

Domain-constrained Synthesis of Inconsistent Key Aspects in Textual Vulnerability Descriptions

Linyi Han^{1†}, Shidong Pan^{2†}, Zhenchang Xing²,
Sofonias Yitagesu¹, Xiaowang Zhang^{1*}, Zhiyong Feng¹,
Jiamou Sun², Qing Huang³

¹Tianjin University, Tianjin, China.

²CSIRO's Data61, Canberra, ACT, Australia.

³Jiangxi Normal University, Nanchang, Jiangxi, China.

*Corresponding author(s). E-mail(s): xiaowangzhang@tju.edu.cn;
Contributing authors: hanly2@tju.edu.cn; Shidong.Pan@data61.csiro.au;
Zhenchang.Xing@data61.csiro.au; sofoniasyitagesu@yahoo.com;
zyfeng@tju.edu.cn; Frank.Sun@data61.csiro.au; qh@jxnu.edu.cn;

†These authors contributed equally to this work.

Abstract

Textual Vulnerability Descriptions (TVDs) are crucial for security analysts to understand and address software vulnerabilities. However, the key aspect inconsistencies in TVDs from different repositories pose challenges for achieving a comprehensive understanding of vulnerabilities. Existing approaches aim to mitigate inconsistencies by aligning TVDs with external knowledge bases, but they often discard valuable information and fail to synthesize comprehensive representations. In this paper, we propose a domain-constrained LLM-based synthesis framework for unifying key aspects of TVDs. Our framework consists of three stages: 1) Extraction, guided by rule-based templates to ensure all critical details are captured; 2) Self-evaluation, using domain-specific anchor words to assess semantic variability across sources; and 3) Fusion, leveraging information entropy to reconcile inconsistencies and prioritize relevant details. This framework improves synthesis performance, increasing the F1 score for key aspect augmentation from 0.82 to 0.87, while enhancing comprehension and efficiency by over 30%. We further develop Digest Labels, a practical tool for visualizing TVDs, which human evaluations show significantly boosts usability.

Keywords: Vulnerability Analysis, Inconsistent Key Aspect, Domain-Constrained Synthesis, Digest Labels System, Vulnerability Repositories

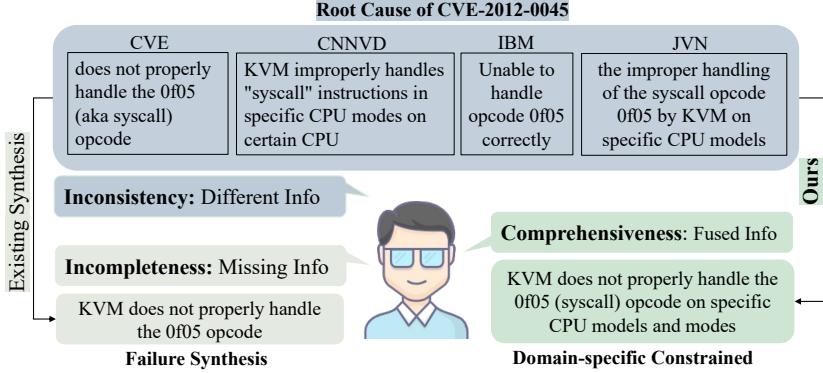


Fig. 1 Different repositories provide varying details. If synthesis loses details, the purpose of achieving comprehensive vulnerability understanding is undermined.

1 Introduction

Software vulnerabilities pose severe risks to modern systems, as they can be exploited by attackers to compromise confidentiality, integrity, and availability. To facilitate mitigation, various vulnerability databases have been established, such as the Common Vulnerabilities and Exposures (CVE) [1], National Vulnerability Database (NVD) [2], and IBM X-Force [3]. These repositories provide *Textual Vulnerability Descriptions* (TVDs), which contain crucial information for security analysts to understand, assess, and repair vulnerabilities. TVDs are expected to capture essential **key aspects** of a vulnerability, including the *Vulnerability Type*, how an attack can be executed (*Attack Vector*), the type of attacker (*Attacker Type*), the *Root Cause*, and its potential *Impact*.

However, inconsistencies often exist between different repositories for the same vulnerability. These inconsistencies may arise because one TVD provides only a partial description, while another emphasizes specific contextual details. For instance, as illustrated in Figure 1, the *Root Cause* of CVE-2012-0045 is described in different ways: CVE provides a basic description (“does not properly handle the 0f05 opcode”), whereas CNNVD offers more detailed information (“KVM improperly handles syscall instructions in specific CPU modes on certain CPU models”). Such differences reflect not errors but multiple perspectives of the same issue. Drawing from the *Blind Men and an Elephant* theory, we argue that instead of discarding inconsistent information, synergistically integrating these complementary perspectives leads to a more comprehensive understanding of vulnerabilities.

These inconsistencies present a significant challenge for analysts. Given the large volume of vulnerabilities and the technical complexity of TVDs, analysts must spend additional effort to reconcile descriptions from multiple repositories. Manually synthesizing this information is time-consuming and error-prone. Recent studies attempt to address the issue by comparing TVDs with external knowledge bases to filter out inconsistencies. Yet, this approach often treats differences as errors and discards details

that may in fact provide valuable complementary information. Consequently, important contextual details are lost, which hinders the ultimate goal of comprehensive vulnerability understanding.

Large Language Models (LLM) offer promising opportunities for synthesizing TVDs, as they are capable of semantic reasoning and integrating diverse inputs. However, existing LLM-based synthesis methods are primarily guided by soft semantic instructions, such as “synthesize vulnerability descriptions with as much detail as possible”. These instructions provide only high-level direction and lack the precision to enforce domain-specific boundaries. As a result, generated descriptions may remain vague and overlook critical information. For example, as shown in Figure 1, when synthesizing the *Root Cause* of CVE-2012-0045, an unconstrained LLM may output “KVM fails to handle the 0f05 syscall opcode”, missing key details such as “specific CPU models” or “modes”.

To overcome these challenges, this paper proposes a domain-constrained LLM-based synthesis approach for TVDs. By introducing domain-specific conditions that restrict the model’s searching space, our approach preserves diverse expressions while ensuring that critical technical details are retained. Based on this framework, we further develop a practical tool, named **Digest Labels (DLs)**, which standardizes the synthesis of TVDs from multiple repositories. In real-world evaluations, DLs significantly improve analysts’ comprehension and efficiency, achieving a 31.3% increase in comprehensiveness and a 33.5% increase in efficiency.

Overall, this work makes the following contributions:

- **Problem Identification:** We reveal that inconsistencies across vulnerability repositories are widespread and stem from complementary perspectives rather than simple errors.
- **Domain-Constrained Synthesis:** We introduce a novel synthesis approach that integrates LLM with domain-specific constraints to preserve completeness and accuracy of TVDs.
- **Digest Labels (DLs):** We design and implement the first nutrition-label inspired system for TVDs, demonstrating its usability and effectiveness through empirical experiments and human evaluation.

The remainder of this paper is organized as follows: Section 2 provides the motivation for synthesizing key aspects. Section 3 introduces the design of Digest Labels (DLs) and explains how labels are structured to represent synthesized information. Section 4 then presents our domain-constrained synthesis approach. Section 5 reports the experimental evaluation, while Section 6 discusses implications and limitations. Section 7 reviews related work, and Section 8 concludes the paper.

2 Motivation for Synthesis of Key Aspect in TVDs

In this motivation study, we systematically retrospect the challenges associated with CVE TVD from three perspectives.

2.1 Industry Standards

Vulnerability management is one of the constitutional problem of software engineering industry. Many authoritative organizations launch a set of industry standards of CVE, including the TVD standardization, with the goal of improving overall quality. Those standards provide the best of practices to all practitioners as reference and we highlight the content related to TVD standardization:

- **[NIST]** The National Institute of Standards and Technology (NIST), under the U.S. Department of Commerce, offers essential standards for vulnerability management. These include naming schemas (NIST SP 800-51), formalization (NIST SP 800-231), and communication guidelines (NIST SP 800-216). The Security Content Automation Protocol (SCAP) provides a standardized way to communicate vulnerability information. As stated by CISA: “*a suite of specifications for standardizing the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans*” [4].
- **[ISO]** The International Organization for Standardization (ISO) released BS ISO/IEC 29147:2018, which provides a standard format for reporting software name and version of vulnerabilities. However, no formal guidelines exist for key aspects in TVDs, making them harder to standardize due to their human-originated nature. As stated in the standard, “*a vulnerability is generally a behaviour or set of conditions that allows the violation of an explicit...*” [5]. Our work enhances these specifications by addressing the gaps in formatting key aspects.

Even though these standards set expectations for online CVE TVDs, we have observed prevalent issues of missing information and inconsistency, specifically discussed in the following section.

2.2 Definition of Key Aspects

We clarify what we mean by *key aspects* of a TVD. Drawing from prior work [6–9] and industry standards such as CVE [1] and IBM X-Force [3], we define the following key aspects:

- **Vulnerability Type:** the category or nature of the vulnerability (e.g., buffer overflow, SQL injection).
- **Attack Vector:** how an attack can be executed (e.g., remote network access, local execution).
- **Attacker Type:** the assumed capabilities or privileges of the attacker (e.g., authenticated user, remote adversary).
- **Root Cause:** the underlying technical reason leading to the vulnerability (e.g., improper opcode handling, incorrect input validation).
- **Impact:** the consequence of successful exploitation (e.g., privilege escalation, denial of service, data leakage).

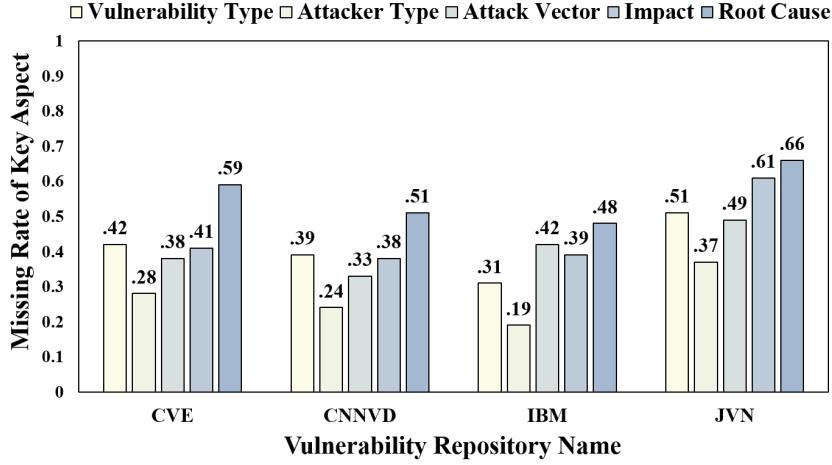


Fig. 2 Missing rates on different vulnerability repositories.

