# From Industrial Practices to Academia: Uncovering the Gap in Vulnerability Research and Practice

Zhuang Liu
*Zhejiang University, China*
liuzhuang@zju.edu.cn

Xing Hu†
*Zhejiang University, China*
xinghu@zju.edu.cn

Jiayuan Zhou
*Queen's University, Canada*
jiayuan.zhou@queensu.ca

Xin Xia
*Zhejiang University, China*
xin.xia@acm.org

***Abstract*—The rising number of vulnerabilities has attracted significant attention from academia and industry. While the Common Vulnerabilities and Exposures (CVE) database is an industry-standard resource for organizing and researching vulnerabilities, it lacks comprehensive analysis, often requiring researchers to conduct additional investigations. This gap in detailed vulnerability information can hinder effective vulnerability management and research. To address this problem, we conduct the first empirical investigation to discover the disparities in vulnerability aspects between academia and industry. We collect a comprehensive dataset comprising 50,254 security bulletins and blogs from 36 CVE Numbering Authorities (CNAs). We extract and summarize the specific characteristics of vulnerabilities from these web pages and identify 15 key aspects for describing vulnerabilities. Our analysis reveals that the detailed information provided by different CNAs varies significantly. The industrial practice primarily emphasizes post-disclosure aspects of vulnerabilities, such as *Impact* (82.1%) and *Measures* (i.e., *Solution* and *Mitigation*), while largely overlooking *Attacker Type* (almost none), *Attack Scenario* (0.3%), and details on *Steps to Reproduce* (0.2%) and *Vulnerability Validation & Exploitation* (almost none). We also systematically review 31 academic papers on vulnerabilities to identify the primary aspects of academic research. Our findings indicate the lack of research on *Attack Scenario* and *Attack Method* in academia. Academic research on vulnerabilities primarily focuses on *Fix/Patch Release* (13 out of 31), with significant attention to patch generation, porting, search, and vulnerability repair. By offering insights to industry and academia, we aim to stimulate the advancement of vulnerability research. In the future, the aspect *Attack scenario* of vulnerabilities holds the potential for breakthrough advancements, benefiting both industry and academia.**

***Index Terms*—Vulnerability, Aspects, Industry, Academia**

## I. Introduction

The increasing number of disclosed vulnerabilities poses a significant challenge to the software industry and security community [1]. Many companies and security service providers refer to these vulnerabilities from the National Vulnerability Database (NVD), which is a widely used public source for new OSS vulnerabilities [2]. NVD fully syncs with the Common Vulnerabilities and Exposures (CVE) list, i.e., once a CVE record is published, it will appear in the NVD immediately [3]. Thus, CVE entries (e.g., vulnerability description) are one of the most important fields for security community [4]. In addition to the CVE information, the NVD also provides other information, including the Common Vulnerability Scoring System (CVSS) vector/score, Common Weakness Enumeration (CWE) category, and Common Platform Enumeration (CPE). Considering the extensive security information, existing studies usually exploit the NVD as the data source to explore vulnerability research, for example, vulnerability exploitability prediction [5], [6], severity level prediction [7], [8], and vulnerability type classification [9]–[11].

However, NVD data has some limitations and potential pitfalls: ❶ *Incomplete Data*. Although MITRE recommends describing a vulnerability from six aspects, the descriptions often lack some aspects and this situation still exists after NVD artificial enhancement [12]. The absence of these aspects affects subsequent vulnerability-related tasks. For example, the lack of "affected products" in the description has negative impacts on vulnerability severity prediction [13]. ❷ *Over-reliance on CVSS Scores*. Previous studies assume that CVSS scores are available during the process of vulnerability prediction [6]. Chen et al. [6] find that 49% of real-world exploits occur before CVSS scores are published. It means that studies relying on CVSS scores for prediction may overlook nearly half of the exploit cases. ❸ *Accuracy and Quality*. Dong et al. [14] present concerns (i.e., the consistency of software names and versions between NVD descriptions and unstructured vulnerability reports) about the quality of NVD information. In addition, there are inaccuracies in NVD entries regarding the publication dates of vulnerabilities, software versions, severity scores, and types of vulnerabilities [2], [14]. ❹ *Limited Context*: Each CVE ID has a web page with a short description of the vulnerability and a list of external references [15]. They lack detailed context, such as the likelihood of exploitation or the impact on specific industries or use cases. For instance, the contextual information for CVE-2023-27882[1] in NVD is far simpler than its corresponding security bulletin[2]. However, most vulnerability-related research overlooks the insights provided by security bulletins [4].

Given the limitations and pitfalls of NVD vulnerability data, some studies have focused on enhancing the quality and completeness of vulnerability information [16]. However, there is a lack of a comprehensive study that clarifies which aspects of vulnerability information these tools or methods

†Corresponding author.

[1]https://nvd.nist.gov/vuln/detail/CVE-2023-27882

[2]https://talosintelligence.com/vulnerability_reports/TALOS-2023-1733

have addressed and improved. Additionally, it is unclear which aspects remain underexplored and require further investigation. This study aims to bridge the gap between the aspects of vulnerabilities that are prioritized in industry and those researched in academia. To comprehensively analyze the aspects of vulnerability that are prioritized in industrial practice, we examine the security bulletins issued by CVE Numbering Authorities (CNAs), which often contain more detailed and comprehensive descriptions of vulnerabilities. By analyzing the CVE list up to 2023, we identify the 36 CNAs most frequently associated with assigned CVE IDs and collect their corresponding published security bulletins. We combine large language models (LLMs) with manual analysis and summarize 15 key vulnerability aspects derived from these security bulletins.

From an academic perspective, we review 31 academic papers published between 2018 and 2023 that focus on various aspects of vulnerability information and management. Finally, we compare the aspects of vulnerabilities prioritized in the industry with those emphasized in academia, identifying the gaps and exploring the reasons for these differences. This will provide insights into how industry and academic researchers prioritize and utilize specific aspects of vulnerabilities.

In particular, we investigate the following three research questions:

**RQ 1: Which aspects of vulnerability information are prioritized by the industry?** This question investigates the aspects of vulnerability information that draw the most attention within the industry. In addition to the seven aspects contained by NVD, CNA primarily emphasizes comprehensive measures after vulnerability discovery, including *Severity*, *Impact*, *Affected Product*, *Mitigation*, and *Solution*. The category of *Vulnerability Verification & exploitation* is sparsely populated by CNAs.

**RQ 2: What aspects of vulnerability are the focus of academic research?** This research question focuses on identifying the aspects of vulnerability information that receive attention within academia. Through a review of 31 academic papers, we find that the primary focus is on *Fix/Patch Release* (11 out of 31), including vulnerability repair, patch search/generation, and backporting. Other aspects that receive considerable attention include *Vulnerability Type*, *Severity*, *Affected Product*, *Impact*, *Root Cause*, and *Attack Vector*. However, there is little to no research on *Attack Method*, *Attack Scenario*, *Step to Reproduce*, and *Solution*. *Proof of Concept*, *Mitigation*, and *Attacker Type* are also underexplored and need further investigation.

**RQ3: What are the similarities and differences between the aspects that industry and academia focus on in vulnerability?** Industry places a high emphasis on *Mitigation* and *Solution* after vulnerability disclosures, but academic research in these aspects is limited. Only one recent study focuses on generating vulnerability mitigation reports. Additionally, there are few academic studies on *Attack Method* and *Proof of Concept* compared to industry. Academia focuses more on patches and repairs (*Fix/Patch Release*) than industry. Both academia and industry pay attention to the *Impact* of vulnerabilities, *Affected Product*, *Severity*, *Vulnerability Type*, *Attack Vector*, and *Root Cause*. However, both academia and industry lack sufficient research on *Steps to Reproduce*, *Exploit Demonstration*, *Attack Scenario*, and *Attacker Type*.

By analyzing security bulletins from the industry and academic papers, we aim to identify the current focus and research progress on vulnerabilities in both academia and industry. This paper makes the following contributions:

- The study empirically analyzes security bulletins published by CNA from the key vulnerability aspect. We collect 50,254 security advisory information published by 36 CNAs and make the dataset open source for further research.
- We summarize the 15 most critical vulnerability aspects and analyze the distribution of these security bulletins. The summarized aspects provide a comprehensive baseline for describing vulnerabilities in the industry.
- We review research literature on vulnerabilities published in the last six years. We analyze the differences in attention to vulnerability information between academia and industrial practice, which can motivate researchers to prioritize industry-relevant vulnerability aspects.

