Answer is Cheap, Show Me the Evidence! Augmenting Automated Vulnerability Assessment with Evidence

ANONYMOUS AUTHOR(S)

Software Vulnerability (SV) assessment is a vital phase in SV management, which characterizes discovered SVs to locate hot spots and prioritize their remediation. To reduce the overhead and latency of manual assessment, prior works have explored automatically predicting assessment results from SV reports (SVRs). However, existing approaches fail to process the information conveyed by the rich text content (e.g., screenshots and code snippets) embedded in SVRs and miss information about vulnerable projects. More importantly, they primarily focus on assessment accuracy while neglecting to provide explanations or evidence supporting their predictions. As a result, these approaches remain impractical in real-world settings, where imperfect accuracy necessitates manual validation. LLMs offer a promising opportunity to address this limitation by performing SV assessment while simultaneously providing supporting evidence. Nevertheless, our extensive evaluation reveals that mainstream LLMs perform poorly on SV assessment tasks, largely due to a lack of assessment-specific knowledge. To address the above challenges, we propose EAVA, a novel framework that effectively leverages LLMs to perform SV assessment and provide supporting evidence. EAVA employs specialized LLM agents to process rich text content in SVRs and incorporate information about vulnerable projects. EAVA builds a dedicated assessment LLM by injecting assessment-specific knowledge through finetuning. Specifically, we enable large-scale reasoning trajectory annotation using off-the-shelf LLMs and adopt a two-stage training paradigm, i.e., supervised instruction tuning to inject domain knowledge, followed by reinforcement learning to enhance the model's intrinsic reasoning capability. Evaluations on a newly collected SVR dataset demonstrate that EAVA outperforms the best-performing baseline by 5.3%-35.2% across multiple evaluation metrics. Ablation studies validate the effectiveness of our design choices for both assessment-specific model training and SV information enrichment. Finally, a user study with security experts confirms that the evidence provided by EAVA is useful and practical for real-world SV assessment.

ACM Reference Format:
Anonymous Author(s). 2026. Answer is Cheap, Show Me the Evidence! Augmenting Automated Vulnerability Assessment with Evidence. 1, 1 (January 2026), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the ever-growing volume of emerging software vulnerabilities (SVs), a critical challenge in SV management is to locate and prioritize the remediation of critical SVs [46], i.e., those tend to be exploited and cause substantial damage (e.g., Log4Shell [1]). Industry standards for SV handling processes (e.g., ISO/IEC 30111 [6]) require software vendors to promptly address critical SVs. In addition, vendors must comply with security Service Level Agreements (SLAs) [3, 9, 11], which mandate the mitigation of critical SVs within a short timeframe (e.g., 15 days).

SV assessment, a critical phase in SV management life-cycle [37, 46], corresponds to the process of prioritizing critical SVs. Specifically, the *assessment* phase characterizes SVs detected in the *detection* phase to locate the "hot spots", and supports to devise a prioritization plan for the *remediation* phase. Currently, Common Vulnerability Scoring System (CVSS) is the most widely used standard for SV assessment [13]. A common practice is to reference expert-curated CVSS

ACM XXXX-XXXX/2026/1-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Report Date: July 1, 2024	**Disclosure Date:** Sept. 15, 2024
Report Title: [Bug]: Folder permissions are changed during synchronization ...	**CVE ID:** CVE-2024-46958
Report Body: **Bug description** ... simply create a new folder for example in the sync root folder. ... After the folder was synchronized, the rights have now changed, in this example ... **Steps to reproduce** 1. Create a new folder with only read, write and execute permission for the user. ... **Expected Behavior** After the synchronization no folder permissions are changed. ...	**CVE Description:** In Nextcloud Desktop Client 3.13.1 through 3.13.3 on Linux, synchronized files (between the server and client) may become world writable or world readable. ... **Initial Analysis Date:** Sept. 20, 2024 **Vector String**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N **CVSS Base Score:** 9.1 (CRITICAL)

Fig. 1. CVE-2024-46958 and its corresponding report

metric values published in SV databases (e.g., NVD [7]) when devising the remediation schedule for newly disclosed SVs [36, 42]. However, manually assigning values to the CVSS metrics for the ever-growing number of SVs is both labor-intensive and time-consuming. For example, the time lag between CVE disclosures and the availability of their CVSS metric values in the NVD has been increasing rapidly in recent years [36, 55]. This has motivated research on automated SV assessment [46, 49, 55]. In this work, we follow the majority studies [46] to use SV report (SVR) as the source to automate SV assessment (see Figure 1 for an example).

Existing approaches typically train machine-learning (ML) or deep-learning (DL) models to classify SVRs into specific metric values [46]. A significant drawback of existing assessment studies is their predominant focus on accuracy, i.e., how accurate the results given by automated approaches are. However, the performance of existing automated assessment approaches remains far from perfect [46, 55]. Manual validation and curation remain inevitable in drawing the final assessment results, especially given that assessment outputs are referenced to determine remediation schedules [46]. To effectively assist human analysts in performing efficient assessments, automated assessment approaches must not only provide their predictions but also present supporting evidence detailing their analysis in a manner that is easily comprehensible to humans. Furthermore, existing approaches fail to fully exploit comprehensive SV information in two key aspects. 1) They are unable to process information conveyed through rich-text content (e.g., screenshots and code snippets) embedded in SVRs. Such content is commonly used to illustrate attack steps and observed vulnerable behaviors, and is therefore essential for accurate SV assessment. 2) They overlook information about the affected project (e.g., its functionality and usage scenarios), which provides context for ensuring correct assessments [31].

Given that Large Language Models (LLMs) have shown strong reasoning capabilities [66] and have achieved promising results on SV-relevant tasks [69]. Applying LLMs to assess SVs and simultaneously present supporting evidence would be a promising direction. In this work, we take the first step to comprehensively investigate the LLM performance in SV assessment. Surprisingly, we find that applying LLMs to SV assessment is not straightforward. In fact, the performance of mainstream LLMs under various prompt settings falls significantly short of existing approaches by a wide margin. Specifically, we find that while general LLMs possess necessary background knowledge to analyze a given SVR, they lack assessment-specific knowledge (e.g., assessment criteria), which leads to their struggling performance in SV assessment.

To this end, we propose EAVA, a novel framework that effectively leverages LLMs to conduct SV assessment while presenting the supporting evidence. EAVA first utilizes three specialized LLM agents to analyze the code snippets and screenshots embedded in SVRs, and to retrieve information on vulnerable projects, respectively. Then, we build a dedicated LLM by effectively injecting assessment-specific knowledge, enabling it to autonomously analyze enriched SV information and make assessment decisions through structured reasoning. In particular, we address two key challenges: ❶ Acquiring assessment-specific knowledge (e.g., assessment criteria, associations between metric values and specific SV features) at scale. Although expert-curated CVSS metric

values are available in existing SV databases, no dataset explains how these values are derived from SV features. Producing such reasoning annotations with security experts would be prohibitively expensive. To enable large-scale data construction using general-purpose LLMs, we ease the task of reasoning out the correct metric value into an *open-book* problem. We provide the LLM with the ground-truth metric value and the key attributes that distinguish it from alternative values, and prompt it to articulate the reasoning process based on the given SV information. ❷ Instilling assessment-specific knowledge into the LLM and developing its intrinsic reasoning capability. Compared with the *open-book* annotation task, the *closed-book* SV assessment task, i.e., autonomously analyzing an SV and inferring its metric values, is substantially more challenging. Unlike prior efforts that transform general-purpose LLMs into domain specialists through a purely instruction-tuning "teacher–student" paradigm [54, 67], our approach augments instruction tuning with reinforcement learning to cultivate the model's inherent reasoning ability through online self-exploration [39]. Moreover, we explicitly investigate whether learning to generate explicit reasoning trajectories improves assessment performance.

We evaluate EAVA on a newly collected SVR dataset, which consists of 6,446 SVRs from 1,986 projects. EAVA is evaluated against a comprehensive collection of baselines, including six ML baselines, two DL baselines, and three advanced LLM baselines. EAVA consistently and significantly outperforms all baselines, achieving improvements of 5.3%-35.2% over the best-performing baselines across various measures. Ablation experiments verify the effectiveness of key designs including the two-stage finetuning and the processing of rich text contents. We also conduct a user study, confirming that EAVA can effectively help security analysts improve assessment efficiency.

In summary, this paper makes the following contributions:

- We present the first comprehensive investigation of SV assessment using LLMs and find that their performance is limited by the lack of domain-specific knowledge. We further propose a framework, EAVA, to enable effective SV assessment using LLMs.
- EAVA provides explicit reasoning to justify its assessment decisions and achieves strong performance through dedicated model construction, including large-scale dataset annotation, instruction tuning, and reinforcement learning. To the best of our knowledge, this is the first application of reinforcement learning to SV assessment.
- EAVA employs specialized LLM agents to effectively process rich text content in SVRs and integrate contextual knowledge about vulnerable projects.
- Extensive evaluations demonstrate that EAVA significantly outperforms baselines, and a user study confirms its practical usefulness.

2 BACKGROUND AND RELATED WORK

In this section, we introduce two aspects of the related work.