These aspects represent the core information units that practitioners rely on to understand and mitigate vulnerabilities.

2.3 Pilot Experiments

In addition to the takeaways from industry standards, we also conducted an empirical experiment to further explore the scope and depth of the aforementioned problems, revealing the extent to which reality falls short of industry standard expectations.

Dataset. We first widely collect four popular web-based vulnerability repositories that are indexed by the CVE-ID: CVE (English), IBM (English), JVN (Japanese), and CNNVD (Chinese). We then extract key aspects using the method described in Section 4.1, which utilizes LLM with regularization templates derived from human guidelines. These templates, summarized from previous research [10–12], replace the need for fixed examples in in-context learning, guiding the LLM to extract key aspects effectively. This approach helps us process TVDs from these four repositories spanning from 1999 to 2023.

Information Missing. The missing rate for each key aspect is calculated by counting the occurrences and dividing by the total number of TVDs, then subtracting from 100%. As shown in Figure 2, CVE and JVN have the highest missing rates, especially for *Root Cause* (0.59 and 0.66). CNNVD shows the lowest missing rates, while IBM is moderate. These high missing rates, particularly for crucial aspects like *Root Cause*, demonstrate a severe lack of essential information across repositories. This gap can hinder security engineers’ ability to fully understand vulnerabilities, resulting in delayed or ineffective responses. [13–15] Standardized TVDs are needed to mitigate this issue.

Information Inconsistency. Previous studies have thoroughly examined the inconsistencies in key aspects like *Affected Product*, *Component*, and *Version* [7–9, 16, 17]. These inconsistencies pose a significant challenge to security engineers as

Table 1 Missing and inconsistency rates for different key aspects.

Methods	Vulnerability Type	Attack Vector	Attacker Type	Impact	Root Cause
I. R.	0.67	0.42	0.36	0.48	0.38
M. M.	0.32	0.20	0.34	0.38	0.49
M. R.	0.15	0.09	0.20	0.17	0.23

“I. R.” refers to “Inconsistency Rate”, “M. M.” represents “Min Missing Rate in Single Repository”, and “M. R.” indicates “Missing Rate”.

they lead to gaps in vulnerability understanding, miscommunication, and even incorrect threat assessments, potentially delaying critical remediation efforts. The severity of the issue is illustrated by the inconsistency rates ranging from 0.36 to 0.67 across different repositories (Table1), indicating that many key aspects, such as *Vulnerability Type* and *Attack Vector*, often differ between sources.

Moreover, merging data from multiple sources substantially reduces the missing rates for key aspects, enhancing the completeness of the information. For instance, before merging, the missing rate for *Root Cause* in single repositories ranged from 0.49 (CNNVD) to 0.66 (JVN). After merging, the missing rate dropped to 0.23, a reduction of more than 50%. Similarly, for *Vulnerability Type*, the missing rate decreased from 0.32 to 0.15, and for *Attack Vector*, from 0.20 to 0.09. This demonstrates that combining information from different repositories not only resolves a large portion of the missing information but also improves the overall reliability and accuracy of the TVDs. The substantial decrease in missing rates highlights the value of data consolidation in delivering more comprehensive and consistent vulnerability data.

Despite industry standards, current TVDs lack standardization, leading to high missing rates and inconsistencies across information sources. Guo et al. [6, 13, 15] highlights that such gaps can lead to increased error rates in vulnerability assessments, inefficiencies in remediation efforts, and potential misinterpretation of security risks. This increases labor costs, delays remediation, and raises the risk of misinterpretation. Therefore, we propose Digest Labels (DLs) that consolidates key aspects from multiple sources, facilitating practitioners in CVE-related downstream tasks.

3 Design of TVD Digest Labels

In the formative study, we discuss the long-standing challenges associated with CVE TVDs, such as the high rates of missing and inconsistent information. Current solutions often fall short in addressing these issues effectively. Our goal is to provide a more complete and standardized representation of vulnerability information for software engineers. Inspired by the concept of nutrition labels and their widespread application in various domains, we aim to adopt this approach to enhance the clarity and usability of TVD information. While our formative study highlights the issues of missing and inconsistent information, our label design is motivated by the need to streamline the information reception process for security engineers. The *Digest Labels* for CVE TVDs are designed to consolidate key aspects from multiple sources, standardize the presentation, and reduce the cognitive load on security engineers. This approach

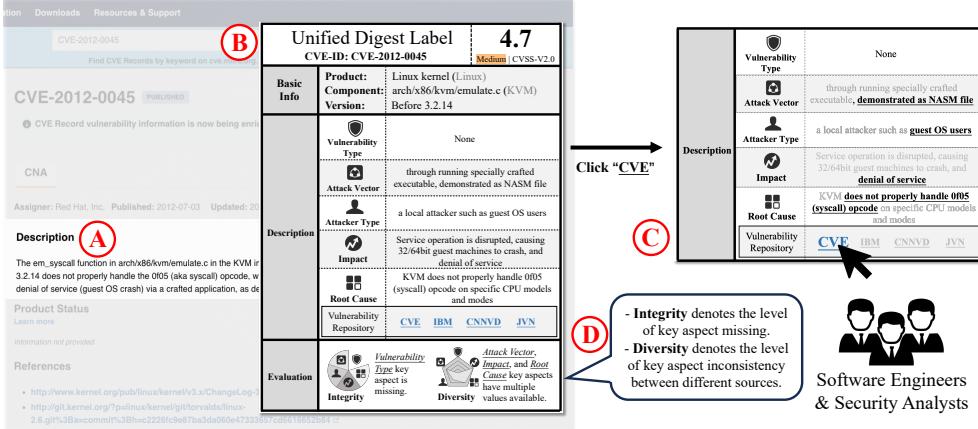


Fig. 3 The application scenario of Digest Labels (DLs). A) is the textual vulnerability description (TVD) available in a CVE repository. B) is the design of proposed DLs. C) is the *Description* section when the “CVE” is selected. D) is the explanations of *Evaluation* section.

enables engineers to quickly and efficiently receive essential information, improving their understanding and response times.

3.1 The Development of Nutrition Label in SE

3.1.1 Pre-millennium

Labels denotes Comments for Codes. In the early stages of software engineering, the concept of labeling primarily manifest in code comments and documentation [18, 19]. These labels provide information about inputs, outputs, code functionality, and dependencies, in a standardized manner, aiding developers to quickly understand and maintain the programs.

3.1.2 Post-millennium

Labels denote Metadata for Software. With the proliferation of open-source software and libraries, package managers such as Maven, NPM, and PyPI introduced metadata labels [20, 21]. These labels include information on versioning, dependencies, and licenses, helping developers make more informed decisions when utilizing third-party libraries. Recently, the Software Bills of Material (SBOM) attract considerable attention, considering as a crucial building block to ensure the transparency of software supply chains [22].

3.1.3 Present

Labels denote Information Transparency for Users and Developers. Currently, labels serve to provide key information through standardization, helping users better understand and assess the dataset [23], model [24], security [25], privacy [26], compliance [27], and other important aspects of software systems. Standardized

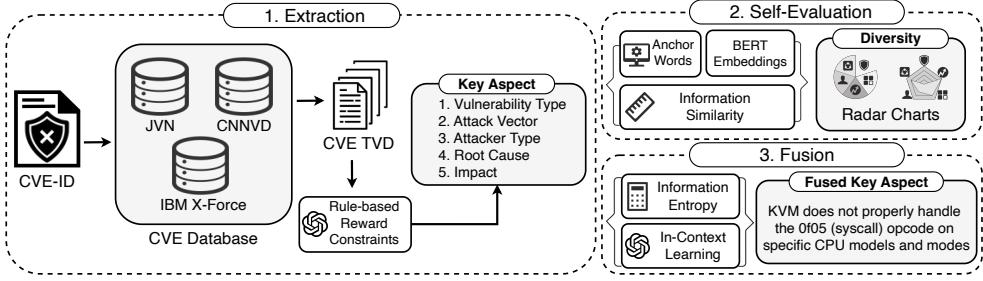


Fig. 4 The overview of framework.

labels enhance transparency and accountability, improve user experience, and promote responsible and transparent development in the software engineering field. As the modern software becomes larger and more sophisticated, developers share the same need as users to understand software information in a timely manner.

3.2 Design of Digest Labels (DLs)

Our DLs design aims to address the two major challenges identified in the formative study, missing and inconsistent key aspects in CVE TVD. The designed label is shown in (a) in Figure 3.

The DLs is indexed by the CVE-ID, as a unique identifier to indicate CVEs. Common Vulnerability Scoring System (CVSS) is a method used to supply a qualitative measure of severity [28], and software engineers highly depend on the scores to estimate the severity of the problem and decide the priority. The DLs contains three sections, Basic Information, Description, and Evaluation. Basic Information section presents the *Product*, *Component*, and *Version*. For any inconsistencies in the software name and version, the most frequent aspects are displayed in solid black, while the remaining inconsistent aspects are shown in lighter grey as reference. The second section is the key aspects, where the default view shows the merged results using the method from Section 4.3. The third section is the evaluation of the TVDs. *Integrity* denotes the level of key aspect missing and *Diversity* denotes the level of key aspect inconsistency between different sources.

To better demonstrate the inconsistencies in information across different repositories, we design a interactive presentation for Description section. This design allows viewers to click different information sources in the “Vulnerability Repository”. For example, clicking the “CVE” option will show the key aspects that mainly extracted and contributed by the “CVE”, as shown in Figure 3(c).

4 Domain-Constrained Synthesis

The domain-constrained synthesis framework for key aspects is illustrated in Figure 4. It consists of three main components: key aspect extraction, self-evaluation and key aspect fusion.

4.1 Extraction through Rule-Based Reward Constraints

Extracting key aspects from TVDs is challenging due to the varied expressions in which information is presented. Traditional methods [6, 29, 30] based on In-Context Learning focus on identifying structured terms but struggle with the diversity of key aspect expressions in vulnerabilities [10–12], leading to missed critical details.

Rule-based Reward is proposed by [31], to address data annotation issues by synthesizing training data with AI feedback, combined with human data, for supervised fine-tuning (SFT) and reward model (RM) training. The essence of Rule Based Reward lies in using general human guidelines to direct AI generation. We apply the Rule Based Reward to prompts by using general human guidelines as substitutes for examples to guide the LLM. In extracting key aspects of vulnerabilities, these general human guidelines are regularization templates, which are summarized by previous researches [6]. These templates, acting as domain-specific constraints, explicitly define the essential elements of vulnerabilities (e.g., “opcode”, “specific CPU models”). By replacing in-context learning examples with LLM extraction constitution, we can overcome the limitation of example numbers. This approach uses human experience to guide the extraction process of LLM. The list of regularization templates for key aspects is available in our artefact repository.