**Paper Organization.** The remained of this paper is organized as follows. Section II preliminary background knowledge on key aspects of vulnerability description. Section III provides a detailed account of our research methodology. Section IV presents our analysis and findings on the questions posed. Section V presents our discussions and insights from academia and industry on key aspects of vulnerability. Section VI presents related work on describing key aspects of vulnerabilities and the industry data of vulnerabilities in industry and academia. Finally, Section VII concludes the paper.

## II. Preliminaries

The CVE database leverages "Description" to document vulnerability details. MITRE recommends the utilization of two standardized vulnerability description templates [17]: ❶ Vulnerability Type in Component in Vendor | Product | Version allows Attacker Type to Impact via Attack Vector; ❷ Component in Vendor | Product | Version Root Cause, which allows Attacker Type to Impact via Attack Vector. The two templates outline six key vulnerability aspects (i.e., vulnerability type, affected product, root cause, attacker type, attack vector, and impact) as detailed subsequently [12]. For example, the description of CVE-2021-37211 is as follows: *"The bulletin function of Flygo does not filter special characters while a new announcement is added. Remoter attackers can use the vulnerability with general user's credential to inject JavaScript and execute stored XSS attacks."*. The above description involves the four key aspects of the *root cause* (does not filter special characters), *attacker type* (remoter attackers), *attack vector*

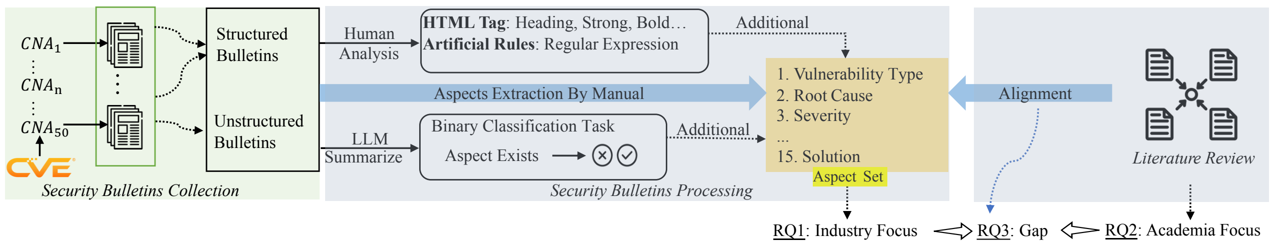


Fig. 1: The Overview of Our Research.

(inject JavaScript), and *vulnerability type* (stored XSS attacks). Furthermore, NVD includes severity as an aspect, reflected by the CVSS score. The following are seven aspects mentioned in NVD.

- **Vulnerability Type.** *Vulnerability Type* serves as an abstract characterization of a vulnerability, commonly classified using the Common Weakness Enumeration (CWE) [18], which is a standardized framework for identifying and describing prevalent security weaknesses in software and hardware systems [19].
- **Affected Product.** In the context of CVE description, the " Component in Vendor | Product | Version " serves as an identifier for the *Affected Product*. When submitting a new CVE record, reporters are required to furnish comprehensive information regarding the affected product, including its vendor and version.
- **Root Cause.** *Root Cause* indicates the errors, encompassing those in programming, value or condition validation, and system or environment configuration, which results in vulnerabilities.
- **Attacker Type.** *Attacker Type* illuminates the diverse methods attackers utilize to leverage system vulnerabilities. It encompasses three primary groups: location-based (remote or local access), permission-based (unauthorized or authorized users), physically proximate, and context-dependent attackers.
- **Attack Vector.** *Attack Vector* describes the specific inputs and/or processes an attacker can leverage to exploit a vulnerability.
- **Impact.** Impact denotes the potential consequences of a successful exploit, which reflects the potential gains an attacker could achieve by leveraging the vulnerability.
- **Severity.** *Severity* is typically indicated by the CVSS score, ranging from 0.0 to 10.0 points, and categorized into five levels: None (0.0), Low (0.1-3.9), Medium (4.0-6.9), High (7.0-8.9), and Critical (9.0-10.0).

## III. Methodology

Figure 1 provides an overview of our research process. First, we extract CNA names from the CVE list. We then gather the websites associated with each CNA, which contain a large number of security bulletins. We use heuristic rules to extract vulnerability aspects from structured security bulletins and employ GPT-3.5[3] to summarize unstructured ones. Subsequently, we conduct a literature review to assess the consistency between the vulnerability aspects addressed in academics and those in the industry.

### A. Industrial Data Collection

**CNAs Collection.** To identify the CNAs, we first download CVE data lists published between 1999 and 2023 from MITRE's website[4]. These data lists contain descriptive information about each CVE record, including the "assignerShortName" field, which identifies the CNA that assigned the corresponding CVE ID. We parse the entire dataset to count the occurrences of each CNA, resulting in a list of the 50 most frequent CNAs, such as *redhat*, *talos*, *f5*, *microsoft*, and *cisco*. We exclude CNAs from which we cannot obtain effective resources, such as *Vuldb* (which requires a subscription) and *huawei* (for which we lack access permissions). We then exclude the *MITRE* from our analysis to ensure a focused examination of vulnerability aspects employed by CNAs other than *MITRE*.

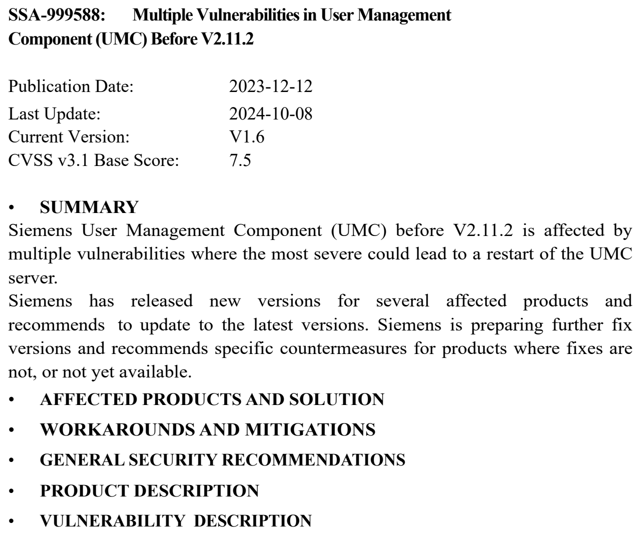
**Security Bulletin Collection.** For each CNA website (i.e., the official websites of CNA or dedicated security advisory pages), we configure a custom crawler to extract web page information related to security bulletins. We use regular expressions to parse and extract key content blocks, retaining only those pages that explicitly contain a CVE ID. We exclude CNAs with fewer than ten pages. As a result, we retain 36 CNAs and their associated valid page content. The list of 36 CNAs is available in our replication package[5].

### B. Bulletin Page Processing

*1) Preliminary Aspect Identification:* We begin with six CVE template-derived aspects and added one for severity from the NVD, totaling seven aspects (as mentioned in section II). Starting from this initial set of aspects, two authors randomly select a page to extract the vulnerability aspects for each CNA. This process continues until no new aspects are identified. They then record all the aspects found for that CNA. Notably, each author may extract different key aspects for a CNA at this stage. After identifying aspects for all CNAs, the authors compare and discuss the extracted aspects to reach

[3]https://platform.openai.com/docs/models#gpt-3-5-turbo
[4]https://www.cve.org/Downloads
[5]https://github.com/zhuang-liu-maker/Vulnerability-Data

a consensus. The two authors view a total of 478 web pages, averaging 13 per CNA. The overall Cohen's Kappa value between the two authors is from 0.57 to 1.0, which indicates substantial agreement between them. We adopt the union of their post-discussion aspects as our set of aspects.