SV Assessment with CVSS. CVSS is the *de facto* standard for SV assessment [13]. CVSS is composed of three metric groups [5]: 1) Base metrics, which capture the intrinsic properties of an SV; 2) Temporal metrics, which reflect characteristics that change over the lifetime of an SV; 3) Environmental metrics, which pertain to attributes of an SV that are specific to particular user environments. Public assessments of SV severity (e.g., NVD [7] and vulnDB [2]) focus exclusively on Base metrics, which represent the fundamental and unchanging characteristics of SVs over time and across different user environments. In this work, we also focus on the Base metrics. The Base group comprises eight specific metrics, as shown in Table 1, which characterize SVs from two general perspectives: exploitability and impact. After assigning values to each CVSS metric, an overall severity score is calculated using standard equations [5]. This score can be further mapped to a qualitative rating (see Figure 1 for an example).

Table 1. Base metric group of CVSS

Category	Metric Names	Metric Values
Exploitability	Attack Vector (AV)	Network, Adjacent, Local, Physical
	Attack Complexity (AC)	Low, High
	Privileges Required (PR)	None, Low, High
	User Interaction (UI)	None, Required
Impact	Confidentiality Impact (C)	None, Low, High
	Integrity Impact (I)	None, Low, High
	Availability Impact (A)	None, Low, High
-	Scope (S)	Unchanged, Changed
Overall	Severity	Low, Medium, High, Critical

Automated SV Assessment. Le *et al.* [46] conducted a comprehensive literature review summarizing research on data-driven SV assessment and prioritization. According to their survey, CVSS is the primary standard and SV reports are the mostly used source to develop automated SV assessment approaches. Existing studies generally train ML or DL classifiers to predict the overall severity metric [40, 45, 51] or the entire group of CVSS metrics [47–49, 55, 62]. Some recent and representative studies are: Le *et al.* [49] suggest that the prediction of specific CVSS metrics may involve shared features and, therefore, propose a model with a unified encoder for the prediction of all CVSS metrics. They demonstrate that adopting multitask learning outperforms developing separate models for each metric. Pan *et al.* [55] point out that specific CVSS metrics are strongly associated and further propose a prompt-tuning based approach (namely PROEVA) to exploit such relations for CVSS metrics prediction.

Different from existing works, we are the first to systematically investigate the application of LLMs for SV assessment. Additionally, we highlight the importance of providing evidence to justify predicted assessment results, a critical aspect overlooked by prior research.

3 MOTIVATION AND PRELIMINARIES

In this section, we outline the challenges faced by the current SV assessment works. Then, we conduct a comprehensive investigation into the application of LLMs in SV assessment.

3.1 Challenges for SV Assessment

Existing automated assessment approaches train ML or DL models to classify SVRs into metric values [46, 55], which suffer from the following three limitations:

Impracticality of assessment results without supporting evidence. Existing approaches [46, 49, 55] have exclusively focused on the assessment accuracy, i.e., how accurately the algorithms assign correct CVSS metric values to a given SV. However, manual validation and curation remain inevitable, as the reliability of assessment results is critical for guiding remediation schedules [34, 37, 46] and yet the performance of automated SV assessment still falls short of perfection. To assist security analysts in validating the automated assessment results, evidence supporting the model decision is essential. Unfortunately, this has been overlooked by existing studies.

We propose to provide supporting evidence (e.g., extracted keywords and phrases) from the given SV information, together with the necessary analysis to justify the model's predictions. This evidence highlights the key information relevant to SV assessment, enabling analysts to quickly develop a comprehensive understanding of the vulnerability and to efficiently validate the automated assessment results. Figure 3 illustrates an example, showing how the inclusion of detailed evidence and analysis (under the `[Clues]` and `[Reasoning]` tags) facilitates more effective manual validation and curation. LLMs have the potential to provide assessment results with evidence given

their strong reasoning capability [32, 66]. However, as we show in Section 3.2, it is not trivial to leverage LLMs to conduct SV assessment.

Ineffectiveness in processing SVRs. ❶ SVRs are not plain texts but embedded with multiple rich text content, e.g., screenshots and code snippets. 24.0% of our collected 6,446 SVRs (see Section 5.1) contain at least one screenshot, and 10.1% contain over three screenshots. For code snippets, 69.2% of our collected SVRs contain at least one, and 16.8% contain over five. Screenshots are commonly used to present attack steps and observed vulnerable behaviors. For example, the report [20] of CVE-2023-31890 (XML de-serialization without proper checks) uses a screenshot to show that the provided attack payload can successfully exploit SV to launch the calculator application. This screenshot helps to confirm that exploiting SV can result in arbitrary code execution, which is essential to determine the value of impact metrics. Code snippets embedded in SVRs could be Proof-of-Concept (PoC) snippets, Address Sanitizer (ASAN) reports, etc. However, existing approaches fail to process the information conveyed by rich text content. A common practice is to replace such content with placeholder tags [55], for example, substituting screenshot URLs with a generic `IMGTAG`. This abstraction discards critical details, such as attack steps or vulnerable behaviors, embedded in the rich content, which can in turn lead to incomplete understanding and inaccurate SV assessment. ❷ The code snippets embedded in SVRs are often lengthy and noisy. In our collected dataset, 22.1% of SVRs contain embedded code snippets exceeding 2,000 characters (approximately 500 tokens). These long snippets typically have low information density, yet they consume a substantial portion of the LLMs' context window and distract the model from other critical information, thereby increasing the likelihood of incomplete reasoning and inaccurate SV assessment results.

To address these challenges, we propose first leveraging specialized LLM agents to analyze the screenshots and code snippets embedded in SVRs, and then replacing these rich textual contents with concise analytical summaries before performing the final assessment. This pipeline decomposes the complex analysis of SVRs and reduces the context length required at each analysis stage.

Lack of information regarding the vulnerable project. Existing approaches rely solely on the SV report itself as input, overlooking contextual information about the vulnerable project (e.g., its functionality and usage scenarios), which is often helpful for accurate SV assessment. For the SV shown in Figure 1, the report only states that the vulnerability involves file system synchronization. With this limited information, *Local*, *Adjacent*, and *Network* are all plausible values for the *Attack Vector* metric. However, knowing that the affected project is `nextcloud/desktop`, a cloud storage and file-sharing application, makes it clear that *Network* is the correct value. Vulnerable project information is recognized by established standards (e.g., NIST 800-30 [64] and Common Criteria [43]) and prior studies [31] as essential context for ensuring accurate SV assessment.

We propose enriching SVRs with information about the vulnerable projects to enable a more comprehensive assessment.

3.2 General LLM for SV Assessment

A straightforward approach to obtaining assessment results, supported by evidence, is to prompt the LLM to reason over the input SV information and draw a conclusion. However, after comprehensive evaluations, we find that off-the-shelf LLMs struggle to make correct assessments. Specifically, we evaluate the performance of three advanced LLMs (i.e., Llama-3.3-70b [10], DeepSeek-v3 [52], and GPT-4.1 [24]) under the following four prompt methods: ❶ zero-shot vanilla prompting, ❷ zero-shot Chain-of-Thought (CoT) prompting [66], ❸ few-shot vanilla prompting, ❹ few-shot CoT prompting. In total, our evaluation considers 12 distinct combinations. Refer to Section 5 for detailed evaluation settings (e.g., evaluation set and metrics).

Figure 2 shows the prompt templates used in our evaluation. The templates are parameterized with input-specific content, indicated by underlined text. Each prompt consists of three segments:

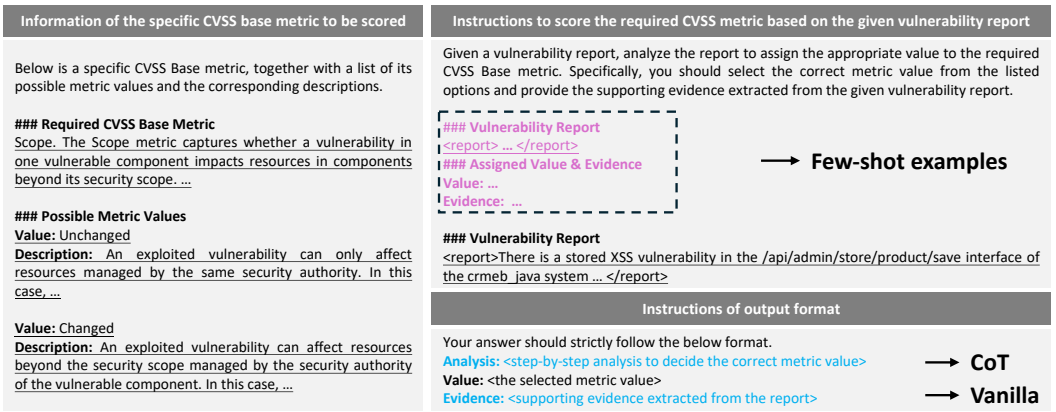


Fig. 2. Prompt templates used in the preliminary study

- **Information of the specific CVSS metric to be scored.** We prompt the LLM to score one CVSS metric at a time, since each metric characterizes an independent aspect of an SV. We supply LLM with the metric introduction and candidate metric values along with their descriptions extracted from the CVSS specification [5]. In preliminary experiments, omitting this detailed metric information led to substantially worse performance.
- **Instructions to score the required CVSS metric.** We instruct the LLM to first analyze the given SVR, then select the correct metric value from the listed options, and provide supporting evidence. For few-shot prompt setting, to the best of our knowledge, the only public source that provides evidence or explanations for the assigned CVSS metric value is the official CVSS document [17]. The document presents 31 examples with expert-curated evidence to demonstrate how to apply CVSS to score specific SVs. We select SVs from these examples to form the few-shot prompt. Specifically, we first apply a hybrid retriever (see Section 4.3) to rank these examples based on their similarities to the given SVR and then select one for each candidate metric value.
- **Instructions of output format.** For vanilla setting, we instruct the LLM to first give the verdict and then further provide evidence. For CoT setting, we instruct the LLM to first perform a step-by-step reasoning before making the final verdict. The reasoning process is regarded as evidence justifying the model decision.