4.2 Self-Evaluation Though Anchor Words Constraints

The Evaluation section in DLs aims to provide a comprehensive quality assessment of the TVDs. While prior studies [7–9, 16, 17] focus on statistical analysis, we evaluate key aspects based on two criteria: information integrity and diversity (Figure 3).

Integrity. Information integrity assesses whether all key aspects are present. Complete key aspects enhance user understanding of vulnerabilities. If all aspects are present, we display a pie chart illustrating the five key aspects and a message stating “There are no missing key aspects”. On the contrary, missing aspects result in blank segments on the pie chart, and curated by side. The calculation of Integrity is merely the difference in the number of key aspects. Therefore, the calculation process relies on the extraction of key aspects, and the accuracy of the Integrity depends on the accuracy of key aspect extraction.

Diversity. Information diversity measures overlap among key aspects. High diversity, indicated by low overlap, offers valuable unique information. We utilize BERT sentence embeddings to represent the semantic vectors corresponding to each key aspect, using SpaCy’s¹ Transformer module (spacy-transformers) to generate BERT embeddings for output. We calculate cosine similarity [32] between vectors. However, inconsistencies can lead to significant distances between vectors for synonymous key aspects. While expressions may differ, computer-specific terms—acting as semantic anchors—remain constant. Although there are dictionaries for general computer terminology², there is no dedicated dictionary for the vulnerability domain. The uniqueness of computer-specific terms makes it challenging to apply general-purpose computer

¹<https://spacy.io/>

²<https://github.com/sohale/cs-glossaries>

Algorithm 1 Distance Calculation of Key Aspects

Require: Aspects = $[a_1, a_2, \dots, a_n]$

```
1: function LLM_EXTRACT(a)
2:   prompt = "Extract computer specific terms from sentence: "
3:   prompt = prompt + a
4:   return LLM(prompt)
5: end function
6: similar = []
7: for each  $i$  in Aspects do
8:   for each  $j$  in Aspects do
9:     temp1 = BertVec(LLM_EXTRACT(i))
10:    temp2 = BertVec(LLM_EXTRACT(j))
11:    sim = Cosine(temp1, temp2)
12:    similar.append(sim)
13:   end for
14: end for
15: res = average(similar)
16: return res
```

dictionaries in this context. For example, terms like “buffer overflow” and “SQL injection” have specific meanings in vulnerability contexts that may not be adequately captured in broader computer science glossaries. To impose constraints in the similarity computation, we employ LLM to extract domain-specific anchor words from each key aspect. These anchor words serve as constraint points that guide the semantic similarity comparison. The process is detailed in Algorithm 1, with *Aspects* representing all key aspects under consideration.

The information dispersion value ranges from $[0, 1]$, with a 0.2 mapping to the 5-point Likert. Both information dispersion and information completeness are represented using radar charts, located in the upper right corner of the label. After calculating the distances between the key aspects, we compute the distances for different values within the same key aspect type and then find the average distance. Once the distances for all five key aspect values are calculated, we plot these values on a radar chart. Any key aspect types with values greater than 2 are also noted in text.

4.3 Fusion through Entropy Constraints

As shown in Table 1, the inconsistency rate for key aspects in TVDs ranges from 0.36 to 0.67, indicating that nearly half have more than one value. We further statistically count the number of values for each key aspect across four vulnerability repositories (CVE, IBM, CNNVD, and JVN), finding that most of them have 2 to 3 different values, as illustrated in Figure 5. Additionally, we conduct a further word count analysis, revealing that the average lengths for key aspects: *Vulnerability Type* (5.1 words), *Attacker Type* (3.8 words), *Attack Vector* (13.6 words), *Impact* (9.7 words), and *Root Cause* (8.4 words). Thus, security engineers read an average of 24 words per key aspect type, totaling around 120 words for all descriptive aspects, many of which are

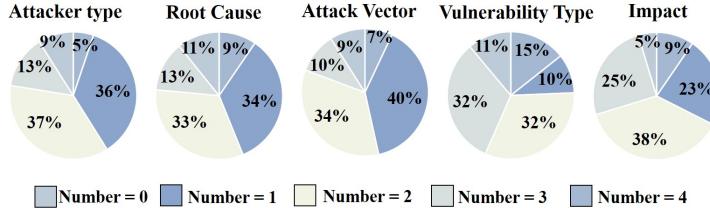


Fig. 5 Numbers proportion of key aspect values

similar, plus the additional time to cross-reference. Our fusion approach seeks to maximize **non-consensus information** from various sources, refining the presentation unlike the structured information in TVDs. Information Entropy [33] is an indicator to measure information richness in a sentence without being affected by expression, and we utilize it to achieve the maximal non-consensus information. By maximizing entropy with an effective aggregation method, we ensure comprehensive retention of information, aiming to avoid critical details being lost or duplicated. By preserving their diversity and clarity, our approach could be useful for vulnerability analysis, management, and CVE-related downstream tasks.

Specifically, the key aspect information entropy calculation method is shown in Equation 2. Let $s_1, s_2..s_n$ denote n key aspects that need to be merged. The function $\text{connect}(s_1, s_2, \dots, s_n)$ represents the concatenation of $s_1, s_2..s_n$. $\text{count}(w_i)$ indicates the number of times the word w_i appears in set S .

$$S = \text{connect}(s_1, s_2, \dots, s_n) \quad (1)$$

$$E = \sum_{i=1}^{\text{len}(S)} \left(\frac{\text{count}(w_i)}{\text{len}(S)} \log \frac{\text{count}(w_i)}{\text{len}(S)} \right) \quad (2)$$

After calculating the information entropy, we need to combine TVDs into semantically readable and syntactically correct descriptions to display. LLM can leverage their sentence rewriting abilities to combine sentences [34, 35]. However, when merging content between sentences, LLM tend to merge all sentences, even if there is no semantic overlap. This increase in sentence complexity can hinder the readability of security engineers. To mitigate this, we introduce entropy-based constraints into the prompt design. By heuristically embedding the calculated information entropy, we guide the LLM to focus on the most informative and non-redundant elements during fusion [36–39]. Even when different expressions describe the same key aspect, the entropy constraint preserves the most relevant content, enforcing a form of information-aware merging. Specifically, we use word-frequency-based information entropy to quantify the richness of each candidate and apply this value as a constraint that discourages over-merging and encourages information preservation. We compute entropy among key aspects and provide these values to the LLM to assist in sentence merging, as shown in Algorithm 2. The sentence_list consists of examples randomly selected by humans, and the merge_result is manually constructed for supervision.

Algorithm 2 Key Aspect Merging by Information Entropy

```
1: Input: sentence_list =  $[s_1, s_2, \dots, s_n]$ 
2: Output: merge_result
3: Example1:
4:   Task: "Based on information entropy, I can merge sentence_list."
5:   Sentence_list:  $[k_1, k_2, \dots, k_n]$ 
6:   Information entropy:  $E$  of sentence_list
7:   Merge result: (merge of sentence_list)
8: Example2: ...
9: Example3: ...
10: Task Description:
11:   Task: "Based on information entropy, I can merge sentence_list."
12:   Sentence_list:  $[s_1, s_2, \dots, s_n]$ 
13:   Information entropy:  $E$  of sentence_list
14:   Merge result
```

Our core objective is to ensure users receive the maximum amount of valuable information without redundancy. We achieve this through two key strategies:

- a) Information presentation. As shown in Figure 3, part B offers four different repository options. Users can click on each repository to view the specific information it provides. To avoid wasting users' time, redundancy must be minimized through content constraints.
- b) Application of Information Entropy as Constraints. During the information fusion process, entropy values act as soft constraints to regulate LLM behavior, encouraging minimal redundancy and maximizing informativeness. Unlike the diversity metric in Section 4.2, which measures semantic difference, entropy captures content richness while ignoring superficial syntactic variations. This constraint mechanism ensures that only meaningful differences are preserved in the final output.

4.4 Synthesis Baseline: Vanilla LLM Workflow

To provide a fair comparison, we introduce a baseline workflow that simulates how practitioners might directly use LLM without additional domain-specific strategies. As shown in Algorithm 3, the vanilla workflow only involves prompting the LLM to (a) summarize TVDs for a given CVE-ID, and (b) directly extract, merge, and evaluate key aspects from the summarized text. This baseline reflects the most straightforward usage of LLM, where no reward-based constraints, anchor words, or entropy-based fusion strategies are applied. We use this baseline to evaluate whether our proposed framework provides substantial improvements over the naive application of LLM.

To further strengthen the baseline, we also consider a *Chain-of-Thought (CoT) prompting* variant, which encourages the LLM to reason step by step before generating the final outputs. This variant builds on the same input/output format of Algorithm 3,

Algorithm 3 Label Generation Prompt

- 1: **Input:** Three randomly selected CVE-IDs
 - 2: **Output:** Extracted Key Aspects, Merged Key Aspects, TVD Evaluation
 - 3: **Task Description:**
 - 4: **Task:** “Based on the provided examples, please return:”
 - 5: **CVE-ID:** (CVE-ID)
 - 6: **TVDs:** {(TVD from CVE), (TVD from CNVND),...}
 - 7: **Please Return:** Extracted Key Aspects, Merged Key Aspects, TVD Evaluation.
-

but modifies the task instruction to explicitly require intermediate reasoning steps, as shown in Algorithm 4

Algorithm 4 Chain-of-Thought Prompting Baseline

- 1: **Input/Output:** Same as Algorithm 3.
 - 2: **Task Description:**
 - 3: **Instruction:** “Think step by step before answering.”
 - 4: Step 1: Read all provided TVDs carefully.
 - 5: Step 2: Identify candidate key aspects from each TVD.
 - 6: Step 3: Compare candidate aspects and remove duplicates or conflicts.
 - 7: Step 4: Return final merged aspects and evaluation.
 - 8: **Please Return:** Same as Algorithm 3.
-

5 Evaluation

This section evaluates the performance of the generation framework and its usefulness.

5.1 Dataset

We collected 289,105 CVE-IDs from 1999 to 2023, resulting in 956,619 TVDs from four repositories: CVE, IBM X-Force (IBM), CNVND, and JVN. A representative subset of 384 CVE-IDs was randomly sampled with a 95% confidence level and 5% margin of error.