**SSA-999588: Multiple Vulnerabilities in User Management Component (UMC) Before V2.11.2**

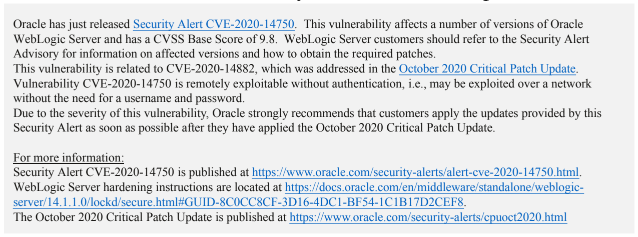
| | |
|---|---|
| Publication Date: | 2023-12-12 |
| Last Update: | 2024-10-08 |
| Current Version: | V1.6 |
| CVSS v3.1 Base Score: | 7.5 |

- **SUMMARY**

Siemens User Management Component (UMC) before V2.11.2 is affected by multiple vulnerabilities where the most severe could lead to a restart of the UMC server.
Siemens has released new versions for several affected products and recommends to update to the latest versions. Siemens is preparing further fix versions and recommends specific countermeasures for products where fixes are not, or not yet available.

- **AFFECTED PRODUCTS AND SOLUTION**
- **WORKAROUNDS AND MITIGATIONS**
- **GENERAL SECURITY RECOMMENDATIONS**
- **PRODUCT DESCRIPTION**
- **VULNERABILITY DESCRIPTION**

(a) Structured vulnerability information description [20].


Oracle has just released Security Alert CVE-2020-14750. This vulnerability affects a number of versions of Oracle WebLogic Server and has a CVSS Base Score of 9.8. WebLogic Server customers should refer to the Security Alert Advisory for information on affected versions and how to obtain the required patches.
This vulnerability is related to CVE-2020-14882, which was addressed in the October 2020 Critical Patch Update.
Vulnerability CVE-2020-14750 is remotely exploitable without authentication, i.e., may be exploited over a network without the need for a username and password.
Due to the severity of this vulnerability, Oracle strongly recommends that customers apply the updates provided by this Security Alert as soon as possible after they have applied the October 2020 Critical Patch Update.

For more information:
Security Alert CVE-2020-14750 is published at https://www.oracle.com/security-alerts/alert-cve-2020-14750.html.
WebLogic Server hardening instructions are located at https://docs.oracle.com/en/middleware/standalone/weblogic-server/14.1.1.0/lockd/secure.html#GUID-8C0CC8CF-3D16-4DC1-BF54-1C1B17D2CEF8.
The October 2020 Critical Patch Update is published at https://www.oracle.com/security-alerts/cpuoct2020.html

(b) Unstructured vulnerability information description [21].

Fig. 2: Vulnerability Information Description From Two Websites.

*2) Page Content Processing:* We devise two semi-automated processing methodologies to handle the diverse website data collected efficiently. The structured page shown in Figure 2a presents information organized into specific subsections or headings, effectively highlighting the key aspects of the vulnerability description. Figure 2b shows a typical unstructured page characterized by text-based content and lacking explicit structural elements. In this case, it is difficult to identify the key vulnerability aspects directly.

*3) Structured Page Processing:* **Structured Information Extraction.** The headings/bold or enlarged (sub)section names on a web page provide a concise overview of its content. It is crucial to extract these heading elements precisely. However, depending solely on the heading tag may not accurately determine context due to varying website designs. To address this challenge, we use *cisco* as an example to illustrate how to extract vulnerability information from structured pages. First, we extract headings from all *cisco* web pages and create a heading dictionary. We manually examine and exclude heading tags not part of the main content (such as *copyright notices*). We then count the number of headings on each web page and exclude headings that appear fewer than five times, as these are special cases and do not represent the structured website. The collection of headings collected from *cisco* is shown in Table I.

TABLE I: Heading Extraction and Summarization for *cisco* Website

| |
|---|
| • Summary/Details |
| • Affected Products |
| - Vulnerable Products |
| - Products Confirmed Not Vulnerable |
| • Fixed Software |
| - Fixed Releases |
| - Fixed Versions |
| • Exploitation and Public Announcements |
| • Workarounds |
| • Mitigations |

**Structured Information Analysis.** We analyze the content under each heading for *cisco* to align with our proposed aspects. Two authors iteratively discuss web pages until no new aspects are found under all headings. Following the analysis, we summarize *cisco*'s each heading's content as follows. *Summary/Details* provides an overview of the vulnerability's root cause, attack vectors, attack methods, types, and impact. *Affected Products* consists of two subsections. *Vulnerable Products* identifies vulnerable products, detailing names and versions. *Products Confirmed Not Vulnerable* lists products confirmed unaffected. *Fixed Software* outlines products receiving patches, including names and versions. It comprises two subsections. *Fixed Releases* provides detailed descriptions of releases containing fixes, specifying product and version information. *Fixed Versions* presents a tabular list of fixed versions. *Exploitation and Public Announcements* indicates whether vulnerabilities have been publicly exploited. *Workarounds* provides solutions for users to address vulnerabilities. *Mitigations* offers measures for users to reduce the impact of vulnerabilities.

Finally, we map the headings of *cisco* to the corresponding vulnerability aspects, as shown in Figure 3. We utilize the mapping relationship shown in Figure 3 to calculate the distribution of vulnerability aspects within the collected *cisco* web pages. The percentage of each aspect in Figure 3 shows the quantity distribution results. We subsequently apply a comparable methodology to analyze all CNA websites containing structured web information systematically. When analyzing structured page information, we incorporate the newly identified vulnerability aspects into the initially recognized set of aspects.

*4) Unstructured Page Processing:* We utilize GPT-3.5 for summarizing and extracting key aspects from lengthy web page content. While there are several alternative models (e.g., LLama 3[6]) with their unique strengths, GPT-3.5 was selected

[6]https://github.com/meta-llama/llama3

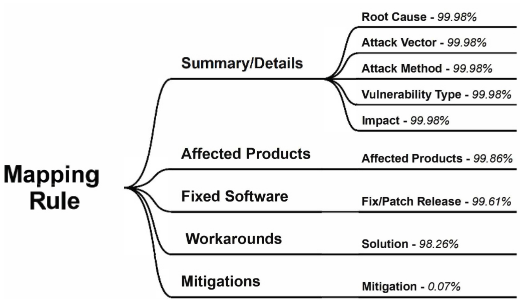


Fig. 3: Mapping Relationship Between *Cisco* Website Heading and Vulnerability Aspect.

TABLE II: Aspects of Vulnerability Information

| Category | Aspects |
|---|---|
| Vulnerability Type | Vulnerability Type |
| Root Cause | Root Cause |
| Attack Model | Attacker Type<br>Attack Scenario<br>Attack Method<br>Attack Vector |
| Severity | Severity |
| Vulnerability Verification & Demonstration | Proof of Concept<br>Exploit Demonstration<br>Steps to Reproduce |
| Fix/Patch Release | Fix/Patch Release |
| Impact | Affected Product<br>Impact |
| Measure | Mitigation<br>Solution |

for its balance of performance, reliability, and the extensive ecosystem support it offers. In addition, GPT-3.5 was chosen for its stability and the extensive dataset it was trained on, which provides a more generalized and reliable performance across various tasks [22], [23]. The model defines each aspect as a binary classification problem, providing a “Yes” or “No” response. A “Yes” indicates the presence of the aspect information within the web page’s content, while a “No” denotes its absence.

We use a generic prompt template so that it can be applied to all website information. We follow OpenAI’s prompt engineering tutorials[7] to construct prompts. Guided by the tutorial’s recommendations, we iteratively refined the prompts based on the following principles: ❶ Assign a role, write clear instructions, and limit the knowledge areas of answers (e.g., defining it as a “security vulnerability bulletin content summarization assistant”). ❷ Request a structured response format. For example, required a binary (Yes/No) indication of vulnerability aspect presence. ❸ Reiterative enhancement of prompt content through experimentation.

Following Sun et al. [24] utilization of machine reading comprehension models for aspect extraction, we incorporate explanatory statements following each queried aspect. We request only one aspect at a time (thus necessitating multiple separate requests for the same page content). Additionally, we conduct further queries on each security bulletin to determine if any other vulnerability aspects are present. Similar to identifying structured security bulletins, we incorporate the newly identified vulnerability aspects into the initially recognized set of aspects.