Table 2 presents the average performance on eight CVSS metrics for each LLM-prompting combination. The results show that the performances of all three general LLMs under various prompt settings are much worse than existing learning-based approaches (i.e., proEVA [55]). Moreover, we observe that neither stronger LLMs nor CoT prompting necessarily lead to improved assessment performance. Under zero-shot settings, Llama-3.3-70B achieves performance comparable to DeepSeek-V3 and GPT-4.1, despite the latter demonstrating substantially stronger performance on general benchmarks [22]. Under both zero- and few-shot settings, CoT prompting yields performance that is comparable to, or slightly worse than, that achieved with vanilla prompting across all three LLMs. In contrast, providing LLMs with additional domain knowledge through few-shot examples consistently and substantially improves the performance of all three models compared to their zero-shot counterparts.

To further understand the role of domain-specific knowledge in SV assessment, we manually examine cases that are correctly assessed by proEVA but misclassified by LLMs. We identify two primary reasons for these errors:

SV reports lack sufficient information to determine the metric value. In such cases, the assessment process is largely experience- or knowledge-based [4], with historical SVs with similar characteristics (e.g., root cause) serving as important information cues. A typical case is memory-related SVs (e.g., use-after-free, heap buffer overflow). Reports of this category of SVs often only present an ASAN output showing that the SV can be successfully triggered with the provided PoC, without detailing its potential impact (e.g., CVE-2024-6064 [21]). As a result, determining the values of the impact metrics for such SVs heavily relies on referencing similar historical cases. For example, similar to CVE-2024-6064, CVE-2022-27147 (a historical SV in the training set of proEVA) is also a use-after-free SV in the GPAC project, which shares the same values for all three impact metrics.

Table 2. The average performance on eight CVSS metrics for three LLMs under four prompt settings

Prompt	LLM	F1	MCC
Zero-Shot Vanilla	Llama	0.747	0.330
	DeepSeek	0.746	0.338
	GPT	0.736	0.358
Zero-Shot CoT	Llama	0.735	0.314
	DeepSeek	0.726	0.318
	GPT	0.740	0.362
Few-Shot Vanilla	Llama	0.778	0.402
	DeepSeek	0.783	0.429
	GPT	0.781	0.485
Few-Shot CoT	Llama	0.745	0.346
	DeepSeek	0.758	0.399
	GPT	0.765	0.456
proEVA		0.831	0.544

LLMs fail to identify implicit clues in the report signaling specific metric values or LLMs misinterpret the assessment criteria. A typical failure case involves determining the value of the *Scope* metric for XSS SVs. For stored XSS SVs, the *Scope* metric is always Changed [17], because the vulnerable component is the web server (where the malicious script is injected), whereas the impacted component is the victim's browser (where the script is executed). However, the best accuracy among all three LLMs under zero-shot vanilla/CoT promptings is only 15.3% (11 out of 72). Taking CVE-2023-1609 as an example, the corresponding report explicitly mentions that it is a stored XSS SV (see Figure 2). LLM correctly analyzes the SV but incorrectly assigns the value Unchanged by stating that *"Both the vulnerable component (the API endpoint that stores the payload into the database) and the impacted component (the front-end page that renders the payload) are managed by the same security authority (the crmeb_java application)"*. We observe that by providing similar examples (i.e., few-shot prompting), the accuracy of LLMs is largely improved (from 11 to 59), though still lower than proEVA (correctly predicts all 72 SVs).

We further observe that given the metric value (Changed), LLM can identify Stored XSS as one supporting evidence. Still taking CVE-2023-1609 as an example, LLM states that *"This indicates that an attacker can inject malicious code into the system, which can then be executed by other users, affecting resources beyond the initial security scope"*. This suggests that, although LLMs struggle to accurately reason out the appropriate metric value, they possess the necessary background knowledge to analyze SVs and identify relevant evidence for a given metric value. Identifying supporting evidence for a provided answer (i.e., a *open-book* problem) is considered easier and more straightforward than selecting the correct answer through reasoning (i.e., a *closed-book* problem).

To summarize, our preliminary findings show that ❶ it is not trivial to leverage general LLMs to conduct SV assessment. The primary reason is their lack of knowledge specific to SV assessment (e.g., historical SV information and assessment criteria); ❷ though we show that supplying LLM with the assessment knowledge (i.e., the metric introduction and the demonstrating examples) extracted from the CVSS official document improves the performance, the improvement is still limited. These findings motivate us to finetune LLMs to effectively inject domain-specific knowledge rather than directly prompting off-the-shelf LLMs to conduct assessment.

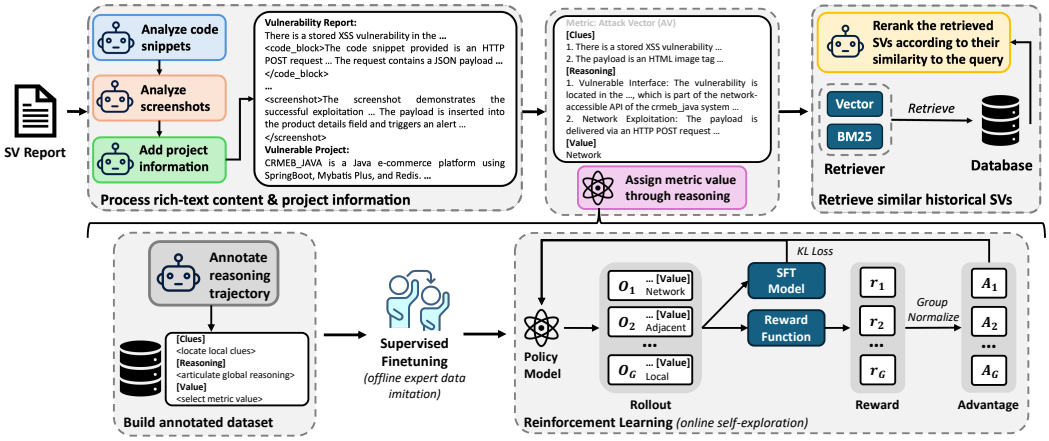


Fig. 3. Overview of EAVA

4 APPROACH

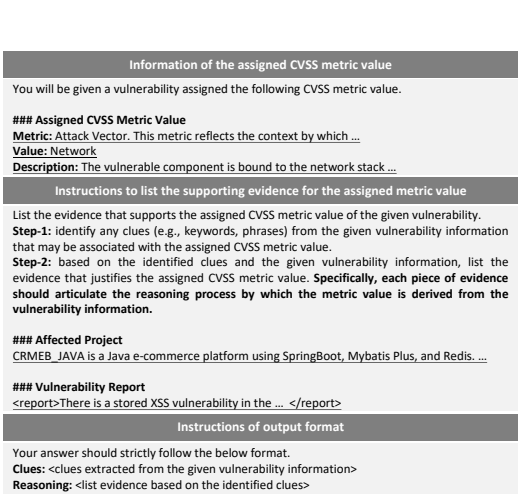
In this section, we introduce our proposed approach, namely EAVA. Figure 3 presents the overview of EAVA. Before applying EAVA to assess future SVs, we first construct a dataset by employing an LLM to annotate reasoning trajectories for disclosed SVs using analyst-curated assessment results. This annotated dataset is then used to finetune a dedicated assessment LLM and to build the database for retrieving similar historical SVs. Specifically, we adopt a two-stage finetuning paradigm that combines supervised finetuning with reinforcement learning. During inference, given an SV report (SVR), EAVA first employs three specialized LLM agents to analyze the embedded code snippets and screenshots, as well as to incorporate information about the vulnerable project. The resulting enriched SV information is fed into the finetuned assessment LLM to reason out the appropriate metric value. The generated reasoning trajectory is subsequently transformed into an index to retrieve similar historical SVs, which are provided as supplementary supporting evidence.

4.1 Processing Rich Text Content and Project Information

We build specialized LLM agents to analyze the rich text content in the SVR and summarize the information of the vulnerable project. The details are as follows:

Analyze code snippets. We first use regular expressions to extract embedded code snippets. Since replacing the original code snippet with a descriptive summary derived from the analysis may result in information loss, we limit our processing to exceptionally lengthy code snippets (i.e., those exceeding 500 characters) that exhibit low information density and are likely to distract LLMs from other crucial information during SV assessment. We analyze each extracted code snippet one at a time, following their order in the report. Specifically, considering that necessary context helps to enhance the understanding of information conveyed by the code snippet, we supply the preceding context from the SVR to the LLM and prompt it to describe the content of the code snippet under analysis. The prompt template (refer to online appendix [27]) is composed of the following three segments: ❶ Instructions to analyze the code snippet. We specify that the code snippet is presented in a SVR and the analysis should focus on SV-relevant aspects. ❷ Code snippet to be analyzed along with its preceding context. ❸ Instructions of output format. We instruct the LLM to conduct a two-step CoT-style analysis, i.e., a comprehensive look through followed by a brief summary.