To further demonstrate the representativeness of our sample, we provide its distribution across time and software types. As shown in Table 2, the 384 CVE-IDs are evenly distributed over the years from 1999 to 2023, and they cover a diverse set of software domains including operating systems, web applications, databases, libraries, and middleware. This distribution indicates that our sample captures both temporal and domain diversity, thereby supporting the generalizability of our evaluation.

5.2 Construction of Ground Truth

Table 2 Distribution of the sampled 384 CVE-IDs across years and software types.

Category	Subcategory	# Samples	Proportion
Year Range	1999–2005	42	10.9%
	2006–2010	56	14.6%
	2011–2015	73	19.0%
	2016–2020	102	26.6%
	2021–2023	111	28.9%
Software Type	Operating Systems	88	22.9%
	Web Applications/Frameworks	105	27.3%
	Database Systems	41	10.7%
	Programming Languages/Libraries	52	13.5%
	Middleware/Virtualization	46	12.0%
	Others (IoT/Drivers)	52	13.5%
Total		384	100%

As there is no existing ground truth for our novel task of TVD synthesis, we manually annotated the dataset to create the benchmark for evaluation. The annotation process was conducted as follows:

Annotator Recruitment and Qualifications: We recruited five evaluators, all of whom are professionals or researchers specializing in software vulnerabilities. Each evaluator had either participated in software vulnerability projects for more than three years or had at least two years of experience and had published papers on security vulnerabilities.

Annotation Task: For each of the 384 CVE-IDs in our sample, evaluators were asked to label the output of each sub-task in our framework as *True* or *False* based on the following criteria:

- For Key Aspect Extraction (Section 5.4.1), *True* means that the extracted key aspect is correct and appears in the source TVD, while *False* indicates an incorrect or missing extraction.
- For Key Aspect Fusion (Section 5.4.3), *True* signifies that the merged key aspects have neither lost essential information nor added extraneous information, whereas *False* indicates a loss or unjustified addition of information.
- For TVD Evaluation (Section 5.4.2), *True* indicates that the algorithm’s computed dispersion level (among the 5 levels) matches the human judgment, while *False* indicates a discrepancy.

Calibration and Adjudication Process: Before the formal annotation, we conducted a calibration stage in which the five evaluators independently annotated a small subset of CVEs. During this stage, Fleiss’ Kappa values in some cases fell below 0.2, which is generally regarded as poor agreement beyond chance [40], reflecting the difficulty and subjectivity of certain samples. To resolve these disagreements, the annotators engaged in structured discussions until they reached a shared understanding of the labeling criteria. The annotation guidelines were refined accordingly to reduce ambiguity. After calibration, the evaluators proceeded to annotate the full dataset,

and any residual disagreements were resolved by majority vote, a standard practice in annotation studies [41, 42].

Final Inter-Annotator Agreement: This process resulted in a high-quality ground truth dataset with substantial agreement. The final Fleiss' Kappa values for the tasks were as follows:

- **0.806** for Key Aspect Extraction (Figure 5.4.1),
- **0.781** for TVD Evaluation (Figure 5.4.2),
- **0.835** for Key Aspect Fusion (Table 3),
- **0.784** for the overall framework evaluation (Table 4).

These values indicate a substantial level of agreement, ensuring the reliability of our ground truth for the subsequent evaluation of the framework's performance.

5.3 Experimental Setup

To ensure the reproducibility of our experiments, we provide a detailed description of our LLM configurations and prompting strategies. All experiments were conducted using the ERNIE-3.5 API³ as the primary large language model, with a fixed parameter configuration: model version `ernie-3.5-8k`, temperature=0.8, top-p=1.0, frequency penalty=0, presence penalty=0, and max output tokens=4,096. We follow the common practices of LLM tasks for those settings. These hyperparameters were kept constant across all tasks.

Our prompting strategy integrates domain-specific constraints into each sub-task. For Key Aspect Extraction (Section 4.1), we employ *rule-based rewards* by embedding human-summarized regularization templates [6] as guidelines within the prompt, instructing the model to adhere to these domain-specific structures for identification. For Self-Evaluation (Section 4.2), the prompt is designed to extract semantic anchor words using the direct instruction: “Extract computer-specific terms from the following sentence: [sentence]”. For Key Aspect Fusion (Section 4.3), we use a few-shot learning prompt (Algorithm 2) where the calculated information entropy of the input sentences is provided as a key constraint to guide the model towards merging for maximal non-redundant information.

5.4 Performance of Synthesis

We first individually evaluate three sub-tasks for each section, and then assess the overall performance of the framework in solving the two challenges: information missing and inconsistency. Notably, we select Ernie-3.5⁴ as our default LLM choice due to its performance, accessibility, and cost-effectiveness.

5.4.1 Key Aspect Extraction

Figure 5.4.1 compares the performance of different key aspect extraction methods, and we select the following baseline methods:

³<https://yiyan.baidu.com/>

⁴<http://research.baidu.com/Blog/index-view?id=185>

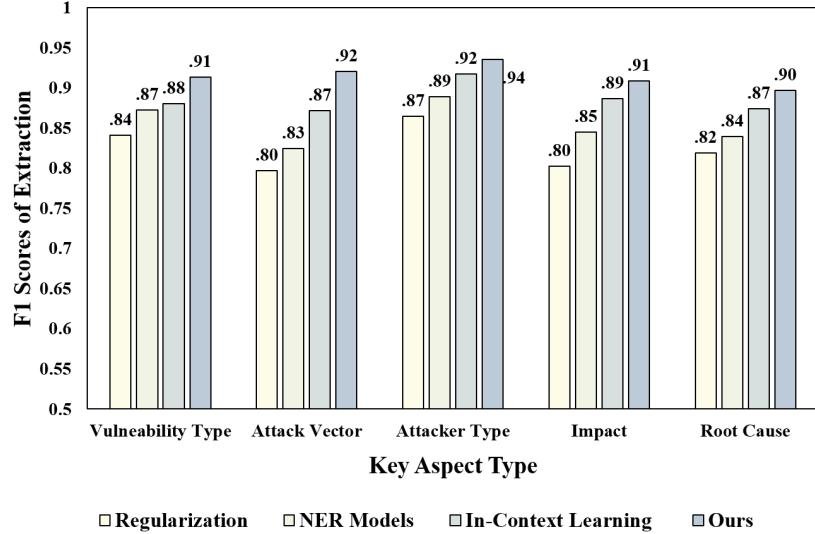


Fig. 6 F1 Score of key aspects extraction by different methods.

- **[Regularization]**: Guo et al. [6] summarize TVDs regularization templates and extract key aspects based on these templates.
- **[NER Models]**: Named-entity recognition (NER) models for vulnerability [43] use unsupervised clustering for the automated classification of key aspects.
- **[In-context Learning]** Example-based in-context learning [10–12] provide several examples in the prompt and then query the LLM to obtain results.

Our rule-based reward method achieves the highest F1 scores across all five key aspects: 0.91 for *Vulnerability Type*, 0.92 for *Attack Vector*, 0.94 for *Attacker Type*, 0.91 for *Impact*, and 0.90 for *Root Cause*. In comparison, the in-context learning method performs slightly worse, followed by NER models and the regularization method. Overall, our approach demonstrates superior performance in extracting key aspects.

5.4.2 Key Aspect Self-Evaluation

We choose two widely-adopted baseline methods to compare with our approach: a) Using BERT similarity directly as the distance between key aspects without employing a multi-agent approach; b) Glossary lists about software vulnerability. Specifically, we obtain the glossary lists by searching keywords “computer science glossary” (dubbed as CSG glossary) and “IT terminology list” (dubbed as ITL glossary), to collect relevant resources and combine them as glossary lists. Both lists are available at our artefact repository. If words or phrases from the TVD appear in these glossaries, we then extract and replace them, i.e., substituting lines 2-4 of Algorithm 1 with the glossary extraction method.

Figure 5.4.2 compares the evaluation F1 score of TVD using different methods across five key aspects. Our method achieves the highest F1 score in all categories, with scores ranging from 0.82 to 0.85. The methods using CSG and ITL glossaries

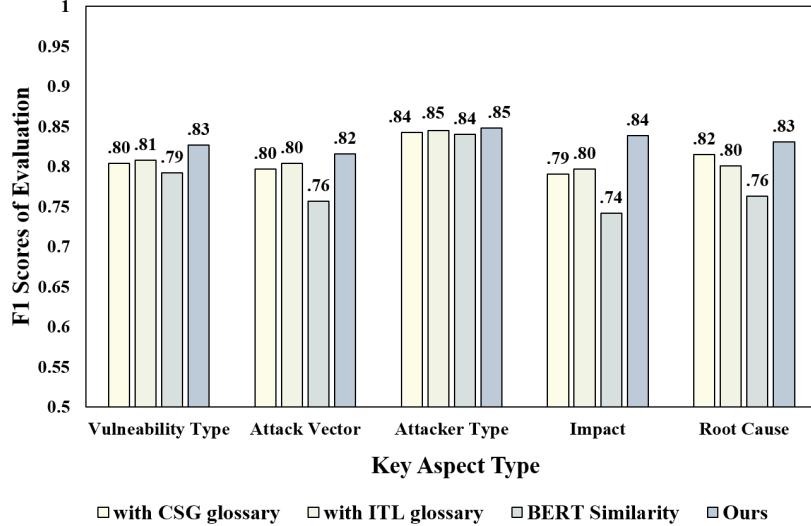


Fig. 7 F1 Score of key aspects self-evaluation by different methods.

Table 3 The F1 Score of key aspects fusion.

Methods	Vulnerability Type	Attack Vector	Attacker Type	Impact	Root Cause
No Heuri.	0.84	0.81	0.91	0.83	0.83
BERT Sim.	0.84	0.79	0.90	0.84	0.82
Ours	0.90	0.83	0.92	0.85	0.85

perform slightly lower, with accuracies between 0.79 and 0.84. The BERT similarity method shows the lowest F1 score, particularly in the *Attack Vector* and *Impact*. This highlights the effectiveness of our approach.

5.4.3 Key Aspect Fusion

Considering the powerful sentence processing capabilities of LLM, we use an LLM-based in-context learning for key aspect fusion. We compare our method with two benchmarks: 1) do not provide constraints in prompts (i.e., removing the “Information entropy” from Algorithm 2), and 2) using BERT semantic similarity between key aspects as information content prompt (i.e., replacing the “Information entropy” in Algorithm 2 with BERTScore similarity).

Figure 3 shows that our method, with information entropy as constraints, achieves high F1 scores across all five key aspects: 0.90 for *Vulnerability Type*, 0.83 for *Attack Vector*, 0.92 for *Attacker Type*, 0.85 for *Impact*, and 0.85 for *Root Cause*. In comparison, the method without an information content prompt and the BERT Similarity method show lower F1 Scores, demonstrating that incorporating information entropy significantly improves key aspect fusion.