### C. Paper Data Collection

Research papers on vulnerabilities are usually published in the software engineering and security communities. Therefore, we review full research papers from the premier publication venues, including ICSE, ESEC/FSE, ASE, ISSTA, MSR, TSE, TOSEM, USENIX, NDSS, S&P, and CSS from 2018 to 2023. We select papers from the above conferences and journals because they are premier publication venues in software engineering and vulnerability research. State-of-the-art findings are published in these conferences and journals. We first collect all the papers from the above venues, searching the titles and abstracts for the keyword “vulnerab*” (where * is a wildcard). We then read the titles and abstracts of all papers that have been filtered by “vulnerab*” and judge whether each paper involves our identified vulnerability aspects (e.g., vulnerability type [25], patches [26], mitigation [27], and severity [28]). There is a large amount of research on vulnerabilities, such as vulnerability detection [1], [29] and attack methods for newly discovered vulnerabilities in the security field. Our paper focuses on the information gaps and research after disclosing open-source vulnerabilities. We filter out papers that deviate from our research focus. We also exclude empirical studies, survey articles, and short papers. For each vulnerability aspect-related article, two authors read and analyze the content to determine the primary focus of the study. Then the two authors discuss the differences in the vulnerability aspect analysis and confirm the final result through further paper reading.

[7]https://learn.deeplearning.ai/chatgpt-prompt-eng

Finally, we get vulnerability labels for each article, based on the aspects extracted from industry bulletin data. For instance, Bao et al. proposed V-SZZ [30], which can identify the versions affected by vulnerabilities, and we assign the label “Affected Product” to the article. It should be noted that an article can be assigned multiple labels. The [31] is assigned the labels “Vulnerability Type”, “Root Cause”, “Attack Vector”, and “Impact” because it uses an encoder-decoder model to predict whether OSS updates in dependent repositories (dependency updates) are vulnerability patches, and generate explainable aspect-level vulnerability information (i.e., the four aspects mentioned above).

## IV. Results

In this section, we first report the key aspects identified from industrial security bulletins. In section *Bulletin Page Processing* (III-B), we identify and summarize eight additional aspects beyond the seven aspects provided by the CVE template and NVD described in our preliminaries (section II). We present the additional categories and their respective aspects as follows:

- **Attack Model** contains four aspects: *Attacker Type*, *Attack Vector*, *Attack Scenario* (describes the background,

TABLE III: The Data Size of the Website Corresponding to the CNAs

| CNA ShortName | DataSize | Structured |
|---|---|---|
| cisco | 4,364 | ✓ |
| adobe | 313 | ✓ |
| intel | 653 | ✓ |
| talos | 1,722 | ✓ |
| dell | 998 | ✓ |
| zdi | 229 | ✗ |
| cert | 33 | ✗ |
| snyk | 155 | ✗ |
| microsoft | 73 | ✗ |
| schneider | 108 | ✗ |

conditions, and environment in which the attack occurs), and *Attack Method* (refers to the specific means or techniques used to achieve attack goals).

- **Vulnerability Verification & Demonstration** represents the verification and demonstration/exploitation of related vulnerabilities, which includes three aspects: *Proof of Concept (PoC)* (refers to a simple example created by vulnerability researchers to confirm the existence and exploitability of a vulnerability, typically including basic exploitation logic or code snippets), *Exploitation Demonstration* (refers to a vulnerability researcher demonstrating how to exploit a vulnerability, which may include showcasing the process of gaining system access or executing malicious code through a specific vulnerability), and *Steps to Reproduce* (a series of steps described in a vulnerability report that can be used to reproduce the vulnerability).
- **Fix/Patch Release** denotes the vendor's response regarding whether a fix or patch has been released and contains the *Fix/Patch Release* aspect.
- **Impact** contains two aspects: upstream impact, referring to the *Affected Product*, and downstream impact, denoting the *Impact* on users (Impact in CVE description templates).
- **Measure** includes *Mitigation* (advice given, recommended methods) and *Solution* (ways to resolve vulnerabilities).

In total, we categorize the 15 aspects into eight categories, as illustrated in Table II. Based on the aspects of vulnerabilities summarized above, we present our analysis and findings on the vulnerabilities within both academic and industrial contexts. We will answer the research questions proposed in the following sections.

## *A. RQ 1: Which aspects of vulnerability information are prioritized by the industry?*

Table III shows the names and data sizes of the ten CNAs. The data size represents the number of security bulletins obtained after filtering. We do not list all CNAs, further information regarding the data can be found in our accompanying attachment.

From the remaining unstructured 1,281 web pages, 10% of the responses are randomly selected for manual verification to evaluate accuracy. The GPT-3.5 demonstrates the following accuracy levels for each aspect: *Vulnerability Type* (90.48%), *Root Cause* (81.74%), *Attacker Type* (94.44%), *Attack Vector* (84.13%), *Attack Method* (92.06%), *Attack Scenario* (96.82%), *Severity* (96.03%), *Proof of Concept* (95.24%), *Exploitation Demonstration* (90.48%), *Steps to Reproduce* (98.41%), *Fix/-Patch Release* (94.44%), *Affected Products* (94.44%), *Impact* (90.48%), *Mitigation* (95.24%), and *Solution* (87.30%). GPT-3.5 performs poorly in identifying *Root Cause*, *Attack Vector*, and *Solution*.

The poor performance of GPT-3.5 in identifying *Root Cause* is due to its inability to recognize underlying root causes (a high false negative rate, accounting for 80% of errors). For example, in a *Jenkins* security advisory, the vulnerability description states: "*Sandbox protection in the Script Security and Pipeline: Groovy Plugins could be circumvented through methods supporting type casts and type coercion.*" The statement describes the root cause of the vulnerability (i.e., Type casting and type coercion functionalities led to the bypassing of sandbox protection). GPT-3.5 struggles to accurately identify such root causes.

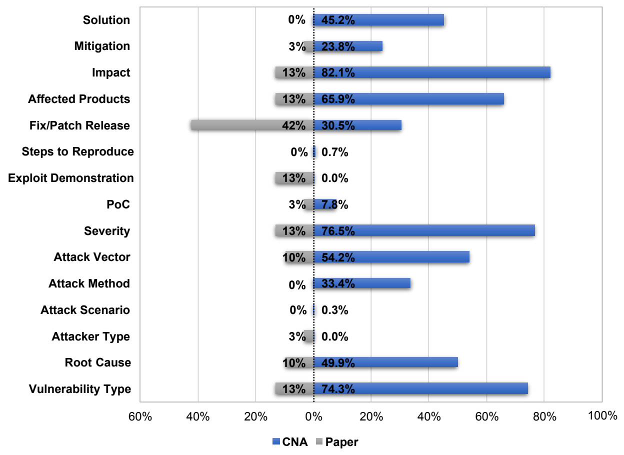


Fig. 4: Comparison of CNAs' Attention to Different Vulnerability Aspects.

Figure 4 shows the proportion of each aspect in the security bulletins we collected. In industrial practice, vulnerability *Impact* (82.1%) is the most common, indicating that CNAs prioritize the consequence following vulnerability discovery. This is followed by *Severity* (76.5%), *Vulnerability Type* (74.3%), and *Affected Product* (66%). The majority of CNAs prioritize *Severity* in their security bulletins/advisories, we find only five CNAs do not contain severity information. In contrast, security advisories from product companies (such as *schneider*, *vmware*, *dell*, *adobe*, and *apple*) typically inform users about affected products. *Attack Vector* (54.2%) and *Root Cause* (49.9%) account for about half of all security advisories. *Solution* and *Mitigation* measures represent the response strategies implemented post-vulnerability disclosure, accounting for 45.2% and 23.8%, respectively. Only three CNAs (i.e., *samsung*, *apple*, and @*huntrdev*) rarely address these two aspects. Further examination of the bulletins shows *samsung* and *apple* mainly focus on the product itself (since they are commercial product companies), while @*huntrdev* prioritizes technical details, such as *Root Cause* and *PoC* (Because it is a bug bounty platform and encourage submitters delve into the technical details of the vulnerability and should provide PoC).

Attack methods (33.4%) represent a small proportion, while PoC (8%) receives limited attention from CNAs, resulting in an overall small proportion. Similarly, vulnerability *Exploit Demonstration*, *Steps to Reproduce*, *Attack Scenario*, and *Attacker Type* are addressed by only a few security advisories, making up a negligible share of the total.

**Finding 1:** Industrial practice primarily emphasizes post-disclosure aspects of vulnerabilities, notably *Impac* (i.e., *Impact* and *Affected Product*), *Severity*, and *Measures*, while largely overlooking *Attacker Type*, *Attack Scenario*, and details on *Vulnerability Validation & Exploitation*.