Analyze screenshots. We first use regular expressions to extract URLs of screenshots embedded in SVRs. Then, we download the screenshot and encode it into a Base64 [12] string, which can be processed by LLM services. We use a multimodal LLM, which is capable of conducting image



(a) Prompt for annotating reasoning trajectories (b) Prompt for finetuning LLM for SV assessment

Fig. 4. Prompt templates used in data annotation and finetuning

comprehension, to perform the analysis. Similar to the analysis of code snippets, we analyze each extracted screenshot sequentially following their order in the report, and provide the preceding context to help LLM better interpret the conveyed information. The prompt template (refer to online appendix [27]) shares a similar design with the one used for code snippet analysis.

Add project information. We first crawl the project introduction from its homepage and then prompt the LLM to give a summarization emphasizing its functionalities and applications.

4.2 Building Dedicated Assessment LLM through Finetuning

To turn a general-purpose LLM into an SV assessment expert, we first construct large-scale reasoning trajectories that illustrate how expert-curated CVSS metric values are derived for historical SVs. We then apply supervised instruction tuning (SFT) to inject assessment-specific knowledge, followed by reinforcement learning (RL) to further enhance the model's reasoning capability.

4.2.1 Annotating Reasoning Trajectories. Although existing SV databases provide expert-curated CVSS metric values, there is currently no annotated dataset that captures the assessment process, specifically, how these metric values are derived from SV features. Constructing such annotations with the involvement of security experts would incur substantial cost. Leveraging LLMs for data synthesis has shown promise across various domains [67]. However, SV assessment requires substantial domain expertise, and our preliminary study (see Section 3.2) indicates that general-purpose LLMs struggle to produce accurate assessment results. To facilitate large-scale data annotation, we therefore reformulate the task of deriving the correct metric value as an *open-book* problem. Under this formulation, the LLM is provided with the ground-truth metric value along with key distinguishing attributes, and is prompted to articulate the reasoning trajectory based on the SV information. The detailed prompt template is shown in Figure 4a. The first segment of the prompt supplies information about the assigned CVSS metric value, including a brief introduction to the metric and the key features that distinguish the assigned value from other candidates, as extracted from the official CVSS specification [5].

To ensure both the comprehensiveness and faithfulness of the annotated reasoning trajectories, we design a structured prompt based on Chain-of-Thought (CoT) techniques [65, 66] that

decomposes the reasoning process into two steps, mirroring the analysis workflow of a security analyst. As illustrated in the second segment of the prompt in Figure 4a, we first instruct the LLM to examine the given SV information and identify relevant *clues* (i.e., local factual features such as keywords and phrases) indicative of the assigned metric value. After locating these clues, the LLM is further guided to synthesize the local evidence into a global understanding and to articulate the reasoning process by which the metric value is derived. This locate-and-analysis process offers the following benefits [65]: 1) decomposing the analysis allows LLMs to focus on individual sub-goals, leading to more comprehensive evidence localization and deeper reasoning; 2) initially collecting surface-level keywords and phrases ensures that the raw evidence, used as the starting point for subsequent reasoning, faithfully originates from the given SV information.

We apply this approach to annotate reasoning trajectories for 51,568 metric-level data points, corresponding to the eight CVSS metrics for each of the 6,446 SVRs collected in Section 5.1. This effort fills a significant data gap in demonstrating the SV assessment process. To validate the quality of the annotated data, we randomly sample 400 data points and engage two domain experts to interdependently evaluate each samples against two criteria: factuality (i.e., whether the evidence is grounded in the given SV information) and relevance (i.e., whether the evidence directly supports the assigned metric value). The validation process required a total of 56 person-hours. The two experts achieved a Cohen's Kappa score of 0.84, indicating substantial inter-rater agreement. Overall, 391 data points (97.8%) satisfy both quality criteria, confirming the effectiveness of our proposed automated annotation approach. In addition, we indirectly evaluate the quality of reasoning trajectories produced by the finetuned model through a userstudy (see Section 6.4).

4.2.2 Finetuning LLM for SV Assessment. We further leverage the annotated reasoning trajectories of historical SVs to train a dedicated LLM for SV assessment, which is substantially smaller than the model used for data annotation. Specifically, the model input is the enriched SV information from the previous step and the output is a reasoning process that leads to a determined metric value.

As demonstrated in our preliminary study, general-purpose LLMs lack assessment-specific knowledge. We therefore first apply supervised instruction tuning (SFT) [57] to inject domain knowledge at scale. The instruction templates used for fine-tuning are shown in Figure 4b. Specifically, we instruct the LLM to reason through three steps that correspond to the *locate-and-analysis* Chain-of-Thought used during reasoning trajectory annotation: 1) locating key clues, 2) outlining the reasoning process, and 3) selecting the correct metric value. The LLM is fine-tuned to imitate the annotated reasoning trajectories by optimizing the causal language modeling loss [59].

Although the instruction-tuned LLM can acquire some level of assessment reasoning ability by imitating expert trajectories, it still suffers from two inherent limitations. ❶ There exists a discrepancy between the *open-book* setting used during data annotation and the *closed-book* reasoning capability required at inference time. ❷ SFT is known to exhibit limited generalization ability, i.e., tuned models often struggle to transfer leaned reasoning patterns across different settings [33, 68]. This limitation is particularly pronounced in our SV assessment task due to the discrepancy between the annotation and inference settings. Thus, in contrast to prior efforts that transform general LLMs into domain specialists using a single teacher–student instruction-tuning paradigm [54, 67], we extend conventional SFT with reinforcement learning (RL). This design enables the model to develop intrinsic reasoning capabilities through online self-exploration [61].

Considering that the SV assessment task can be automatically and reliably verified, by comparing the model's predicted severity level with the ground truth, we adopt the the Group-relative Policy Optimization (GRPO) algorithm [61] to further refine the instruction-tuned model. Different from general RL algorithms like Proximal Policy Optimization (PPO) that requires training a separate value model [60], GRPO estimates advantages in a group-relative manner and can thus take

advantage of verifiable outcome to reduce complexity in reward computing and assign consistent reward signals. Specifically, we define the reward function as follows:

$$\mathcal{R}(o) = \begin{cases} 0, & \text{if the output format is invalid,} \\ 0.1, & \text{if } \text{value}^m_{\text{pred}} \text{ is not a valid metric value,} \\ 1.5 - \dfrac{\text{dis}(\text{value}^m_{\text{pred}}, \text{value}^m_{\text{gt}})}{\text{max\_dis}^m}, & \text{otherwise.} \end{cases} \tag{1}$$

Here, $\text{dis}(\text{value}^m_{\text{pred}}, \text{value}^m_{\text{gt}})$ measures the distance between the predicted and ground-truth metric values, and max_dis^m represents the maximum possible distance for the CVSS metric m. Taking *Confidentiality* metric as an example, max_dis equals two, corresponding to the distance between *None* and *High* values (see Table 1). This design encourages the LLM to generate outputs that are both syntactically valid and semantically close to the ground truth. Improperly formatted or invalid outputs are penalized, while predictions closer to the correct value yield higher rewards.

Given a training set $\mathcal{D}$ where each instance consists of 1) SVR, enriched SV information described in Section 4.1; 2) m, one of the eight CVSS metrics; 3) value_{gt}, the ground-truth metric value. The input prompt is formatted using the template shown in Figure 4b, denoted as $q = \text{prompt}(\text{SVR}, m)$. For each sampled instance, the policy LLM π_θ tries to assign the appropriate metric value through reasoning multiple times, i.e., generating G (the group size) outputs given the prompt q. Each output o_i is scored using the reward function defined above and normalized within the group to get the corresponding group-relative advantage A_i. Finally, the policy LLM is optimized by maximizing the following GRPO objective (see Figure 3):

$$\mathcal{J}(\theta) = \mathbb{E}\Bigg[\frac{1}{G} \sum_{i=1}^{G} \Bigg(\min\Big(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)} A_i,\ \text{clip}\Big(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)}, 1-\epsilon,\ 1+\epsilon \Big) A_i \Big) - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \Bigg) \Bigg], \tag{2}$$

$$\text{where} \quad (\text{SVR}, m, \text{value}_{\text{gt}}) \sim \mathcal{D},\ q = \text{prompt}(\text{SVR}, m),\ \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q).$$

Here, ϵ and β are hyperparameters, $\pi_{\theta_{\text{old}}}$ and $\pi_{\theta_{\text{ref}}}$ denote the old and reference policies, respectively. The ratio $\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$ measures the relative likelihood of generating the output o_i under the new policy π_θ versus the old policy $\pi_{\theta_{\text{old}}}$, and is used to correct the advantage A_i for distributional shift. To stabilize training, this ratio is clipped to $[1-\epsilon, 1+\epsilon]$, and the objective for each token is defined as the minimum of the clipped and unclipped terms. A KL-divergence penalty βD_{KL} regularizes the policy update to limit deviation from the reference policy. The token-level objectives are averaged over all tokens and outputs in the batch to form the final loss.