Table 4 The overall performance on tackling the information missing and information inconsistency in CVE TVDs.

	Methods	Extract	Fusion	Self-evaluation	Average
F1	ERNIE + Alg. 3	0.88	0.84	0.76	0.82
	ERNIE + Alg. 4	0.87	0.86	0.78	0.84
	Ours (GPT-3.5)	0.90	0.88	0.82	0.86
	Ours (ERNIE)	0.92	0.87	0.83	0.87
Accuracy	ERNIE + Alg. 3	0.80	0.75	0.70	0.75
	ERNIE + Alg. 4	0.79	0.77	0.72	0.76
	Ours (GPT-3.5)	0.83	0.78	0.74	0.78
	Ours (ERNIE)	0.86	0.81	0.80	0.82
Precision	ERNIE + Alg. 3	0.85	0.78	0.72	0.78
	ERNIE + Alg. 4	0.84	0.80	0.74	0.79
	Ours (GPT-3.5)	0.88	0.83	0.75	0.82
	Ours (ERNIE)	0.90	0.81	0.79	0.83
Recall	ERNIE + Alg. 3	0.82	0.80	0.74	0.79
	ERNIE + Alg. 4	0.81	0.82	0.76	0.80
	Ours (GPT-3.5)	0.85	0.80	0.78	0.81
	Ours (ERNIE)	0.89	0.84	0.81	0.85

5.4.4 Overall Performance

Having defined the baseline workflow in Section 4.4, we now compare its performance with our proposed framework. Table 4 presents the results across the three sub-tasks: extraction, fusion, and self-evaluation. In addition to the vanilla LLM workflow, we also tested a stronger prompting variant using chain-of-thought (CoT) prompting as an alternative baseline. While CoT prompting improves certain sub-tasks slightly (e.g., fusion), it still shows limitations, particularly in overall consistency and self-evaluation (average F1=0.84). In contrast, our framework consistently outperforms both baseline variants, with ERNIE achieving the best overall performance (average F1=0.87). These results confirm that the domain-constrained design substantially improves the ability of LLM to handle information missing and inconsistency.

While our framework yields moderate overall gains in average performance, it remains unclear (1) why the improvement is not more substantial and (2) to what extent the gain stems from the underlying capability of LLM versus our integration algorithm. To address these questions, we conduct a fine-grained analysis across different key aspects. Specifically, we evaluate performance separately on each aspect—such as *Attack Vector*, *Root Cause*, and *Vulnerability Type*—to investigate whether the observed improvements concentrate on certain types of information. This helps reveal whether some aspects are inherently more difficult to extract and thus benefit more from our fusion strategy.

We test our framework with five representative LLM: GPT-3.5, GPT-4, Deepseek-R1, ERNIE, and LLaMA-3. These models are chosen to cover a broad spectrum of capabilities and architectural origins. GPT-3.5 and GPT-4 serve as widely adopted

Table 5 Performance across different key aspects with different LLM

LLM	Method	Metric	Att. Vec.	Ro. Cau.	Imp.	Vul. Typ.	Att. Typ.
GPT-3.5	Alg. 3	Acc	0.78	0.71	0.75	0.82	0.87
		Prec	0.80	0.66	0.70	0.79	0.86
		Rec	0.72	0.77	0.75	0.82	0.81
		F1	0.76	0.71	0.72	0.81	0.84
GPT-4	Ours	Acc	0.86	0.78	0.81	0.86	0.92
		Prec	0.85	0.75	0.76	0.85	0.91
		Rec	0.84	0.81	0.82	0.90	0.89
		F1	0.84	0.78	0.79	0.87	0.90
Deepseek	Alg. 3	Acc	0.83	0.77	0.79	0.87	0.90
		Prec	0.82	0.74	0.77	0.85	0.88
		Rec	0.81	0.78	0.79	0.88	0.89
		F1	0.81	0.76	0.78	0.86	0.88
Ours	Ours	Acc	0.87	0.81	0.82	0.90	0.92
		Prec	0.85	0.78	0.80	0.87	0.91
		Rec	0.84	0.82	0.83	0.91	0.90
		F1	0.86	0.80	0.81	0.90	0.91
ERNIE	Alg. 3	Acc	0.76	0.72	0.73	0.81	0.85
		Prec	0.77	0.68	0.71	0.79	0.84
		Rec	0.72	0.74	0.72	0.83	0.82
		F1	0.74	0.71	0.71	0.81	0.83
Ours	Ours	Acc	0.84	0.78	0.79	0.86	0.90
		Prec	0.83	0.75	0.76	0.85	0.89
		Rec	0.81	0.79	0.80	0.89	0.88
		F1	0.82	0.77	0.78	0.87	0.89
LLaMA-3	Alg. 3	Acc	0.71	0.75	0.69	0.83	0.83
		Prec	0.79	0.66	0.72	0.78	0.85
		Rec	0.66	0.75	0.70	0.87	0.84
		F1	0.71	0.73	0.74	0.83	0.87
Ours	Ours	Acc	0.88	0.82	0.80	0.87	0.90
		Prec	0.87	0.72	0.78	0.85	0.88
		Rec	0.81	0.82	0.81	0.91	0.95
		F1	0.84	0.80	0.79	0.89	0.90

general-purpose baselines, with GPT-4 representing the strongest closed-source model. Deepseek-R1 and LLaMA-3 are included as emerging open-source LLM with competitive performance, while ERNIE represents a knowledge-enhanced LLM that integrates external information into pretraining, making it a good testbed for evaluating synthesis methods. This selection allows us to examine whether our framework’s effectiveness is consistent across models of different strength, training data, and design philosophy.

Table 5 provides a fine-grained analysis across five key aspects and reveals that the performance improvement of our framework is not evenly distributed. Specifically, aspects such as *Attack Vector*, *Root Cause*, and *Impact* show relatively larger gains, reflecting their higher variability and the greater need for effective synthesis. In contrast, the improvements for *Vulnerability Type* and *Attacker Type* are comparatively smaller. For example, using ERNIE, the F1 score for *Root Cause* increases from 0.73 to 0.80, for *Impact* from 0.74 to 0.79, and for *Attack Vector* from 0.71 to 0.84. Similar patterns are observed in GPT-3.5 and Deepseek-R1, where diverse aspects benefit more from our synthesis framework.

At the same time, the magnitude of improvement varies across LLM. We observe the largest relative gains on weaker models such as ERNIE and Deepseek-R1, while stronger models like GPT-4 exhibit smaller but still consistent improvements (e.g., F1 for *Root Cause* increases from 0.76 to 0.81). This trend indicates that as LLM become stronger, they already capture much of the information, leaving less room for improvement, yet our framework continues to provide added value. Overall, the results demonstrate that our domain-constrained synthesis method consistently outperforms the baseline across all tested LLM, highlighting both its robustness and its particular effectiveness in handling diverse and challenging aspects.

5.4.5 Hallucination Evaluation in LLM-generated TVDs

In addition to performance metrics, we assess the reliability of generated key aspects by measuring hallucination, i.e., completely fabricated information. In cybersecurity, each key aspect serves as guidance for engineers to fix vulnerabilities. Even a single hallucinated element can mislead remediation efforts, so we adopt a strict criterion: any key aspect containing fabricated information is treated as a hallucination.

We randomly select 100 CVE entries from the test set, and generate key aspects for each entry using both the prompt-based baseline(Alg. 3) and our domain-constrained synthesis framework. Each generated key aspect is evaluated independently and assigned a binary label: Correct (1) if all information matches official vulnerability records, or Hallucinated (0) if it contains any fabricated information. We do not distinguish partially incorrect information, since in practice any hallucination in fusion outputs can severely mislead security engineers. For each key aspect type (e.g., *Attack Vector*, *Root Cause*, etc.), the hallucination rate is computed as:

$$\text{Hallucination Rate} = \frac{\text{hallucinated aspects}}{\text{total generated aspects}} \times 100\% \quad (3)$$

Table 6 Hallucination Rate for Each Key Aspect (calculated from fusion outputs)

LLM	Att. Vec.	Ro. Cau.	Imp.	Vul. Typ.	Att. Typ.
GPT-3.5 (Baseline)	0.15	0.22	0.18	0.14	0.12
GPT-3.5 (Ours)	0.08	0.14	0.12	0.10	0.08
GPT-4 (Baseline)	0.12	0.18	0.15	0.11	0.09
GPT-4 (Ours)	0.06	0.10	0.09	0.06	0.05
Deepseek (Baseline)	0.20	0.23	0.21	0.17	0.15
Deepseek (Ours)	0.11	0.15	0.13	0.10	0.09
ERNIE (Baseline)	0.25	0.21	0.24	0.19	0.17
ERNIE (Ours)	0.12	0.10	0.11	0.09	0.07
LLaMA-3 (Baseline)	0.14	0.20	0.16	0.12	0.10
LLaMA-3 (Ours)	0.07	0.11	0.10	0.06	0.05

Table 6 shows hallucination rates for each key aspect. Our framework consistently reduces hallucinations across all LLM. For example, GPT-3.5 drops from 0.22 to 0.14 in *Root Cause* and from 0.15 to 0.08 in *Attack Vector*. ERNIE has the highest baseline hallucinations, but our method effectively lowers them. Reductions are largest for complex aspects like *Root Cause* and *Impact*, demonstrating that the framework improves both accuracy and reliability by mitigating fabricated outputs. Our framework effectively constrains outputs through rule-based rewards, anchor-word alignment and entropy-guided fusion, reducing hallucination consistently across LLM and key aspects. This strict, aspect-wise metric provides a conservative yet practical measure of reliability.

To further demonstrate the differences between baselines and our method, we analyze the root cause of CVE-2012-0045 of Figure 1. The input TVDs from different vulnerability knowledge bases are as follows: CVE states that “KVM does not properly handle the 0f05 opcode”; CNNVD specifies that “KVM improperly handles syscall instructions in specific CPU modes on certain CPUs”; IBM reports “unable to handle opcode 0f05 correctly”; and JVN describes “the improper handling of the syscall opcode 0f05 by KVM on specific CPU models.”

The vanilla baseline (GPT-4, Alg. 3) generates “KVM fails to handle the 0f05 syscall opcode.” The CoT baseline (GPT-4, Alg. 4) improves slightly with “KVM improperly handles the 0f05 syscall in specific CPU modes.” In contrast, our method produces a more precise description: “KVM does not properly handle the 0f05 (syscall) opcode on specific CPU models and modes.”