## *B. RQ 2: What aspects of vulnerability are the focus of academic research?*

Our initial search yields a total of 981 papers. The specific numbers for each venue can be seen in the "Initial Retrieval" column of Table IV. Finally, we identify 31 papers from the premier publication venues, as shown in the "Retained Papers" column of Table IV. Table V presents the relevant research papers focusing on different vulnerability aspects and shows the number of papers for each vulnerability aspect. According to our summary, no articles address *Attack Scenario*, *Attack Method*, *Steps to Reproduce*, and *Solution*.

TABLE IV: Number of Papers Retrieved from Each Venue

| Venues | Initial Retrieval | Retained Papers |
|---|---|---|
| ICSE | 80 | 3 |
| ESEC/FSE | 47 | 3 |
| ASE | 36 | 3 |
| ISSTA | 65 | 3 |
| MSR | 32 | 1 |
| TSE | 59 | 4 |
| TOSEM | 17 | 3 |
| USENIX | 272 | 7 |
| CCS | 99 | 2 |
| NDSS | 125 | 1 |
| S&P | 149 | 1 |
| Total | 981 | 31 |

*1)* ***Missing Aspects****:* An examination of collected papers reveals an absence of research on *Attack Scenario*, *Attack Method*, *Steps to Reproduce*, and *Solution*. *Attack Scenario* and *Attack Method* typically involve complex technical details. The technical complexity makes it extremely challenging to describe the attack scenarios for each vulnerability. Different types of vulnerabilities (e.g., buffer overflows, SQL injection, and cross-site scripting) require different attack techniques and strategies. Therefore, systematically covering possible attack scenarios and methods is a difficult task. *Steps to Reproduce* requires detailed and comprehensive descriptions, covering all necessary prerequisites and specific operations. Due to the diversity of vulnerabilities and the complexity of environments, developing a universal set of reproduction steps is highly challenging. Many studies focus more on the discovery and description of vulnerabilities while lacking in-depth exploration of specific solutions. This can be attributed to the diversity and complexity of solutions, as well as the practical challenges in different environments.

TABLE V: Summary of Vulnerability Aspects in Published Papers

| Category | Aspects | Papers |
|---|---|---|
| Vulnerability Type | Vulnerability Type (4)* | [25] [31] [32] [33] |
| Root Cause | Root Cause (3) | [31] [34] [33] |
| Attack Model | Attacker Type (1) | [33] |
| | Attack Scenario (0) | – |
| | Attack Method (0) | – |
| | Attack Vector (3) | [31] [34] [33] |
| Severity | Severity (4) | [35] [36] [28] [37] |
| Verification** | PoC (1) | [38] |
| | Exploit Demonstration (4) | [39] [40] [41] [42] |
| | Steps to Reproduce (0) | – |
| Fix/Patch Release | Fix/Patch Release (13) | [43] [26] [44] [45] [46] [47] [48] [49] [50] [44] [51] [52] [53] |
| Impact | Affected Product (4) | [30] [54] [33] [14] |
| | Impact (4) | [31] [34] [33] [27] |
| Measure | Mitigation (1) | [27] |
| | Solution (0) | – |

* The numbers in parentheses indicate the number of papers addressing the aspect.
** Verification denotes *Vulnerability Verification & Demonstration*.

*2)* ***Emphasized Aspects****:* We find that *Fix/Patch Release* is the most prominent aspect, with 13 papers (42%) conducting research around this aspect. These include: (1) Patch Generation. Existing patch generation research has primarily concentrated on generating patches for vulnerabilities in Android kernels, Android mobile applications, and OT device control applications. Xu et al. [48] conduct automatic patch generation for Android kernel CVEs and develop the VULMET tool. VULMET can generate correct patches from official patches for 55 CVEs with low-performance overhead. Duan et al. [55] propose the OSSPatcher which can generate 675 functional-level patches for 39 OSS projects and 1,000 Android applications that use vulnerable versions. For Operational Technology (OT) devices, Rajput et al. [49] develop the ICSPatch framework, which uses data dependency graphs (DDGs). (2) Patch Porting. The research on patch porting is primarily focused on Linux and web applications. Shariffdeen et al. [46] firstly propose the patch transplantation problem and introduce PatchWeave, a technique that automatically extracts patches and transplants them to another target program that fails on the same test case. Subsequently, they propose a technique (i.e., FixMorph) for automatically porting patches from the Linux mainline to older stable versions [44]. Shi et al. [47] design the SKYPORT framework to backport injection vulnerability patches for web applications. (3) Patch Search/Tracing. Tan et al. [50] develop PatchScout, which transforms the patch-searching problem into a ranking problem. By leveraging the correlation between vulnerabilities and code commits, PatchScout helps identify security patches for OSS vulnerabilities in code repositories. Xu et al. [26] develop Tracer, the first automated method for tracking OSS vulnerability patches from multiple knowledge sources. (4) Vulnerability Repair. Automated patch fixing primarily focuses on the C and Java programming languages and operates at the function level. Chen et al. propose VRepair [52], a vulnerability repair method for C functions based on transfer learning. Compared to VRepair, VulRepair achieves higher perfect prediction rates, making it more effective in generating correct patches [43]. Chi et al. [53] develop SeqTrans, which uses a Neural Machine Translation (NMT) model to learn and generalize common patterns from historical data for vulnerability fixing, aiming to improve the accuracy and generalizability of generated patches. To address the overfitting problem in vulnerability repair, Gao et al. [45] propose ExtractFix, which can repair program vulnerabilities without requiring extensive test suites. Given the popularity of large language models (LLMs) in software engineering tasks, Wu et al. [51] are the first to use LLMs for Java vulnerability repair.

*3) **Slightly Delved Aspects**:* Three studies are capable of generating information on multiple aspects of vulnerabilities [31], [33], [34]. Yitagesu et al. [34] propose an unsupervised method to label and extract important vulnerability concepts (i.e., *Root Cause*, *Attack Vector*, and *Impact*). Building on this, they expand the extracted vulnerability aspects to include *Vulnerability Type*, *Affected Product*, *Root Cause*, *Attacker Type*, *Impact*, and *Attack Vector* [33]. Sun et al. [31] develop a framework that improves the reliability of vulnerability alerts and their acceptance by users by identifying explainable aspects such as *Vulnerability Type*, *Root Cause*, *Attack Vector*, and *Impact*. Our *Root Cause*, *Attack Vector*, and *Impact* aspects are mainly derived from the three papers mentioned above.

*Vulnerability Type*. The few studies on vulnerability types in academia are almost inseparable from CWE [31], [33]. Pan et al. [25] propose TreeVul, which can predict vulnerability category information at the commit level. Fu et al. introduce a new data grouping approach based on CWE abstract types and a teacher-student learning framework to overcome the data imbalance issue in the software vulnerability classification task [32].

*Severity*. The assessment of vulnerability severity in academia predominantly relies on CVSS indicators and scores [4]. To perform automated software vulnerability assessment with concept drift using vulnerability descriptions, Le et al. [28] propose a systematic approach that combines both character and word features to automate software vulnerability assessment using vulnerability descriptions. In addition, they propose the first code-commit-level evaluation solution, DeepCVA [36], which can perform multi-class prediction for the seven CVSS metrics. Similarly, Li et al. introduce the Context-aware, Graph-based, Commit-level Vulnerability Detection and Assessment Model (VDA) [35], which outperforms DeepCVA in its evaluation. Wu et al. [37] propose an OS-aware vulnerability prioritization (namely DIFFCVSS) that conducts different severity analyses for vulnerabilities (for Linux).

*Exploit Demonstration.* Exploit demonstration can be categorized into automated exploit generation and exploit migration. Park et al. [39] introduce FUGIO, the first automatic exploit generation (AEG) tool for PHP Object Injection (POI) vulnerabilities. Wang et al. [41] propose a new solution recovery method for heap-based vulnerabilities. Wu et al. [40] introduce KEPLER, a framework that utilizes a novel "one-shot" exploitation technique to transform control flow hijacking primitives into classical stack overflow vulnerabilities (for Linux kernel). In contrast to automatic exploit generation [40], Jiang et al. [42] focus on automatic exploit migration (AEM) to evaluate the cross-version exploitability of Linux kernel vulnerabilities.