4.3 Retrieving Similar Historical SVs

During inference, we incorporate similar historical SVs to enrich the supporting evidence. Presenting analysts with historical SVs that share similar characteristics and are assigned with the same metric value helps strengthen the credibility of the model's predictions, particularly when SVRs provide limited information and assessments rely heavily on expert experience (see Section 3.2). The retrieval process is described below:

Index/Retrieval proxy. A straightforward way is to use the raw SVR to retrieve similar historical SVs. However, the raw SVRs are often noisy (e.g., containing code snippets, URLs), unstructured,

and include SV-irrelevant information (e.g., replication environments). Moreover, each CVSS metric captures a distinct aspect of an SV, whereas raw SVRs are not tailored to any specific metric, failing to achieve a precise metric-specific retrieval. We propose to utilize the evidence collected from the raw SVR and further organized by the LLM (i.e., the reasoning trajectory) as the retrieval proxy.

Retriever. We conduct a hybrid retrieval [53] by assembling a *sparse* retriever and a *dense* retriever. The sparse and dense retriever individually perform a search and return top-k historical SVs based on keywords and semantics, respectively. Their outputs are assembled to get the final retrievals.

LLM re-ranker. After using the built hybrid retriever to narrow down the pool of candidate historical SVs, we leverage an LLM as a reranker to further assess the relevance of the retrieved SVs to the query SV. Notably, we batch the retrieved historical SVs and query the LLM to rate their relevance scores simultaneously. Our initial experiments show that this batch design outperforms the one-by-one approach (i.e., rating the relevance score of each historical SV individually), possibly because it is more intuitive to determine the relative order of relevance than to maintain consistency in assigning absolute scores. Refer to online appendix [27] for the detailed prompt.

5 EXPERIMENT SETUP

5.1 Data Collection

We follow existing studies [55, 56, 58] to use GitHub issue reports referenced by CVE items as a proxy of SV reports (SVR). The detailed data preparation steps are as follows:

STEP1: Collect SV information. We first collect all CVE records from two widely recognized SV databases, i.e., NVD [7] and OSV [8], respectively. Notably, OSV is a well-known open-source software (OSS) SV database developed by Google. Considering that NVD and OSV may offer different SV metadata (e.g., CVSS, CWE, and external references), we use OSV solely as a supplementary source to enrich external references, while extracting all other metadata from NVD to ensure consistency. Using the CVE ID as the unified SV identifier, we merge the external references from both databases and extract GitHub issue links using regular expressions.

STEP2: Collect SVR information. Based on the extracted issue links, we crawl issue data (e.g., title, body, creation date) from GitHub. Then, we associate each crawled issue data with its corresponding CVE record and label it using the CVSS v3 metrics provided by the NVD.

STEP3: Data cleaning. We exclude the following SVR data: ❶ SVRs whose corresponding CVE records lack valid CVSS v3 metrics (i.e., labels). ❷ SVRs from GitHub projects with less than 100 stars. These projects are not widely used and are thus not representative. Besides, we observe that issues of these projects are likely to be of low quality. ❸ SVRs created after the disclosure date of the corresponding CVE records. These SVRs are outside the scope of early assessment, i.e., most of them are used to track the remediation process of disclosed SVs rather than initially reporting newly discovered SVs. Besides, these SVRs may reference information from the disclosed SVs, leading to potential data leakage. ❹ SVRs correspond to multiple CVEs. These SVRs are tangled (i.e., reporting multiple SVs simultaneously), which could confuse the assessment approach.

Finally, we collect 6,446 SVRs from 1,986 projects. For each project, we consider only its primary programming language as identified by the GitHub API. Our dataset spans 54 different programming languages, covering all of GitHub's top-10 languages [25]. Our dataset also includes 159 distinct CWE categories, encompassing the entire MITRE CWE Top-25 list [18]. Detailed distributions of programming languages and CWE categories are provided in the online appendix [27]. Collectively, these statistics demonstrate that our dataset is highly representative.

5.2 Experiment Setting

Implementation Details. Except for the dedicated assessment LLM (highlighted by the purple box in Figure 3), all other agents in EAVA are powered by Llama-3.3-70B [10] in our implementation.

The default temperature is set to 0.1 to ensure a consistent output. We fine-tune Llama-3.1-8B [23] to develop the dedicated assessment LLM. During the SFT stage, we set the learning rate to $1e^{-5}$ and train the model for two epochs. In the subsequent RL stage, the learning rate is reduced to $1e^{-6}$ and training proceeds for two epochs. For GRPO, we configure the rollout number to five, meaning that five outputs are sampled for each prompt.

Baselines. In experiments, we include the following baselines. ❶ *ML-based baselines.* ML classifiers are widely adopted for automated SV assessment [41, 46, 48]. Recent studies [41, 47] show that their performance is comparable to DL baselines in SV assessment tasks. We follow previous studies [48–50, 55] to include the following six ML baselines: Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), K-Nearest Neighbors (KNN), XGBoost (XGB), and Light Gradient Boosting Machine (LGBM). We adopt the same configuration and follow the same hyper-parameter tuning process as in previous studies [48–50, 55]. For detailed settings, please refer to our online appendix [27]. For each ML baseline, we report the test performance under the optimal hyper-parameter setting selected by performing grid search on the validation set. ❷ *DL-based baselines.* We include DeepCVA [49] and PROEVA [55] as two DL-based baselines. Refer to Section 2 for a brief introduction of these two approaches. We adopt the same hyper-parameters from existing works [49, 55] to configure these DL baselines. ❸ *LLM-based baselines.* We include three advanced LLMs, i.e., Llama-3.3-70b [10], DeepSeek-v3 [52], and GPT-4.1 [24], as baselines. We adopt the same few-shot prompt setting introduced in our preliminary study (see Section 3.2), which has been proven to achieve the best assessment performance among the various prompt settings.

Evaluation Metrics. We follow existing studies [47, 51, 55] to adopt classification metrics including weighted F1 and Matthews Correlation Coefficient (MCC) [38] to measure assessment performance. In addition to the performance on each of the eight individual CVSS metrics, we report two types of overall assessment performance: 1) the average performance across all eight CVSS metrics, 2) the performance in terms of severity rating, which is calculated from the predicted values of eight CVSS metrics using the official equations [5]. Note that the severity performance is not directly proportional to the average performance. This is because 1) the impact of different CVSS metrics on the overall severity rating varies, 2) opposing deviations in predictions of different CVSS metrics can offset each other, potentially resulting in no change to the overall severity rating. Thus, the average performance of the two approaches should be comparable before it becomes meaningful to further evaluate their performance in terms of severity rating.

6 EXPERIMENT RESULTS

In the experiment, we aim to answer the following RQs:

- **How effective is EAVA compared to baselines for SV assessment?**
- **How effective are the design choices for building a dedicated assessment LLM?**
- **How effective are the design choices for enriching SV information?**
- **Can the evidence provided by EAVA help security analysts in assessing SVs?**

6.1 RQ1. Effectiveness of EAVA for SV Assessment

Method. To verify the effectiveness of EAVA in SV assessment, we compare its performance with ML, DL, and LLM baselines using the dataset collected in Section 5.1. We divide the data set into train, validation, and test sets with a ratio of 8: 1: 1. Specifically, the dataset is split chronologically following existing works [41, 49, 55], i.e., the assessment approach is trained (if necessary) using historical SVs and tested to assess future SVs. The time-aware split has proven to be crucial for ensuring accurate and reliable evaluations of SV-related prediction tasks [35, 44].

Results. Table 3 presents the performance comparisons between EAVA and baselines. LGBM achieves the best performance among the ML baselines. The two DL baselines (i.e., DeepCVA and

Table 3. The performance comparison between EAVA and baselines for SV assessment

CVSS Metric	Metrics	Method											
		RF	SVM	LR	KNN	XGB	LGBM	DeepCVA	proEVA	Llama	DeepSeek	GPT	EAVA
AV	F1	0.727	0.788	0.782	0.767	0.758	0.744	0.796	0.790	0.759	0.790	0.665	**0.856**
	MCC	0.269	0.391	0.375	0.327	0.308	0.257	0.451	0.406	0.473	0.531	0.422	**0.615**
AC	F1	**0.972**	**0.974**	0.970	**0.972**	**0.974**	**0.977**	**0.972**	**0.982**	**0.977**	0.951	0.967	**0.979**
	MCC	0.000	0.261	0.096	0.000	0.159	0.281	0.119	**0.497**	0.389	0.236	0.322	0.405
PR	F1	0.747	0.815	0.801	0.769	0.812	0.809	0.809	0.792	**0.831**	0.821	0.795	**0.830**
	MCC	0.207	0.401	0.352	0.300	0.395	0.389	0.399	0.321	**0.466**	0.423	0.386	**0.464**
UI	F1	0.742	0.768	0.789	0.613	0.764	0.770	0.788	0.813	0.659	0.705	0.769	**0.877**
	MCC	0.486	0.517	0.556	0.328	0.503	0.515	0.553	0.607	0.298	0.364	0.516	**0.736**
S	F1	0.939	0.948	0.953	0.924	0.962	0.958	0.950	0.949	0.779	0.826	0.918	**0.976**
	MCC	0.714	0.750	0.778	0.637	0.818	0.801	0.767	0.763	-0.056	0.325	0.651	**0.888**
C	F1	0.740	0.684	0.740	0.594	0.711	0.769	0.758	0.737	0.708	0.718	0.664	**0.809**
	MCC	0.572	0.482	0.567	0.332	0.529	0.618	0.599	0.567	0.551	0.549	0.529	**0.686**
I	F1	0.709	0.712	0.708	0.555	0.692	0.689	0.734	0.754	0.700	0.683	0.725	**0.802**
	MCC	0.512	0.520	0.514	0.263	0.494	0.490	0.566	0.590	0.517	0.481	0.556	**0.673**
A	F1	0.784	0.811	0.808	0.709	0.827	0.833	0.845	0.828	0.809	0.774	0.749	**0.866**
	MCC	0.507	0.553	0.548	0.298	0.596	0.611	0.643	0.600	0.577	0.524	0.499	**0.696**
Average	F1	0.795	0.812	0.819	0.738	0.812	0.819	0.832	0.831	0.778	0.783	0.781	**0.874**
	MCC	0.408	0.484	0.473	0.311	0.475	0.495	0.512	0.544	0.402	0.429	0.485	**0.646**
Severity	F1	0.468	0.540	0.546	0.450	0.556	0.572	0.549	0.587	0.572	0.492	0.405	**0.672**
	MCC	0.249	0.304	0.309	0.154	0.334	0.359	0.298	0.363	0.328	0.218	0.108	**0.490**

proEVA) slightly improve the performance of LGBM. For three LLM baselines, their performance is worse than the ML- and DL-based baselines by a large margin. This finding is in line with our preliminary study, which verifies the necessities of domain-specific knowledge in SV assessment.

