysis, Evolution and Reengineering (SANER). His research interests include software engineering, in particular software testing and maintenance, security vulnerability detection and understanding, large language models for software engineering, software repository mining, and empirical software engineering. He received two ACM SIGSOFT distinguished paper awards in ICSE 2021 and ICPC 2023. He is the editorial board member of Information and Software Technology. More information can be found at: <https://xchencs.github.io/index.html>.

Ziming Li is currently pursuing the Bachelor degree at the School of Artificial Intelligence and Computer Science, Nantong University. His research interests include software repository mining.

Jiacheng Xue is currently pursuing the Master degree at the School of Artificial Intelligence and Computer Science, Nantong University. Her research interests include software repository mining.

Chaoyang Gao is currently pursuing the Master degree at the School of Artificial Intelligence and Computer Science, Nantong University. His research interests include software repository mining.