*Impact*. The *Impact* aspects are divided into upstream impact (*Affected Product*) and downstream impact (*Impact*). Some papers that discuss *Impact* do not solely focus on itself but also cover other related aspects. Shi et al. [54] propose a vulnerability-centric approach for precise (un)affected version analysis for web vulnerabilities, and Bao et al. [30] automatically identify product versions affected by vulnerabilities through commit information. Dong et al. develop VIEM [14] to extract vulnerable software names and versions from unstructured text.

*4) **Sparsely Investigated Aspects**:* Only one paper focuses on *Attacker Type* by extracting information about the types of attackers from textual vulnerability descriptions [34]. Yang et al. propose a new directed differential fuzzing solution, 1dFuzz, which can effectively reproduce 1-day vulnerabilities (for *PoC*) [38]. There is also limited research on vulnerability *Mitigation*, with only one notable study. Wang et al. [27] propose a npm ecosystem-level technique called Plumber, which generates feasible remediation strategies to boost the propagation of vulnerability fixes. The Plumber can provide customized remediation suggestions along with vulnerability impact analysis.

**Finding 2:** Academic research on vulnerabilities primarily focuses on *Fix/Patch Release*. Aspects such as *Attack Scenario*, *Attack Method*, *Steps to Reproduce*, and *Solution* are largely missing due to their technical complexity and diverse environments.

### *C. RQ 3: What are the similarities and differences between the aspects that industry and academia focus on in security vulnerability?*

The insights gained from analyzing RQ 1 and RQ 2 empower us to solve the third question. Table VI presents the aspects of vulnerabilities that both industry and academia are interested in.

TABLE VI: Comparison between Industry and Acadimia

| **Industry** | **Acadimia** | |
|---|---|---|
| | **Focus** | **Not Focus** |
| **Focus** | Vulnerability Type<br>Root Cause<br>Attack Vector<br>Severity<br>Fix/Patch Release<br>Affected Product<br>Impact | Attack Method<br>Mitigation<br>Solution |
| **Not Focus** | Attacker Type<br>Exploit Demonstration | Attack Scenario<br>PoC<br>Steps to Reproduce |

*1) **Shared Aspects of Concern in Academia and Industry**:* Academic research on vulnerabilities benefits from industrial data. Academic studies on *Vulnerability Type* (Common Weakness Enumeration, CWE), *Root Cause*, *Attack Vector*, and *Impact* primarily rely on vulnerability descriptions. Similarly, *Severity* assessments depend on CVSS scores. For the above aspects, when information is incomplete or ambiguous in security bulletins, academic methodologies can complement industrial practices by generating missing details. For the industry, there is some focus on *Fix/Patch Release* (30.5%). Some academic work focuses on patch automation and vulnerability repair. Although these studies currently address specific scenarios, they provide new methods and tools that can improve vulnerability management and remediation. In addition, both academia and industry place a high value on identifying and communicating the *Affected Product*. The academic community develop methods for the analysis of impacted versions, while security bulletins from the industry provide detailed listings of specific product versions and configurations that are affected. Academic studies can offer data-driven insights into vulnerability trends and patterns, helping the industry to better predict and mitigate future risks.

*2) **Neglected Aspects in Academia and Industry**:* Both academia and industry rarely focus on the *Attacker Type*. However, vulnerability descriptions may implicitly contain information about the types of attackers. Existing academic research can generate attacker type information, which can complement the lack of such information in industrial data. In the future, there may be more focus and investment in the aspect. Both industry and academia also pay little attention to *Attack Scenario*, *PoC*, and *Steps to Reproduce*. For the industry, providing comprehensive descriptions of these aspects requires significant resources, which also hinders in-depth research in academia. The industry avoids detailed descriptions of the above aspects to minimize the risk of attacks because they often include technical information that could be exploited. In the future, there may be more focus and investment in these aspects, with crucial attention to balancing information disclosure and risk management.

*3) **Distinct Aspects of Interest in Academia and Industry**:* Below we describe some of the gaps we find between industry and academia. 33.4% of industrial security advisories mention *Attack Method* to provide specific protection guidelines. In contrast, the academic community lacks dedicated research on attack methods. This can be attributed to two main factors: (1) a lack of systematic data to support such research and (2) a focus on studying new attack methods for recently discovered vulnerabilities rather than existing ones.

Many CNAs typically omit details regarding their verification and exploitation. Several academic papers concentrate on the automated generation of exploit code, addressing a notable gap in industrial focus. However, due to the complexity and potential risks involved, current research in this aspect is still in its early stages. We categorize *Measure* into *Mitigation* and *Solution*. CNAs concentrate on the mitigations (23.8%) or solutions (45.2%) implemented after vulnerability discovery. Academic research pays little attention to measures, with only one related work being identified. In the future, these two aspects may benefit from more in-depth research building on existing work in vulnerability remediation and automated patch generation.

**Finding 3:** Both academia and industry prioritize *Vulnerability Type*, *Impact*, and *Affected Product*. However, both of them pay less attention to *Attacker Type*, *Attack Scenario*, *PoC*, and *Steps to Reproduce*.

## V. Discussion

### *A. Collaboration and Directions in Vulnerability Management*

It is beneficial for both industry and academia if the academic community follows all aspects of industrial practices in vulnerability management. Current academic research on vulnerabilities can alleviate manual work by automating the organization and summarization of descriptive information, generating vulnerability types, identifying root causes, assessing severity, determining attacker types, and listing affected products. By organizing vulnerability information from various sources, the quality of industrial vulnerability information can be significantly improved [4], [56]. For academia, these aspects currently contain a large amount of data in the industry, which can stimulate the motivation for innovation using new technologies. Additionally, due to the lack of clarity in attacker-type information in industrial bulletins, the academic community can focus on generating this information. These overlapping aspects are highly suitable for collaboration, especially when guided by well-established standards, such as those implemented by the NVD. This alignment is evident and widely recognized.

For the *Solution* that the industry is more focused on, academia should also follow the priority. This alignment is beneficial for academia. Currently, the sources of vulnerability information in academia are relatively limited, with only a few studies incorporating security bulletins/advisories from a few CNAs. There is already work focused on vulnerability *Mitigation* [27], which can generate mitigation recommendations. Combining this with the current academic interest in patch generation and vulnerability repair, a future research direction could be the generation of comprehensive solutions through patching and vulnerability remediation, which would benefit both academia and industry.

Both industry and academia do not focus on several aspects, such as *Attack Scenario* and *Steps to Reproduce*. The industry should prioritize *Attack Scenario*, as they are simpler and more feasible to describe compared to *Steps to Reproduce*. Cheng et al. [57] constructed a knowledge graph for vulnerability information, which can be used to search for related vulnerabilities. They used BERT to cover various aspects, including *Affected Product*, *Attacker Type*, *Vulnerability Location* (in specific code files), *Exploitation Condition* (corresponding to *Attack Method*), and *Occurrence Scenario* (corresponding to *Attack Scenario*). This research heavily relies on industrial data, and their findings show that effective descriptions of attack scenarios are very limited. Industry should place greater emphasis on the collection and analysis of attack scenario information. In the future, this aspect of vulnerabilities holds the potential for breakthrough advancements, benefiting both industry and academia.

### B. Threats to Validity

*Internal Validity.* There are some threats associated with collecting CNA heading data and mapping rules. To mitigate this issue, we invest manual effort in developing unique rules for each CNA website. We calculate the Cohen's Kappa value between the two authors from 0.57 to 1, indicating consistency in aspect extract for heading. Future enhancements involve eliminating manual rule design and leveraging diverse NLP technologies for automated aspect summarization in long web pages. We acknowledge the limitation when using prompt templates for vulnerability aspect extraction. Additionally, the accuracy calculation for ChatGPT's extractions is limited by the use of only 10% of the web pages. To mitigate this, we employ a random selection process. While it is true that 10% is a commonly used ratio for machine learning test sets, a larger sample size could have yielded more robust accuracy estimates.

*External Validity.* We have prioritized selecting top-tier conferences and journals that are widely recognized for their influence and the quality of their content [58] . These venues are considered to be at the forefront of academic research, and their inclusion provides a strong base for our study. We understand that the scope of our current work may not encompass the full breadth of the field. By prioritizing top-tier venues, we may have inadvertently excluded relevant studies from less prominent sources, such as workshops, preprint repositories, or industry reports, which could limit the generalizability of our findings. Additionally, our focus on the 2018-2023 timeframe may miss foundational work published earlier or emerging trends not yet widely disseminated. Furthermore, the emphasis on mainstream research directions may overlook niche or interdisciplinary contributions that could provide unique insights, particularly at the intersection of vulnerability detection and other fields such as machine learning or software engineering. To mitigate these limitations, future work could expand the scope of literature reviews to include a wider range of sources, time periods, and interdisciplinary perspectives. CNA web page selection relies on the official URLs for vulnerability information disclosure from the top 50 CNAs with the highest CVE ID assignment frequency. This method might have overlooked relevant websites. In future iterations, incorporating additional web page discovery techniques could broaden the scope of the analysis.