Regarding the two overall measures, our approach yields the best performance across all evaluation metrics. For the average measure, EAVA surpasses the best-performing baseline (i.e., proEVA) by 5.3% and 18.7% in terms of weighted F1 and MCC, respectively. For the severity measure, the improvements are 14.4% and 35.2%, respectively. Notably, the improvements in MCC are considerably more pronounced than those in weighted F1, which can be attributed to the imbalanced class distributions across CVSS metrics [49]. While weighted F1 is primarily driven by performance on majority classes, MCC measures the global consistency between predictions and ground truth across all classes, and thus more effectively captures improvements in minority-class performance and inter-class discrimination. The substantial gains in MCC therefore indicate that EAVA excels at distinguishing between classes.

Regarding performance on individual CVSS metrics, EAVA consistently outperforms proEVA across all metrics except *Attack Complexity*. The most substantial improvements are observed on *Confidentiality*, *Attack Vector*, *User Interaction*, and *Integrity*, with F1 gains of 9.7%, 8.3%, 7.9%, and 6.5%, respectively. In contrast, improvements on the remaining metrics, namely *Privileges Required*, *Availability*, and *Scope*, are less pronounced, at 4.7%, 4.7%, and 2.9% in F1, respectively. This is primarily because proEVA already achieves strong performance on these metrics. For example, proEVA attains a high F1 of 0.949 on the *Scope* metric, while EAVA further improves it to 0.976. Overall, these results confirm the consistent advantages of EAVA over existing approaches across CVSS metrics.

RQ1: EAVA *significantly outperforms baselines in SV assessment, consistently improving performance across CVSS metrics, with especially large gains on those metrics the baselines are weaker.*

Table 4. The results for assessing the effectiveness of designs for training a dedicated assessment LLM

Metrics	Approach	AV	AC	PR	UI	S	C	I	A	Average	Severity
F1	**w/o finetune**	0.795	0.933	**0.830**	0.687	0.805	0.760	0.713	0.832	0.794	0.584
	w/o reason	0.728	**0.972**	0.807	0.830	**0.975**	0.786	0.768	0.849	0.839	0.566
	w/o RL	0.797	**0.975**	**0.822**	0.821	**0.971**	0.784	0.748	**0.866**	0.848	0.617
	EAVA	**0.856**	**0.979**	**0.830**	**0.877**	**0.976**	**0.809**	**0.802**	**0.866**	**0.874**	**0.672**
MCC	**w/o finetune**	0.541	0.250	0.448	0.403	0.074	0.620	0.538	0.578	0.431	0.351
	w/o reason	0.454	0.000	0.388	0.636	**0.881**	0.657	0.623	0.659	0.537	0.337
	w/o RL	0.490	0.226	0.423	0.614	0.865	0.646	0.584	**0.704**	0.569	0.397
	EAVA	**0.615**	**0.405**	**0.464**	**0.736**	**0.888**	**0.686**	**0.673**	**0.696**	**0.646**	**0.490**

6.2 RQ2. Effectiveness of Designs for Training a Dedicated Assessment LLM

Method. We further verify the effectiveness of key designs for finetuning LLM to effectively inject assessment-specific knowledge. We compare the performance of EAVA with several variants, each lacking one of the key designs:

- The *w/o finetune* variant is designed to examine the necessity of finetuning when building a specialized assessment LLM. We remove fine-tuning, which embeds historical SV data into the model's internal knowledge (i.e., its parameters). Instead, we use historical SV data (i.e., the training and validation sets) to construct an external knowledge base, which the vanilla LLM accesses through a retrieval-augmented generation (RAG) process. Note that the historical SV data is enriched with evidence prepared using the methods described in Section 4.2. Our initial tries indicate that including evidence, in addition to the label metric value, in the demonstration examples significantly improves performance. We adopt the same *few-shot* prompt setting as in our preliminary study, which has been proven to be effective. Specifically, we retrieve the top 10 historical SVs using a hybrid retriever (see Section 4.3) and select one example per candidate metric value to construct the few-shot demonstrations. We use Llama-3.3-70b for this variant instead of Llama-3.1-8b used to build the dedicated assessment LLM, which offers stronger off-the-shelf ability.
- The *w/o reason* variant is designed to examine whether explicitly generating a reasoning process improves the assessment performance. For this variant, we finetune the model to directly predict the metric value without first producing an explicit reasoning trajectory.
- The *w/o RL* variant is designed to assess the effectiveness of applying reinforcement learning (RL) on top of conventional SFT in enhancing the model's reasoning capability. Specifically, we report the performance of using SFT alone, without the subsequent GRPO training (see Section 4.2).

Results. Table 4 presents the performance comparisons between EAVA and variants. Regarding the average measure, EAVA outperforms the *w/o finetuning* variant by 10.1%, and 49.6% in terms of weighted F1 and MCC, respectively. Regarding the severity measure, the improvements are 15.1% and 39.6%, respectively. Besides, EAVA consistently outperforms the *w/o finetuning* variant on every specific CVSS metric. The most significant enhancements are observed on *User Interaction* and *Scope*, with improvements of 27.6% and 21.3% in terms of weighted F1, respectively. For the assessment of these CVSS metrics, the domain knowledge is the most demanding. These results verify the necessity of injecting assessment-specific knowledge through finetuning (i.e., a dense solution), which is more efficient than RAG (i.e., a sparse solution). Compared with the *w/o reason* variant, EAVA consistently achieves better performance on both overall measures and all individual CVSS metrics. Specifically, on the average measure, EAVA improves weighted F1 and MCC by 4.2% and 20.1%, respectively. These results suggest that explicitly learning the reasoning process enables LLMs to more accurately determine the correct metric values. Furthermore, EAVA also significantly

Table 5. The results for assessing the effectiveness of designs for enriching SV information

Metrics	Approach	AV	AC	PR	UI	S	C	I	A	Average	Severity
F1	w/o code	0.826	**0.979**	**0.838**	0.854	**0.973**	0.800	0.791	0.797	0.857	0.634
	w/o screen	0.833	**0.978**	0.813	0.865	**0.973**	**0.821**	**0.805**	0.854	**0.868**	0.658
	w/o repo	0.829	**0.979**	0.822	0.849	**0.975**	0.802	**0.799**	**0.863**	**0.865**	0.646
	EAVA	**0.856**	**0.979**	**0.830**	**0.877**	**0.976**	0.809	**0.802**	**0.866**	**0.874**	**0.672**
MCC	w/o code	0.509	**0.405**	**0.484**	0.684	0.873	0.672	0.652	0.564	0.605	0.433
	w/o screen	0.546	0.328	0.399	0.710	0.873	**0.704**	**0.677**	0.665	0.613	0.470
	w/o repo	0.517	**0.405**	0.431	0.678	**0.881**	0.676	**0.668**	**0.693**	0.619	0.451
	EAVA	**0.615**	**0.405**	0.464	**0.736**	**0.888**	0.686	**0.673**	**0.696**	**0.646**	**0.490**

outperforms the *w/o RL* variant across both overall measures and all individual CVSS metrics. For the average measure, the gains reach 3.1% in weighted F1 and 13.5% in MCC, demonstrating that applying RL on top of conventional SFT further enhances the model's intrinsic reasoning capability. In addition, EAVA exhibits similar performance gains over both the *w/o reason* and *w/o RL* variants across individual CVSS metrics. The largest improvements are observed for *Attack Vector*, *User Interaction*, and *Integrity*, suggesting that effective reasoning over surface-level SV information is particularly critical for accurately assessing these metrics.

RQ2: *Injecting assessment-specific knowledge through fine-tuning is more effective than a RAG-based solution. Explicitly learning the reasoning process for determining metric values further improves assessment performance. Applying RL on top of conventional SFT enhances the model's intrinsic reasoning capability.*

6.3 RQ3. Effectiveness of Designs for Enriching SV Information

Method. We further verify the effectiveness of the designs for enriching SV information by processing rich text content in SVRs and incorporating information of the affected repository. Specifically, we compare the performance of EAVA with three variants, i.e., *w/o code*, *w/o screen*, and *w/o repo*, each of which removes one type of enriched SV information.