The vanilla baseline omits critical qualifiers and produces an oversimplified root cause. The CoT baseline partially recovers “CPU modes” but still misses the “CPU models” qualifier. Our method integrates all key information from multiple sources, preserving both “CPU modes” and “CPU models,” which avoids misleading generalizations and ensures precise knowledge fusion.

5.5 Digest Labels System in Practice

Nutrition labels in the food industry integrate key ingredients and components for quick consumer understanding. This concept has been widely adopted across various fields in computer science, including security [25], privacy [26], and compliance [27].

Inspired by this, we further develop Digest Labels (DLs) system, aiming to synergistically synthesize information from different TVDs in a standardized and efficient manner, as shown in Figure 3. We evaluate the usability of DLs system for security analysts, focusing on enhancing vulnerability remediation.

A Real-world Scenario. New TVDs, often lacking CVE-IDs and missing key aspects, are frequently published. Practitioners typically search for similar TVDs to fill in missing information. In this scenario, participants select the correct missing key aspect from five options to demonstrate their understanding of the vulnerability.

Metrics. We use two performance indicators: **Comprehension:** If the selected option is correct, it is marked *True*, otherwise *False*. The F1 Score is calculated based on these predictions. **Efficiency:** The average time spent per participant is recorded.

Evaluation Design. We randomly mask one key aspect for all 384 CVEs and form five candidate options with the masked aspect and four irrelevant options. Participants are divided into three groups: Control Group 1 (CVE-only), Control Group 2 (all four repositories), and the Experiment Group (with DLs). All non-English TVDs are translated into English via Google Translate. A follow-up survey is conducted after the session.

In total, twelve participants were recruited, including nine graduate students specializing primarily in software security and three doctoral students with advanced research backgrounds in software and system security. While the majority of participants focused on software security, some had broader research interests that spanned multiple areas, including database systems and dialogue systems, which added complementary perspectives to the study. In terms of gender, nine participants were male and three were female. We intentionally sought to avoid an excessive gender imbalance, but given the engineering-oriented nature of our research institution, recruiting female researchers with relevant expertise was inherently more challenging. Nevertheless, this configuration still provided diversity across gender and research focus, while maintaining the necessary specialization for high-quality annotations. All participants were from CS/IT backgrounds with prior security-related research experience, ensuring that the task's domain-specific demands could be met. They were evenly distributed across the three groups and first familiarized with the DLs and other methods. The graduate students contributed practical perspectives grounded in hands-on vulnerability analysis, while the doctoral students approached the tasks from a higher-level research-oriented standpoint, bringing broader conceptual insights. This mixed configuration was designed to improve the robustness of the study without relying solely on increasing the number of participants of the same profile.

Nevertheless, we acknowledge that our study remains limited by its modest scale and academic context. While the inclusion of doctoral students provides complementary expertise, a larger and more diverse pool of participants, especially security practitioners from industry, would be necessary to further validate the generalizability of our findings. That said, our current participant setting is consistent with the practice of prior qualitative studies: Sai et al.[44] involved 10 participants, Zhang et al.[45] included 4 participants, and Jonathan et al. [46] relied on 8 participants, indicating that a small but focused group can provide valid and meaningful insights in this type of research. And these examples come from software engineering, database, and

Table 7 The comprehension and efficiency in the Real-world scenario.

Methods	Control Group 1	Control Group 2	Experimental Group
Comprehension (F1 Score)	0.65	0.82	0.86
Efficiency (Time)	26.51	42.14	27.80

The F1 Score of question answering reflects the comprehension, and the average completion time (seconds) per question denotes the efficiency.

AI domains, respectively, suggesting that our participant scale aligns with common practice across multiple research areas.

Table 8 Question sheet of user study.

No.	Question
Q1	How often do you encounter missing/inconsistent TVD information? (0: never, 5: often)
Q2	Can you describe your usual approach to understanding security vulnerabilities?
Q3	What difficulties do you face when analyzing security vulnerabilities using TVDs?
Q4	How user-friendly is the tool for the given task? (0: not, 5: very)
Q5	How useful is the tool in completing the given task? (0: not useful, 5: very useful)
Q6	How effective is the tool in understanding vulnerabilities? (0: ineffective, 5: effective)
Q7	How has DLs changed your approach to analyzing and understanding vulnerabilities?
Q8	Please provide your recommendations for improving the tool (DLs).

Notes: “TVD” refers to textual vulnerability descriptions; “DLs” refers to the proposed Digest Labels in Figure 3.

Table 9 Quantitative results of user study.

No.	Control Group 1	Control Group 2	Experimental Group (with DLs)
Q1	3.4	3.7	3.5
Q2	–	–	–
Q3	–	–	–
Q4	3.7	2.4	4.5
Q5	2.4	3.8	4.1
Q6	2.7	3.6	3.8
Q7	–	–	–
Q8	–	–	–

Results. Table 7 compares the performance of three groups in terms of comprehension and efficiency in tackling unseen incomplete TVDs. The method utilizing multiple web-based vulnerability repositories (multi-repositories) achieves an F1 score of 0.82 and requires an average of 42.14 seconds per evaluation. Using only the CVE database (with just CVE) results in a lower F1 score of 0.64 but is faster, with an average time of 26.51 seconds. The approach employing the Digest Labels demonstrates

the highest F1 Score of 0.86 while maintaining a relatively efficient average time of 27.80 seconds. This indicates that the DLs system strikes a satisfying balance between comprehension and efficiency, outperforming existing methods.

As shown in Table 8 and Table 9, after each participant completes their task, we conduct a survey to qualitatively evaluate the overall usefulness of DLs and its web-based generation tool. Firstly, **for challenges faced when using TVDs**, participants across all groups identify several common challenges when using TVDs for vulnerability analysis. They frequently encounter missing or inconsistent information (Q1), with ratings between 3 and 4, indicating the prevalence of identified challenges. To understand vulnerabilities (Q2), they typically rely on multiple databases, often supplementing with external resources like forums or technical blogs, though this method is time-consuming. For difficulties during analysis (Q3), participants mention that inconsistent data structures across different databases make it hard to perform comparisons. Additionally, they struggle to distinguish between similar vulnerabilities and outdated entries, as these issues often lead to confusion and delays.

Secondly, **for the tool-specific feedback** (Q4, Q5), the experimental group gives DLs consistently high ratings, finding it convenient and useful, with average scores closer to 4.5. They note that DLs' clear structure and intuitive design make it significantly easier to complete their tasks. In contrast, Control Group 1, which uses multiple repositories, finds the process cumbersome, often rating convenience and usefulness between 2 and 4 due to the time required to navigate different sources. Control Group 2, using only CVE, also provides moderate ratings, citing the lack of in-depth information as a barrier to task completion.

Thirdly, **for understanding vulnerabilities** (Q6), the experimental group reports higher effectiveness compared to both control groups, as DLs provide clear, well-organized information. Control Group 1 also performs reasonably but continues to struggle with fragmented data from multiple repositories, leading to lower scores. Control Group 2 finds the CVE entries too brief and lacking in detail, which hampers their ability to fully understand vulnerabilities, resulting in the lowest ratings among the groups.

Finally, **for the impact of DLs on analyzing vulnerabilities** (Q7 and Q8), participants emphasize that DLs simplifies their workflow, making it easier to locate key details without switching between multiple sources.

Overall, participants frequently report encountering missing or inconsistent information in TVDs (Q1), and describe in open-ended responses (Q2–Q3) how such fragmentation makes vulnerability comprehension time-consuming and error-prone. Meanwhile, participants rate the proposed DLs highly in terms of usefulness and clarity (Q4–Q6), suggesting that multi-aspect, structured information substantially eases the analytical burden. These findings highlight a key need in vulnerability analysis: synthesizing scattered details into coherent, accessible summaries. DLs directly address this gap (Q7–Q8) by offering a unified, aspect-based view that streamlines vulnerability comprehension.

Some participants recommend improving the search function and adding options like sorting by relevance or severity to enhance usability further. Additionally, two notable suggestions are: a) adding more links to security reports for quick access, and

b) enhancing the DLs interface with a vulnerability history tracking feature to improve visibility into repair progress. Overall, the DLs and its web-based generation tool are widely appreciated for their delivered information and user-friendly design.

6 Discussion

6.1 Significance for Practitioners

Assist Vulnerability Management. From Figure 2, it is evident that the quality of web-based vulnerability repositories established by different institutions varies significantly. Web-based vulnerability repositories are the cornerstone of vulnerability management. Despite different institutions needing to maintain the same software (e.g., Python packages, GitHub open-source software), there is a considerable disparity in vulnerability defect management levels. This disparity leads to a gap between the demand for software maintenance and the institution’s maintenance capabilities, with the gap being wider for institutions with lower vulnerability management levels. The proposed DLs consolidates TVDs from multiple data sources, timely contributing to bridging this gap.

New Vulnerability Description System. Vulnerability key aspects are diverse, and different perspectives lead to varied vulnerability descriptions. Using a single-sentence TVD as a vulnerability description is unreasonable, as there may be multiple key aspects values in the same key aspect type. A single sentence cannot adequately capture this complexity. Instead, our label format can be adopted to build a new vulnerability description system.

6.2 Reproducibility and Manual Efforts

Our framework is designed to be highly reproducible and low-effort, with minimal human intervention throughout the pipeline. All major components—including key aspect extraction, evaluation, and fusion—are either fully automated or require only minimal, one-time setup.

Key aspect extraction is entirely automated using prompts based on domain-specific templates adapted from prior work [6]. These templates are embedded directly into the prompts and require no manual labeling or in-context example crafting. The evaluation stage, which assesses both integrity and diversity, is also fully automated. Diversity is computed via cosine similarity of BERT embeddings with LLM-assisted anchor word selection, while integrity is determined by parsing LLM outputs for the presence of key aspects. Neither step involves manual inspection or labeling. The fusion stage uses entropy-guided prompt construction and LLM-based rewriting to synthesize coherent outputs. Only a small set of initial examples (typically 3–5) needs to be prepared in advance, and these are reused across all samples. All merging steps are handled by LLM without manual editing. Prompt templates for all components follow fixed structures and are generated programmatically, with no need for per-sample customization. Data retrieval from public vulnerability repositories (e.g., CVE, CNVND)

is conducted automatically. The only manual input is a user-specified list of CVE-IDs of interest.

In summary, after an initial one-time setup of domain templates and fusion examples, the entire framework can be executed end-to-end without additional human effort. This design enables strong reproducibility across datasets and LLM backends, and allows easy extension to other domains.