## VI. Related Work

### A. Vulnerability Key Aspects and Extraction

Some researchers have conducted research based on key aspects of vulnerability information. Guo et al. [12] proposed a neural network-based method to predict the critical information missing from vulnerabilities. Sun et al. [59] employed three NER methods to extract vulnerable products, versions, components, and vulnerability types while leveraging Q&A techniques for parsing free-form phrases related to root causes, attack vectors, and impacts. Guo et al. [13] obtained a large number of semi-structured vulnerability reports from security databases such as CVE, Security Focus, and IBM X-Force Exchange. They introduced a customized NER method based on deep neural networks, focusing on the extraction of six key aspects. Sun et al. [24] proposed a method using vulnerability elements (i.e., Attacker, Impact, Vector, Situation, Cause, Product, Version) to evaluate the performance of six CVSS indicators (i.e., Access Vector, Access Complexity, Authentication, Confidentiality Impact, Integrity Impact, and Availability Impact) to assess the severity of the vulnerability.

Existing studies for extracting aspects from vulnerability descriptions are effective for standardized or concise descriptions but perform worse in full-text web pages. We employ a rule-matching method for structured web page information. Recognizing the limitations of established methods for handling our larger website-based text corpus, we opt for GPT-3.5 inspired by the recent advancements in LLMs. We design Q&A prompts querying GPT-3.5 for the presence of specific vulnerability aspects.

### B. Industrial Data and Academic Research in Vulnerability Management

Le et al. [4] conducted a survey on data-driven software vulnerability assessment. They analyzed the industry data sources used in existing data-driven work, finding that most studies rely on vulnerability descriptions of NVD. Only a few studies incorporate data from project security reports.

Massacci et al. [60] focused on the quality of vulnerability databases, analyzing and comparing different security metric papers and relevant data sources. Their research delved into the discrepancies between the available data and academic studies. VulData7 [61] automatically collected vulnerability instances from software archives, including vulnerability report data (i.e., description, CVE number, CWE number, CVSS severity score, and others), vulnerable code instance (list of versions), and corresponding patches (list of fixing commits) and files (before and after fix). It can extract and link information from the related software archives (through Git and NVD reports) to create a dataset that is continuously updated with the latest information available. This resource was invaluable for software developers and researchers.

OVANA [16] leveraged machine learning (ML) and natural language processing (NLP) techniques to analyze the quality of information in vulnerability databases. OVANA provided information that should be included in structured fields to enhance the uniqueness of vulnerability entries between different vulnerability entries. Okutan et al. [56] used ML, NLP, and information-theoretic techniques to automatically characterize CVEs, providing a detailed vulnerability characterization using 28 attributes across five domains. KEKÜL et al. [62] systematically reviewed the most recent and accessible vulnerability databases widely used in the research. They analyzed the strengths and weaknesses of each to help researchers choose the most suitable database.

In contrast to the previous work which utilized data from diverse vulnerability databases, our analysis primarily leverages security bulletins issued by CNAs. Additionally, we provide an overview of the aspects of vulnerabilities currently receiving attention in academia.

## VII. Conclusion

Vulnerabilities are a common concern in the software and security fields. Note some studies rely on the NVD database; however, the existing data has several limitations. In this work, we first collected 50,254 security advisories from 36 CNA websites as an open-source data set and identified 15 key vulnerability aspects. We then conducted a literature review to investigate the academic focus on our proposed vulnerability aspects. We compared industrial practices with academic research for these aspects, revealing disparities between them. We analyzed the concerns and differences between them in each aspect. Our findings indicate a lack of research on attack scenarios and methods in academia. Researchers can extract data regarding aspects of security advisories for their studies. The industry places greater emphasis on solutions, in addition to the seven aspects outlined by NVD. However, some aspects of security advisories are still absent. In the future, researchers could focus on enhancing the extraction and integration of security advisory information in industry practice.

## Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2024YFB4506400) and Ningbo Natural Science Foundation (No. 2023J292).

## References

[1] G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang, "Software vulnerability detection using deep neural networks: A survey," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1825–1848, 2020.

[2] A. Anwar, A. Abusnaina, S. Chen, F. Li, and D. Mohaisen, "Cleaning the nvd: Comprehensive quality assessment, improvements, and analyses," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 4255–4269, 2022.

[3] R. Aranovich, M. Wu, D. Yu, K. Katsy, B. Ahmadnia, M. Bishop, V. Filkov, and K. Sagae, "Beyond nvd: Cybersecurity meets the semantic web." in *Proceedings of the 2021 New Security Paradigms Workshop*, ser. NSPW '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 59–69.

[4] T. H. M. Le, H. Chen, and M. A. Babar, "A survey on data-driven software vulnerability assessment and prioritization," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–39, dec 2022.

[5] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 105–114.

[6] H. Chen, R. Liu, N. Park, and V. Subrahmanian, "Using twitter to predict when vulnerabilities will be exploited," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 3143–3152.

[7] R. Sharma, R. Sibal, and S. Sabharwal, "Software vulnerability prioritization using vulnerability description," *International Journal of System Assurance Engineering and Management*, vol. 12, no. 1, pp. 58–64, Feb 2021.

[8] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 125–136.

[9] E. Aghaei, W. Shadid, and E. Al-Shaer, *ThreatZoom: Hierarchical Neural Network for CVEs to CWEs Classification*. Springer International Publishing, 2020, p. 23–41.

[10] H. Venter, J. Eloff, and Y. Li, "Standardising vulnerability categories," *Computers & Security*, vol. 27, no. 3, pp. 71–83, 2008.

[11] S. Neuhaus and T. Zimmermann, "Security trend analysis with cve topic models," in *2010 IEEE 21st International Symposium on Software Reliability Engineering*, 2010, pp. 111–120.

[12] H. Guo, S. Chen, Z. Xing, X. Li, Y. Bai, and J. Sun, "Detecting and augmenting missing key aspects in vulnerability descriptions," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 3, apr 2022.

[13] H. Guo, Z. Xing, S. Chen, X. Li, Y. Bai, and H. Zhang, "Key aspects augmentation of vulnerability description based on multiple security databases," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2021, pp. 1020–1025.

[14] Y. Dong, W. Guo, Y. Chen, X. Xing, Y. Zhang, and G. Wang, "Towards the detection of inconsistencies in public security vulnerability reports," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 869–885.

[15] D. Mu, A. Cuevas, L. Yang, H. Hu, X. Xing, B. Mao, and G. Wang, "Understanding the reproducibility of crowd-reported security vulnerabilities," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 919–936.

[16] P. Kuehn, M. Bayer, M. Wendelborn, and C. Reuter, "Ovana: An approach to analyze and improve the information quality of vulnerability databases," in *ARES 2021: The 16th International Conference on Availability, Reliability and Security.*, ser. ARES '21. New York, NY, USA: Association for Computing Machinery, 2021.

[17] J. Evans. Mitre key details phrasing. http://cveproject.github.io/docs/content/key-details-phrasing.pdf. [Online]. Available: http://cveproject.github.io/docs/content/key-details-phrasing.pdf

[18] (2019) Cwe - common weakness enumeration. [Online]. Available: https://cwe.mitre.org/

[19] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahashi, "Automation of vulnerability classification from its description using machine learning," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–7.

[20] (2024) Structured page example - siemens security advisory. [Online]. Available: https://cert-portal.siemens.com/productcert/html/ssa-999588.html

[21] E. Maurice. (2020) Unstructured page example - oracle. [Online]. Available: https://blogs.oracle.com/security/post/security-alert-cve-2020-14750-released

[22] Y. Tan, D. Min, Y. Li, W. Li, N. Hu, Y. Chen, and G. Qi, "Can chatgpt replace traditional kbqa models? an in-depth analysis of the question answering performance of the gpt llm family," in *International Semantic Web Conference*. Springer, 2023, pp. 348–367.