Results. Table 5 presents the performance comparisons between EAVA and the variants. EAVA achieves the best overall performance. Preprocessing embedded code snippets yields the most significant benefits among the three types of enriched SV information. For the severity measure, this results in improvements of 8.5% and 13.4% in terms of weighted F1 and MCC, respectively. This is likely because long and noisy code snippets can distract LLMs from the key information needed for accurate assessment. Incorporating screenshot information improves the severity measure by 2.2% and 4.3% in terms of weighted F1 and MCC, respectively. For individual CVSS metrics, incorporating screenshot information consistently improves performance across all metrics except *Confidentiality*, with the largest gains observed for *Attack Vector* and *Privileges Required*, achieving MCC improvements of 12.7% and 16.3%, respectively. Incorporating information about the vulnerable project yields consistent improvements on both overall measures and all individual CVSS metrics, achieving gains of 4.4% in weighted F1 and 8.9% in MCC for the severity measure. Collectively, these results confirm the effectiveness of processing rich text content in SVRs and integrating vulnerable project information for accurate SV assessment.

RQ3: *Incorporating rich text content from SVRs and information about the vulnerable project enhances assessment performance.*

6.4 RQ4. Necessities and Effectiveness of the Evidence Provided by EAVA in Practice

Method. In the previous RQs, we demonstrate that EAVA achieves accurate SV assessment. We further examine the necessity and effectiveness of providing supporting evidence (i.e., reasoning processes) for the assigned metric values. As discussed in Section 3.1, prior studies focus primarily on assessment accuracy while overlooking the need to explain how assessment results are derived. We therefore conduct a user study to evaluate how the supporting evidence provided by EAVA assists analysts in determining assessment results.

Tasks: We randomly select 40 SVs from our test set, resulting in a total of 320 metric-wise assessment tasks. For each SV, we provide participants with 1) the SV report and relevant CVSS standard, 2) the value assigned and the analysis generated by EAVA, 3) the information of similar historical SVs, including CVE-ID, report, and the analysis generated by EAVA. We then ask participants to answer the following questions for each of the eight CVSS metrics.

1) Choose the appropriate metric value.
2) Rate (using 5-point Likert scale) the usefulness of the provided evidence.
3) Rate the factuality (i.e., the evidence is grounded in the given SV information), relevance (i.e., the evidence directly supports the assigned metric value), and completeness (i.e., the evidence covers key metric-relevant details from the given SV information) of the provided evidence.
4) Rate the usefulness of the provided historical SV information in determining the metric value. Participants can skip Q4 if they think the information of the SV itself is enough to make decision.

Participants: We invite 8 security practitioners as participants. All of them have three to five years of experience in the software security domain and are familiar with the CVSS standard.

Results. Among the 284 cases where EAVA predicts the correct metric value, participants acknowledge the usefulness of the provided evidence (rating 4/5) in 275 (96.8%) cases. For these cases, participants generally rate the provided evidence with high factuality, relevance, and completeness (average rating all above 4.6). We further investigate 54 cases where participants think the provided evidence is less useful (rating$\leq$4). We observe that the ratings for factuality and completeness remain high, but downgrade on relevance. Additionally, we also observe a much higher ratio of historical SV information being examined. We interview participants and confirm that, in these cases, they feel less confident making decisions based solely on the information of the SV itself, prompting them to further examine the provided historical SV information. Regarding the usefulness of the historical SV information provided by EAVA, among the 89 cases examined by participants, 78 (87.6%) is rated with 4/5. For the left cases where the retrieved historical SVs are less useful, we find that this is mostly because the query SV is closely tied to the project-specific attributes.

Among the 36 cases where EAVA makes the wrong prediction, we find that participants are unlikely to be mislead. Participants select the correct metric value in 12 cases. We find that, in these cases, the evidence quality is rated with the same high factuality, but lower relevance and completeness compared to cases where EAVA makes correct predictions. This indicates that the evidence given by EAVA remains fact-based, but can be irrelevant or miss the key information related to the correct metric value. Through interviews with participants and a manual check of evidence in these cases, we find that EAVA accurately analyzes the SV but either misinterprets the CVSS criteria or provides general evidence that does not lead to any specific metric values. We further investigate the left 24 cases where participants fail to select the correct metric value. We find that in 18 cases, the publicly available SV information is insufficient to determine a unique CVSS metric value, or the value assigned by NVD may be inappropriate [16, 29]. For example, CVE-2023-39138 [30] is a path traversal SV caused by insufficient validation of whether a symlink resides within the directory during ZIP file extraction. The impact of this SV mostly concerns *Confidentiality* and *Integrity*, as attackers can assess and modify sensitive information via crafted

symlinks. The SV report does not mention any impact on *Availability*. However, NVD assigns *High* for *Availability* metric, while Synk [15] (a commercial SV database) and EAVA assign *None*.

RQ4: *Evidence provided by* EAVA *can effectively help analysts in determine the assessment results. Participants are unlikely to be misled by* EAVA.

7 DISCUSSION

7.1 Validity of Analyzing Screenshots and Code Snippets

We randomly sample 50 screenshots and 50 code snippets and engage two annotators to independently assess the faithfulness of LLM's analysis using 5-point likert scale. For screenshots, the two annotators provide identical ratings in 48 cases and the remaining two cases are discussed to reach a consensus. 48 screenshots receive the highest score of 5. For the remaining two cases: ❶ The LLM includes redundant information from previously analyzed screenshots rather than focusing on the current one (rated 4). ❷ LLM mistakenly identifies the calculator application launched via a remote-code-execution vulnerability as a hex editor (rated 3). For code snippets, both annotators consistently assign a score of 5 across all 50 cases. These results indicates that our proposed LLM agent (see Section 4.1) can faithfully analyze the rich-text content embedded in SVRs.

7.2 Threats to Validity

Internal Validity We use the CVSS values provided by NVD as labels. However, the values assigned by NVD may be inaccurate [16, 29]. This is primarily because the SV assessment process is highly dependent on the information available to the analyst, the analyst's familiarity with the SV context (e.g., the affected projects), and their expertise and experience [28, 31, 63]. It is further demonstrated by the fact that different SV databases may assign varying CVSS values to the same SV. For example, NVD and Synk assign different CVSS scores to CVE-2023-29578 (i.e., 8.8 and 5.5). However, NVD CVSS scores are widely recognized by both industry and government as the standard for software vulnerability assessment [3, 14, 26].

External Validity We build our SVR dataset using only the SV data from NVD and report data from GitHub, which may not represent all SV reports. Future works should investigate the generalizability of EAVA regarding different SVR proxies, i.e., a different SV database (e.g., Synk) or a different source of SV reports (e.g., Bugzilla). Another threat is that we only consider CVSS as the standard for SV assessment. There are other assessment standards, e.g., Exploit Prediction Scoring System (EPSS) is a dedicated standard for estimating the likelihood of SVs being exploited in the wild [19].

8 CONCLUSION

In this paper, we propose EAVA, a novel framework that effectively leverages LLMs to perform SV assessment while providing supporting evidence. Through extensive evaluations with mainstream LLMs, we demonstrate that off-the-shelf LLMs struggle to produce accurate assessments in the absence of domain-specific knowledge. At the core of EAVA is a dedicated assessment LLM, built through large-scale trajectory annotation and a two-stage fine-tuning paradigm that combines SFT and RL to inject assessment-specific knowledge. EAVA first employs specialized LLM agents to process rich text content in SVRs and incorporate information about vulnerable projects. The dedicated assessment LLM then reasons over the enriched SV information to determine assessment results. Finally, similar historical SVs are retrieved as supplementary evidence to support assessment decisions. Experimental results show that EAVA significantly outperforms baselines and validate the effectiveness of our key design choices. Moreover, a user study confirms both the necessity and usefulness of the evidence provided by EAVA in facilitating SV assessment.

9 DATA AVAILABILITY

The replication package of our work is publicly available at [27].