6.3 Potential for Downstream Tasks: Enhancing Security Tasks via DLs

We analyzed the downstream potential of our Digest Labels (DLs) system by referring to recent survey papers on LLM in software security [47, 48]. We focus on survey literature because such works offer a consolidated view of the challenges and needs in real-world tasks, which helps to identify stable and high-impact integration points. Among these, a recurring issue is the limited utility of raw textual vulnerability descriptions due to inconsistency, incompleteness, and lack of structure.

Our work addresses these issues by introducing DLs, a unified labeling system that extracts and organizes five key aspects of vulnerabilities. DLs are not a model but a curated output format that synthesizes multiple sources into structured, semantically complete records. This system offers consistency across repositories, preserves domain-critical details, and provides a stable interface for downstream tasks.

With DLs in place, we plan to explore new directions for enhancing a range of TVD-related security tasks. First, DLs enable models to shift from unstructured text processing to structured reasoning, allowing aspect-level comparisons and classification, such as vulnerability classification or CWE/CVSS prediction. Second, DLs offer a foundation for interpretable rule construction, helping analysts identify key aspect combinations that frequently appear in high-risk cases, which is valuable for tasks like vulnerability function identification (VFI). Third, DLs can improve LLM-based security systems by serving as clean, disambiguated inputs for prompt design or training data preparation, thereby facilitating tasks such as patch matching and exploit prediction. These directions apply broadly to TVD-driven tasks and are part of our planned future work.

6.4 Threats to Validity

Internal Validity. The first step in DLs is key aspect extraction. We use the rule based reward method, employing regularization templates as human data to guide LLM generation. However, we are not certain that regularization templates are the most suitable form of human guidance for LLM. In the future, we aim to explore the optimal integration of human expertise with LLM.

External Validity. External validity threats include limited generalizability to other web-based vulnerability repositories. Our method relies on web-based vulnerability repositories being indexed by CVE-ID. However, many repositories, such as BSI

and CERT-FR, use their own independent indexing systems. This hinders the transferability of our method to BSI and CERT-FR. Once those repositories are indexed by CVE-ID, our method can be expanded accordingly.

Construct Validity. Our hallucination evaluation relies on official records as ground truth and a binary labeling scheme. While this provides a clear benchmark, the manual verification process may introduce subjectivity in borderline cases. Additionally, the sample size of 100 CVEs, though sufficient for comparison, might not capture all edge cases of hallucination.

7 Related Work

Inconsistencies in Vulnerability Descriptions. Textual Vulnerability Descriptions (TVDs) provided by repositories such as CVE [1], NVD [2], CNNVD, and IBM X-Force [3] often exhibit inconsistencies for the same vulnerability. Prior work has investigated this issue from the perspective of knowledge comparison and alignment. For example, Dong et al. [7] and Sun et al. [9] studied linguistic discrepancies in TVDs and attempted to mitigate them through comparison against authoritative knowledge bases. Similarly, Huang et al. [49] surveyed methods for detecting and resolving inconsistencies in software models, which has been extended to the security domain. He et al. [8] further revealed the pervasiveness of inconsistencies across repositories. However, these studies generally treat discrepancies as errors to be eliminated, filtering out details that do not align with external knowledge. As illustrated in Figure 1, such filtering can lead to the loss of critical contextual details—e.g., while CVE describes the root cause of CVE-2012-0045 as “does not properly handle the 0f05 opcode,” CNNVD adds important specificity, noting that the issue arises in “specific CPU modes on certain CPU models.” Existing methods risk discarding such complementary perspectives.

LLM-Driven Synthesis. Recent advances in Large Language Models (LLMs) have opened new opportunities for synthesizing inconsistent information. Common strategies such as in-context learning and chain-of-thought prompting [29, 50, 51] guide models to merge multiple descriptions into a unified output. Nevertheless, these strategies typically rely on high-level semantic cues, e.g., “synthesize vulnerability descriptions with as much detail as possible,” which provide only soft guidance. Without explicit domain-specific constraints, LLMs may produce vague or incomplete results. For instance, when tasked to synthesize the root cause of CVE-2012-0045, unconstrained models often generate “KVM fails to handle the 0f05 syscall opcode,” omitting critical qualifiers such as “specific CPU models” or “modes.” Thus, while promising, current LLM-based synthesis approaches struggle to enforce structured, domain-sensitive rules during generation.

Usability in Software Security. Another line of work emphasizes usability in security tools, aiming to improve practitioners’ efficiency. Garfinkel and Spafford [52] highlight the importance of user-friendly vulnerability management systems, while Green and Smith [53] show that poor usability often leads to tool resistance. Chu et al. [54] and Han et al. [55] propose “security labels” inspired by nutrition labels to enhance users’ understanding of security information. Similar approaches have been explored in static and dynamic analysis tools [56–58], which provide more accessible interfaces and real-time feedback. These works demonstrate the value of standardized

and interpretable representations for practitioners, though they have not been applied to synthesizing vulnerability descriptions.

Summary and Differentiation. In summary, prior studies either (i) mitigate inconsistencies by aligning TVDs with external knowledge bases, thereby discarding potentially valuable complementary details, or (ii) explore the usability of security tools through interface design and labeling strategies. Our work differs in two key aspects. First, rather than treating inconsistencies as errors, we view them through the lens of the *Blind Men and an Elephant* theory: different repositories offer partial perspectives that, when integrated, yield a more comprehensive understanding of vulnerabilities. For example, the case of CVE-2012-0045 shows how details about “specific CPU models” and “modes” are crucial for understanding exploitability, but would be lost under prior inconsistency-mitigation approaches. Second, while LLM-based synthesis has been attempted, we introduce domain-constrained synthesis, which enforces structured, security-specific rules (e.g., anchor terms like “opcode” or “privilege escalation”) during generation to retain technical precision. Finally, building upon usability research, we are the first to design **Digest Labels (DLs)**, a nutrition-label inspired system for vulnerabilities, which synergistically combines domain-constrained synthesis with a standardized, practitioner-friendly representation. This unique integration of synthesis and usability directly addresses gaps left open by both inconsistency-focused and usability-focused prior work.

8 Conclusion

In this paper, we introduce a domain-constrained LLM-based synthesis framework for synthesizing vulnerability key aspects from multiple repositories. By incorporating domain-specific constraints such as Rule-Based Rewards, Anchor Words, and Information Entropy, our method improves the extraction, self-evaluation, and fusion of vulnerability data. Results show the framework improves the F1 score of synthesis performance from 0.82 to 0.87, while enhancing comprehension and efficiency by over 30%. We further develop Digest Labels, a practical tool for visualizing TVDs, which human evaluations show significantly boosts usability.

References

- [1] Common Vulnerabilities and Exposures. <https://cve.mitre.org/>
- [2] Standards, N.I., Technology: National Vulnerability Database (NVD). Accessed: 2024-07-13 (2023). <https://nvd.nist.gov>
- [3] IBM X-Force Exchange. <https://exchange.xforce.ibmcloud.com/>
- [4] Vulnerability management resource guide. Technical report, Cybersecurity and Infrastructure Security Agency (CISA) (2022). URL: https://www.cisa.gov/sites/default/files/publications/CRR_Resource_Guide-VM_0.pdf

- [5] BS ISO/IEC 29147:2018 - Information Technology – Security Techniques – Vulnerability Disclosure. British Standards Institution. <https://www.bsigroup.com/en-GB/standards/bs-iso-iec-29147-2018/>
- [6] Guo, H., Chen, S., Xing, Z., Li, X., Bai, Y., Sun, J.: Detecting and augmenting missing key aspects in vulnerability descriptions. ACM Trans. Softw. Eng. Methodol. **31**(3), 49–14927 (2022) <https://doi.org/10.1145/3498537>
- [7] Dong, Y., Guo, W., Chen, Y., Xing, X., Zhang, Y., Wang, G.: Towards the detection of inconsistencies in public security vulnerability reports. In: Heninger, N., Traynor, P. (eds.) 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019, pp. 869–885. USENIX Association, Berkeley, CA, USA (2019). <https://www.usenix.org/conference/usenixsecurity19/presentation/dong>
- [8] He, Y., Wang, Y., Zhu, S., Wang, W., Zhang, Y., Li, Q., Yu, A.: Automatically identifying CVE affected versions with patches and developer logs. IEEE Trans. Dependable Secur. Comput. **21**(2), 905–919 (2024) <https://doi.org/10.1109/TDSC.2023.3264567>
- [9] Sun, H., Ou, G., Zheng, Z., Liao, L., Wang, H., Zhang, Y.: Inconsistent measurement and incorrect detection of software names in security vulnerability reports. Comput. Secur. **135**, 103477 (2023) <https://doi.org/10.1016/J.COSE.2023.103477>
- [10] Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.: REALM: retrieval-augmented language model pre-training. CoRR **abs/2002.08909** (2020) [2002.08909](https://arxiv.org/abs/2002.08909)
- [11] Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q.V., Chi, E.H.: Least-to-most prompting enables complex reasoning in large language models. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, Kigali, Rwanda (2023). <https://openreview.net/pdf?id=WZH7099tgfM>
- [12] Massacci, F.: The holy grail of vulnerability predictions. IEEE Secur. Priv. **22**(1), 4–6 (2024) <https://doi.org/10.1109/MSEC.2023.3333936>
- [13] Sun, J., Xing, Z., Xu, X., Zhu, L., Lu, Q.: Heterogeneous vulnerability report traceability recovery by vulnerability aspect matching. In: IEEE International Conference on Software Maintenance and Evolution, ICSME 2022, Limassol, Cyprus, October 3-7, 2022, pp. 175–186. IEEE, Piscataway, NJ, USA (2022). <https://doi.org/10.1109/ICSME55016.2022.00024> . <https://doi.org/10.1109/ICSME55016.2022.00024>
- [14] Sun, J., Xing, Z., Lu, Q., Xu, X., Zhu, L.: A multi-faceted vulnerability searching website powered by aspect-level vulnerability knowledge