[23] K. S. Kalyan, "A survey of gpt-3 family large language models including chatgpt and gpt-4," *Natural Language Processing Journal*, vol. 6, p. 100048, 2024.

[24] X. Sun, Z. Ye, L. Bo, X. Wu, Y. Wei, T. Zhang, and B. Li, "Automatic software vulnerability assessment by extracting vulnerability elements," *Journal of Systems and Software*, vol. 204, p. 111790, 2023.

[25] S. Pan, L. Bao, X. Xia, D. Lo, and S. Li, "Fine-grained commit-level vulnerability type prediction by cwe tree structure," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 957–969.

[26] C. Xu, B. Chen, C. Lu, K. Huang, X. Peng, and Y. Liu, "Tracking patches for open source software vulnerabilities," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 860–871.

[27] Y. Wang, P. Sun, L. Pei, Y. Yu, C. Xu, S.-C. Cheung, H. Yu, and Z. Zhu, "Plumber: Boosting the propagation of vulnerability fixes in the npm ecosystem," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3155–3181, 2023.

[28] T. H. M. Le, B. Sabir, and M. A. Babar, "Automated software vulnerability assessment with concept drift," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 371–382.

[29] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey," *ACM Comput. Surv.*, vol. 50, no. 4, Aug. 2017.

[30] L. Bao, X. Xia, A. E. Hassan, and X. Yang, "V-szz: automatic identification of version ranges affected by cve vulnerabilities," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2352–2364.

[31] J. Sun, Z. Xing, Q. Lu, X. Xu, L. Zhu, T. Hoang, and D. Zhao, "Silent vulnerable dependency alert prediction with vulnerability key aspect explanation," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 970–982.

[32] M. Fu, V. Nguyen, C. K. Tantithamthavorn, T. Le, and D. Phung, "Vulexplainer: A transformer-based hierarchical distillation for explaining vulnerability types," *IEEE Transactions on Software Engineering*, vol. 49, no. 10, pp. 4550–4565, 2023.

[33] S. Yitagesu, Z. Xing, X. Zhang, Z. Feng, X. Li, and L. Han, "Extraction of phrase-based concepts in vulnerability descriptions through unsupervised labeling," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 5, Jul. 2023.

[34] S. Yitagesu, Z. Xing, X. Zhang, Z. Feng, X. Li, and L. Han, "Unsupervised labeling and extraction of phrase-based concepts in vulnerability descriptions," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 943–954.

[35] Y. Li, A. Yadavally, J. Zhang, S. Wang, and T. N. Nguyen, "Commit-level, neural vulnerability detection and assessment," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 1024–1036.

[36] T. H. Minh Le, D. Hin, R. Croft, and M. Ali Babar, "Deepcva: Automated commit-level vulnerability assessment with deep multi-task learning," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 717–729.

[37] Q. Wu, Y. Xiao, X. Liao, and K. Lu, "OS-Aware vulnerability prioritization via differential severity analysis," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 395–412.

[38] S. Yang, Y. He, K. Chen, Z. Ma, X. Luo, Y. Xie, J. Chen, and C. Zhang, "1dfuzz: Reproduce 1-day vulnerabilities with directed differential fuzzing," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 867–879.

[39] S. Park, D. Kim, S. Jana, and S. Son, "FUGIO: Automatic exploit generation for PHP object injection vulnerabilities," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 197–214.

[40] W. Wu, Y. Chen, X. Xing, and W. Zou, "KEPLER: Facilitating control-flow hijacking primitive evaluation for linux kernel vulnerabilities," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1187–1204.

[41] Y. Wang, C. Zhang, X. Xiang, Z. Zhao, W. Li, X. Gong, B. Liu, K. Chen, and W. Zou, "Revery: From proof-of-concept to exploitable," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1914–1927.

[42] Z. Jiang, Y. Zhang, J. Xu, X. Sun, Z. Liu, and M. Yang, "Aem: Facilitating cross-version exploitability assessment of linux kernel vulnerabilities," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 2122–2137.

[43] M. Fu, C. Tantithamthavorn, T. Le, V. Nguyen, and D. Phung, "Vulrepair: a t5-based automated software vulnerability repair," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 935–947.

[44] R. Shariffdeen, X. Gao, G. J. Duck, S. H. Tan, J. Lawall, and A. Roychoudhury, "Automated patch backporting in linux (experience paper)," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 633–645.

[45] X. Gao, B. Wang, G. J. Duck, R. Ji, Y. Xiong, and A. Roychoudhury, "Beyond tests: Program vulnerability repair via crash constraint extraction," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 2, Feb. 2021.

[46] R. S. Shariffdeen, S. H. Tan, M. Gao, and A. Roychoudhury, "Automated patch transplantation," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 1, Dec. 2021.

[47] Y. Shi, Y. Zhang, T. Luo, X. Mao, Y. Cao, Z. Wang, Y. Zhao, Z. Huang, and M. Yang, "Backporting security patches of web applications: A prototype design and implementation on injection vulnerability patches," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1993–2010.

[48] Z. Xu, Y. Zhang, L. Zheng, L. Xia, C. Bao, Z. Wang, and Y. Liu, "Automatic hot patch generation for android kernels," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2397–2414.

[49] P. H. N. Rajput, C. Doumanidis, and M. Maniatakos, "ICSPatch: Automated vulnerability localization and Non-Intrusive hotpatching in industrial control systems using data dependence graphs," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 6861–6876.

[50] X. Tan, Y. Zhang, C. Mi, J. Cao, K. Sun, Y. Lin, and M. Yang, "Locating the security patches for disclosed oss vulnerabilities with vulnerability-commit correlation ranking," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3282–3299.

[51] Y. Wu, N. Jiang, H. V. Pham, T. Lutellier, J. Davis, L. Tan, P. Babkin, and S. Shah, "How effective are neural networks for fixing security vulnerabilities," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 1282–1294.

[52] Z. Chen, S. Kommrusch, and M. Monperrus, "Neural transfer learning for repairing security vulnerabilities in c code," *IEEE Transactions on Software Engineering*, vol. 49, no. 1, pp. 147–165, 2023.

[53] J. Chi, Y. Qu, T. Liu, Q. Zheng, and H. Yin, "Seqtrans: Automatic vulnerability fix via sequence to sequence learning," *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 564–585, 2023.

[54] Y. Shi, Y. Zhang, T. Luo, X. Mao, and M. Yang, "Precise (un)affected version analysis for web vulnerabilities," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023.

[55] R. Duan, A. Bijlani, Y. Ji, O. Alrawi, Y. Xiong, M. Ike, B. Saltaformaggio, and W. Lee, "Automating patching of vulnerable open-source software versions in application binaries." in *NDSS*, 2019.

[56] A. Okutan, P. Mell, M. Mirakhorli, I. Khokhlov, J. C. S. Santos, D. Gonzalez, and S. Simmons, "Empirical validation of automated vulnerability curation and characterization," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3241–3260, 2023.

[57] X. Cheng, X. Sun, L. Bo, and Y. Wei, "Kvs: a tool for knowledge-driven vulnerability searching," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1731–1735.

[58] X. Hu, X. Xia, D. Lo, Z. Wan, Q. Chen, and T. Zimmermann, "Practitioners' expectations on automated code comment generation," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1693–1705. [Online]. Available: https://doi.org/10.1145/3510003.3510152

[59] J. Sun, Z. Xing, X. Xia, Q. Lu, X. Xu, and L. Zhu, "Aspect-level information discrepancies across heterogeneous vulnerability reports: Severity, types and detection methods," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 2, dec 2023.

[60] F. Massacci and V. H. Nguyen, "Which is the right source for vulnerability studies? an empirical analysis on mozilla firefox," in *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, ser. MetriSec '10. New York, NY, USA: Association for Computing Machinery, 2010.

[61] M. Jimenez, Y. Le Traon, and M. Papadakis, "[engineering paper] enabling the continuous analysis of security vulnerabilities with vuldata7," in *2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2018, pp. 56–61.

[62] H. KEKÜL, B. ERGEN, and H. ARSLAN, "Comparison and analysis of software vulnerability databases," *International Journal of Engineering and Manufacturing*, vol. 12, no. 4, p. 1, 2022.