REFERENCES

[1] 2021. CVE-2021-44228. https://nvd.nist.gov/vuln/detail/CVE-2021-44228.
[2] 2022. Vuldb. https://vuldb.com/?doc.sources.
[3] 2023. BOD 19-02: Vulnerability Remediation Requirements for Internet-Accessible Systems | CISA. https://www.cisa.gov/news-events/directives/bod-19-02-vulnerability-remediation-requirements-internet-accessible-systems.
[4] 2023. CVSS v3.1 Examples. https://www.first.org/cvss/v3-1/examples.
[5] 2023. CVSS v3.1 Specification Document. https://www.first.org/cvss/v3.1/specification-document.
[6] 2023. ISO/IEC 30111:2019 Information technology Security techniques Vulnerability handling processes. https://www.iso.org/standard/69725.html.
[7] 2023. National Vulnerability Database. https://nvd.nist.gov/.
[8] 2023. OSV Vulnerability Database. https://osv.dev/.
[9] 2023. Using SLAs for Better Vulnerability Management & Remediation: Improving Developer's Workflow - Phoenix Security. https://phoenix.security/using-slas-for-better-vulnerability-management-remediation-improving-developers-workflow/.
[10] 2024. Llama 3.3 | Model Cards and Prompt formats. https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/.
[11] 2024. Vulnerability Management SLAs Guide. https://hostedscan.com/blog/vulnerability-management-slas-guide.
[12] 2025. Base64 - Wikipedia. https://en.wikipedia.org/wiki/Base64.
[13] 2025. Common Vulnerability Scoring System. https://www.first.org/cvss/.
[14] 2025. CSP Vulnerability Scanning Requirements. https://www.fedramp.gov/assets/resources/documents/CSP_Vulnerability_Scanning_Requirements.pdf.
[15] 2025. CVE-2023-39138 | Synk. https://security.snyk.io/vuln/SNYK-SWIFT-WEICHSELZIPFOUNDATION-5876635.
[16] 2025. CVE and NVD CVSS Score Enhancements. https://updates.snyk.io/cve-and-nvd-cvss-score-enhancements-upcoming-data-changes-278498.
[17] 2025. CVSS v3.1 Examples. https://www.first.org/cvss/v3-1/examples.
[18] 2025. CWE - 2024 CWE Top 25 Most Dangerous Software Weaknesses. https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html.
[19] 2025. Exploit Prediction Scoring System (EPSS). https://www.first.org/epss/.
[20] 2025. glazedlists/Issue-709. https://github.com/glazedlists/glazedlists/issues/709.
[21] 2025. gpac/Issue-2874. https://github.com/gpac/gpac/issues/2874.
[22] 2025. LLM Leaderboard 2025 - Verified AI Rankings. https://llm-stats.com/.
[23] 2025. meta-llama/Llama-3.1-8B | Hugging face. https://huggingface.co/meta-llama/Llama-3.1-8B.
[24] 2025. Model - OpenAI API. https://platform.openai.com/docs/models/gpt-4.1.
[25] 2025. Octoverse: AI leads Python to top language as the number of global developers surges - The GitHub Blog. https://github.blog/news-insights/octoverse/octoverse-2024/#the-most-popular-programming-languages.
[26] 2025. PCI DSS v3.2.1 Quick Reference Guide. https://listings.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf.
[27] 2025. Replication Package. https://figshare.com/s/b0ac80383971ae7ed8ea.
[28] 2025. Scoring Security Vulnerabilities: 101 Introducing CVSS for CVE. https://snyk.io/blog/scoring-security-vulnerabilities-101-introducing-cvss-for-cve/.
[29] 2025. Why-does-Snyk-have-a-different-CVSS-to-NVD-for-specific-CVEs. https://support.snyk.io/s/article/Why-does-Snyk-have-a-different-CVSS-to-NVD-for-specific-CVEs.
[30] 2025. ZIPFoundation/Issue-282. https://github.com/weichsel/ZIPFoundation/issues/282.
[31] Luca Allodi, Sebastian Banescu, Henning Femmer, and Kristian Beckers. 2018. Identifying relevant information cues for vulnerability assessment using CVSS. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 119–126.
[32] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023* (2023).
[33] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. [n. d.]. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. In *Forty-second International Conference on Machine Learning*.

[34] Nesara Dissanayake, Asangi Jayatilaka, Mansooreh Zahedi, and Muhammad Ali Babar. 2022. An Empirical Study of Automation in Software Security Patch Management. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

[35] Davide Falessi, Jacky Huang, Likhita Narayana, Jennifer Fong Thai, and Burak Turhan. 2020. On the need of preserving order of data when validating within-project defect classifiers. *Empirical Software Engineering* 25 (2020), 4805–4830.

[36] Andrew Feutrill, Dinesha Ranathunga, Yuval Yarom, and Matthew Roughan. 2018. The effect of common vulnerability scoring system metrics on vulnerability exploit delay. In *2018 Sixth International Symposium on Computing and Networking (CANDAR)*. IEEE, 1–10.

[37] Park Foreman. 2019. *Vulnerability management*. CRC Press.

[38] Jan Gorodkin. 2004. Comparing two K-category assignments by a K-category correlation coefficient. *Computational biology and chemistry* 28, 5-6 (2004), 367–374.

[39] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Mingchuan Zhang, Y.K. Li, Yu Wu, Daya Guo, Wenfeng Liang, et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in Large Language Models via Reinforcement Learning. *Nature* 645, 8081 (2025), 299–306. https://doi.org/10.1038/s41586-025-09422-z

[40] Zhuobing Han, Xiaohong Li, Zhenchang Xing, Hongtao Liu, and Zhiyong Feng. 2017. Learning to predict severity of software vulnerability using only vulnerability description. In *2017 IEEE International conference on software maintenance and evolution (ICSME)*. IEEE, 125–136.

[41] Emanuele Iannone, Giulia Sellitto, Emanuele Iaccarino, Filomena Ferrucci, Andrea De Lucia, and Fabio Palomba. 2024. Early and Realistic Exploitability Prediction of Just-Disclosed Software Vulnerabilities: How Reliable Can It Be? *ACM Transactions on Software Engineering and Methodology* 33, 6 (2024), 1–41.

[42] Nasif Imtiaz, Aniqa Khanom, and Laurie Williams. 2022. Open or Sneaky? Fast or Slow? Light or Heavy?: Investigating Security Releases of Open Source Packages. *IEEE Transactions on Software Engineering* (2022).

[43] Public Key Infrastructure and Token Protection Profile. 2002. Common criteria for information technology security evaluation. *National Security Agency* (2002).

[44] Matthieu Jimenez, Renaud Rwemalika, Mike Papadakis, Federica Sarro, Yves Le Traon, and Mark Harman. 2019. The importance of accounting for real-world labelling when predicting software vulnerabilities. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 695–705.

[45] Patrick Kwaku Kudjo, Jinfu Chen, Minmin Zhou, Solomon Mensah, and Rubing Huang. 2019. Improving the Accuracy of Vulnerability Report Classification Using Term Frequency-Inverse Gravity Moment. In *Proceedings of 19th International Conference on Software Quality, Reliability and Security (QRS)*. 248–259.

[46] Triet HM Le, Huaming Chen, and M Ali Babar. 2022. A survey on data-driven software vulnerability assessment and prioritization. *Comput. Surveys* 55, 5 (2022), 1–39.

[47] Triet Huynh Minh Le and Muhammad Ali Babar. 2024. Mitigating data imbalance for software vulnerability assessment: Does data augmentation help?. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 119–130.

[48] Triet Huynh Minh Le and M Ali Babar. 2022. 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*. 621–633.

[49] Triet Huynh Minh Le, David Hin, Roland Croft, and M Ali Babar. 2021. Deepcva: Automated commit-level vulnerability assessment with deep multi-task learning. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 717–729.

[50] Triet Huynh Minh Le, Bushra Sabir, and Muhammad Ali Babar. 2019. Automated software vulnerability assessment with concept drift. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 371–382.

[51] Xiangwei Li, Xiaoning Ren, Yinxing Xue, Zhenchang Xing, and Jiamou Sun. 2023. Prediction of vulnerability characteristics based on vulnerability description and prompt learning. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 604–615.

[52] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[53] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics* 9 (2021), 329–345.

[54] Qiheng Mao, Zhenhao Li, Xing Hu, Kui Liu, Xin Xia, and Jianling Sun. 2025. Towards Explainable Vulnerability Detection With Large Language Models. *IEEE Transactions on Software Engineering* 51, 10 (2025), 2957–2971. https://doi.org/10.1109/TSE.2025.3605442

[55] Shengyi Pan, Lingfeng Bao, Jiayuan Zhou, Xing Hu, Xin Xia, and Shanping Li. 2024. Towards More Practical Automation of Vulnerability Assessment. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[56] Shengyi Pan, Jiayuan Zhou, Filipe Roseiro Cogo, Xin Xia, Lingfeng Bao, Xing Hu, Shanping Li, and Ahmed E Hassan. 2022. Automated unearthing of dangerous issue reports. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 834–846.

[57] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction Tuning with GPT-4. *arXiv preprint arXiv:2304.03277* (2023). Available at https://arxiv.org/abs/2304.03277.

[58] Serena E. Ponta, Henrik Plate, Antonino Sabetta, Michele Bezzi, and Cédric Dangremont. 2019. A Manually-Curated Dataset of Fixes to Vulnerabilities of Open-Source Software. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR '19)*. 383–387.

[59] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI Technical Report* (2018). https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf Preprint; introduced the GPT-1 model trained with a next-token prediction (causal LM) objective.

[60] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 https://arxiv.org/abs/1707.06347

[61] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* (2024). https://arxiv.org/abs/2402.03300 Introduces GRPO—Group Relative Policy Optimization, a key RL algorithm used in DeepSeek models.

[62] Georgios Spanos and Lefteris Angelis. 2018. A multi-target approach to estimate software vulnerability characteristics and severity scores. *Journal of Systems and Software* 146 (2018), 152–166.

[63] Jonathan Spring, Eric Hatleback, A Manion, and D Shic. 2018. Towards improving CVSS. *SEI, CMU, Tech. Rep* (2018).

[64] Gary Stoneburner, Alice Goguen, Alexis Feringa, et al. 2002. Risk management guide for information technology systems. *Nist special publication* 800, 30 (2002), 800–30.

[65] Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text Classification via Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 8990–9005.

[66] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[67] Yongda Yu, Guoping Rong, Haifeng Shen, He Zhang, Dong Shao, Min Wang, Zhao Wei, Yong Xu, and Juhong Wang. 2025. Fine-Tuning Large Language Models to Improve Accuracy and Comprehensibility of Automated Code Review. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 34, 1 (2025), 14:1–14:26. https://doi.org/10.1145/3695993

[68] Wen Zhang, Yang Feng, and Qun Liu. 2021. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 4790–4794.

[69] Xin Zhou, Sicong Cao, Xiaobing Sun, and David Lo. 2024. Large language model for vulnerability detection and repair: Literature review and the road ahead. *ACM Transactions on Software Engineering and Methodology* (2024).