- graph. In: 45th IEEE/ACM International Conference on Software Engineering: ICSE 2023 Companion Proceedings, Melbourne, Australia, May 14-20, 2023, pp. 60–63. IEEE, Piscataway, NJ, USA (2023). <https://doi.org/10.1109/ICSE-COMPANION58688.2023.00025> . <https://doi.org/10.1109/ICSE-Companion58688.2023.00025>
- [15] Guo, H., Xing, Z., Chen, S., Li, X., Bai, Y., Zhang, H.: Key aspects augmentation of vulnerability description based on multiple security databases. In: IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC 2021, Madrid, Spain, July 12-16, 2021, pp. 1020–1025. IEEE, Piscataway, NJ, USA (2021). <https://doi.org/10.1109/COMPSAC51774.2021.00138> . <https://doi.org/10.1109/COMPSAC51774.2021.00138>
- [16] Qin, Y., Xiao, Y., Liao, X.: Vulnerability intelligence alignment via masked graph attention networks. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023, pp. 2202–2216. ACM, Copenhagen, Denmark (2023). <https://doi.org/10.1145/3576915.3616686> . <https://doi.org/10.1145/3576915.3616686>
- [17] Sun, J., Xing, Z., Xia, X., Lu, Q., Xu, X., Zhu, L.: Aspect-level information discrepancies across heterogeneous vulnerability reports: Severity, types and detection methods. *ACM Trans. Softw. Eng. Methodol.* **33**(2), 49–14938 (2024) <https://doi.org/10.1145/3624734>
- [18] Boehm, B.W.: A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes* **11**(4), 14–24 (1988)
- [19] Pressman, R.S.: Software Engineering: A Practitioner’s Approach. McGraw-Hill, New York, USA (2005)
- [20] Microsoft: Maven: A software project management and comprehension tool. In: Proceedings of the 2009 International Conference on Software Engineering (2009)
- [21] npm, I.: npm: A node.js package manager. *IEEE Software* **31**(2), 11–14 (2014)
- [22] Xia, B., Bi, T., Xing, Z., Lu, Q., Zhu, L.: An empirical study on software bill of materials: Where we stand and the road ahead. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 2630–2642 (2023). IEEE
- [23] Pushkarna, M., Zaldivar, A., Kjartansson, O.: Data cards: Purposeful and transparent dataset documentation for responsible ai. In: Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, pp. 1776–1826 (2022)
- [24] Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., *et al.*: Model cards for model reporting. In: Proceedings of the Conference on

Fairness, Accountability, and Transparency, pp. 220–229 (2019)

- [25] Simko, L., Roesner, F., Kohno, T.: Ask the experts: What should be on an iot privacy and security label? *IEEE Security & Privacy* **17**(5), 8–16 (2019)
- [26] Pan, S., Hoang, T., Zhang, D., Xing, Z., Xu, X., Lu, Q., Staples, M.: Toward the cure of privacy policy reading phobia: Automated generation of privacy nutrition labels from privacy policies. arXiv preprint arXiv:2306.10923 (2023)
- [27] Si, M., Pan, S., Liao, D., Sun, X., Tao, Z., Shi, W., Xing, Z.: A solution toward transparent and practical ai regulation: Privacy nutrition labels for open-source generative ai-based applications. arXiv preprint arXiv:2407.15407 (2024)
- [28] National Institute of Standards and Technology. <https://www.nist.gov/>
- [29] Han, L., Pan, S., Xing, Z., Sun, J., Yitagesu, S., Zhang, X., Feng, Z.: Don't chase your tail! missing key aspects augmentation in textual vulnerability descriptions of long-tail software through feature inference. CoRR **abs/2405.07430** (2024) <https://doi.org/10.48550/ARXIV.2405.07430> 2405.07430
- [30] Yitagesu, S., Xing, Z., Zhang, X., Feng, Z., Li, X., Han, L.: Unsupervised labeling and extraction of phrase-based concepts in vulnerability descriptions. In: 36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021, pp. 943–954. IEEE, Piscataway, NJ, USA (2021). <https://doi.org/10.1109/ASE51524.2021.9678638> . <https://doi.org/10.1109/ASE51524.2021.9678638>
- [31] OpenAI: Rule-based rewards for language model safety (2023)
- [32] Luo, C., Zhan, J., Xue, X., Wang, L., Ren, R., Yang, Q.: Cosine normalization: Using cosine similarity instead of dot product in neural networks. In: Kurková, V., Manolopoulos, Y., Hammer, B., Iliadis, L.S., Maglogiannis, I. (eds.) Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11139, pp. 382–391. Springer, Cham, Switzerland (2018). https://doi.org/10.1007/978-3-030-01418-6_38 . https://doi.org/10.1007/978-3-030-01418-6_38
- [33] Shannon, C.E.: A mathematical theory of communication. *Bell System Technical Journal* **27**(3), 379–423 (1948) <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>
- [34] Shu, L., Luo, L., Hoskere, J., Zhu, Y., Liu, Y., Tong, S., Chen, J., Meng, L.: Rewritelm: An instruction-tuned large language model for text rewriting. In: Wooldridge, M.J., Dy, J.G., Natarajan, S. (eds.) Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on

- Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada, pp. 18970–18980. AAAI Press, Palo Alto, California, USA (2024). <https://doi.org/10.1609/AAAI.V38I17.29863> . <https://doi.org/10.1609/aaai.v38i17.29863>
- [35] Tan, H., Sun, F., Yang, W., Wang, Y., Cao, Q., Cheng, X.: Blinded by generated contexts: How language models merge generated and retrieved contexts for open-domain qa? CoRR **abs/2401.11911** (2024) <https://doi.org/10.48550/ARXIV.2401.11911> **2401.11911**
 - [36] Yu, S., Cong, J., Liang, J., Liu, H.: The distribution of information content in english sentences. CoRR **abs/1609.07681** (2016) **1609.07681**
 - [37] Zhai, C., Lafferty, J.D.: A study of smoothing methods for language models applied to ad hoc information retrieval. SIGIR Forum **51**(2), 268–276 (2017) <https://doi.org/10.1145/3130348.3130377>
 - [38] Bentz, C., Alikaniotis, D., Cysouw, M., Ferrer-i-Cancho, R.: The entropy of words - learnability and expressivity across more than 1000 languages. Entropy **19**(6), 275 (2017) <https://doi.org/10.3390/E19060275>
 - [39] Levshina, N.: Frequency, informativity and word length: Insights from typologically diverse corpora. Entropy **24**(2), 280 (2022) <https://doi.org/10.3390/E24020280>
 - [40] McHugh, M.L.: Interrater reliability: the kappa statistic. Biochemia Medica **22**(3), 276–282 (2012)
 - [41] Bornmann, L., Daniel, H.-D.: Manuscript and reviewer characteristics that influence the peer review process: The case of the journals in the ‘web of science’subject category ‘information science & library science’. Journal of Informetrics **5**(1), 137–158 (2011)
 - [42] Hallgren, K.A.: Computing inter-rater reliability for observational data: An overview and tutorial. Tutorials in Quantitative Methods for Psychology **8**(1), 23–34 (2012)
 - [43] Yitagesu, S., Xing, Z., Zhang, X., Feng, Z., Li, X., Han, L.: Extraction of phrase-based concepts in vulnerability descriptions through unsupervised labeling. ACM Trans. Softw. Eng. Methodol. **32**(5), 112–111245 (2023) <https://doi.org/10.1145/3579638>
 - [44] Zhang, S., Xing, Z., Guo, R., Xu, F., Chen, L., Zhang, Z., Zhang, X., Feng, Z., Zhuang, Z.: Empowering agile-based generative software development through human-ai teamwork. ACM Trans. Softw. Eng. Methodol. **34**(6), 156–115646 (2025) <https://doi.org/10.1145/3702987>

- [45] Zhang, S., Zhang, J., Song, X., Adeshina, S., Zheng, D., Faloutsos, C., Sun, Y.: Page-link: Path-based graph neural network explanation for heterogeneous link prediction. In: Ding, Y., Tang, J., Sequeda, J.F., Aroyo, L., Castillo, C., Houben, G. (eds.) Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023, pp. 3784–3793. ACM, ??? (2023). <https://doi.org/10.1145/3543507.3583511> . <https://doi.org/10.1145/3543507.3583511>
- [46] Skaggs, J., Richards, M., Morris, M., Goodrich, M.A., Crandall, J.W.: Fostering collective action in complex societies using community-based agents. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024, pp. 211–219. ijcai.org, ??? (2024). <https://www.ijcai.org/proceedings/2024/24>
- [47] Zhu, X., Zhou, W., Han, Q., Ma, W., Wen, S., Xiang, Y.: When software security meets large language models: A survey. IEEE CAA J. Autom. Sinica **12**(2), 317–334 (2025) <https://doi.org/10.1109/JAS.2024.124971>
- [48] Yitagesu, S., Xing, Z., Zhang, X., Feng, Z., Bi, T., Han, L., Li, X.: Systematic literature review on software security vulnerability information extraction. ACM Trans. Softw. Eng. Methodol. (2025) <https://doi.org/10.1145/3745026> . Just Accepted
- [49] Huang, X., Li, X., Zheng, S.: Detecting and resolving inconsistencies in software models: A survey. IEEE Transactions on Software Engineering **46**(2), 165–180 (2020) <https://doi.org/10.1109/TSE.2019.2925660>
- [50] Guo, S., Wang, Q., Gao, Y., Xie, R., Song, L.: Depth-guided robust and fast point cloud fusion nerf for sparse input views. In: Wooldridge, M.J., Dy, J.G., Nataraajan, S. (eds.) Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada, pp. 1976–1984. AAAI Press, Palo Alto, California, USA (2024). [https://doi.org/10.1609/aaai.v38i3.27968](https://doi.org/10.1609/AAAI.V38I3.27968) . <https://doi.org/10.1609/aaai.v38i3.27968>
- [51] Wang, L., Xu, W.: Hybrid approaches for defect detection: Combining static and dynamic analysis techniques. IEEE Transactions on Reliability **71**(1), 105–120 (2022) <https://doi.org/10.1109/TR.2022.3148729>
- [52] Garfinkel, S., Spafford, G.: Practical Unix and Internet Security. O'Reilly Media, Inc., Sebastopol, California, USA (2002)
- [53] Green, M., Smith, J.: Developer barriers to security tool usage: Understanding and addressing usability challenges. IEEE Security & Privacy **16**(5), 45–53 (2018)
- [54] Chu, H., Zhang, M.: Security labels for software packages: Improving security

- understanding and decision-making. *Journal of Information Security* **8**(2), 127–138 (2017)
- [55] Han, L., Liu, W.: A framework for standardized security labels: Enhancing software security comprehension. *IEEE Transactions on Software Engineering* **46**(3), 344–355 (2020)
 - [56] Xie, T., Li, P.: Static analysis tool for vulnerability detection: Improving accuracy with user-friendly interfaces. *ACM Transactions on Software Engineering and Methodology* **27**(4), 1–26 (2018)
 - [57] Kim, J., Lee, S.: Code analysis tool combining static and dynamic techniques for enhanced vulnerability detection. *Journal of Systems and Software* **158**, 110404 (2019)
 - [58] Li, X., Zhang, Q.: Interactive security tools with real-time feedback for integrated software development. *Empirical Software Engineering* **25**(6), 4854–4878 (2020)