The Flaw Within: Identifying CVSS Score Discrepancies in the NVD

Siqi Zhang*, Minjie Cai*, Mengyuan Zhang*, Lianying Zhao†, Xavier de Carné de Carnavalet*
*The Hong Kong Polytechnic University, Hong Kong, China
*siqi0510.zhang@connect.polyu.hk, {minjie.cai,mengyuan.zhang,xdecarne}@polyu.edu.hk
†Carleton University, Ottawa, Canada, lianying.zhao@carleton.ca

***Abstract*—Cloud security frameworks, like OpenSCAP, rely on vulnerability databases such as the National Vulnerability Database (NVD) to assess threats, ensure compliance, and manage patches efficiently. However, despite their popularity, vulnerability databases are not exempt from errors. Prior research showed inconsistencies between multiple databases, as well as incorrect software or vendor names, and publication dates. In this study, we discovered and proposed a systematic approach to detect a new form of inconsistency whereby entries with identical or semantically similar vulnerability descriptions are assigned distanced scores, which can skew risk assessments, and potentially misguide mitigation strategies. Our analysis identified 12,866 entries suffering from such inconsistencies, highlighting the most error-prone Common Vulnerability Scoring System (CVSS) metrics and vulnerability types, as well as the observed score deviation. We believe our study can bring this inconsistency issue to the community's attention and pave the way for further investigation thereof.**

***Index Terms*—NVD, Security assessment, CVSS Machine Learning, Vulnerability analysis, Inconsistency in NVD**

I. Introduction

Publicly maintained vulnerability databases, such as the National Vulnerability Database (NVD [1]) are widely referenced in both industry for risk assessment and patch management [2], [3], and in academia for empirical studies related to software vulnerabilities, e.g. [4], [5], [6]. In particular, the Common Vulnerability Scoring System (CVSS) provides a way to quantify the severity level of vulnerabilities into a score via various metrics, thus enabling comparison and ranking of multiple vulnerabilities [7], [8]. Such practices assume and depend on the relative correctness and consistency of the CVSS metrics.

Previous works have already hinted at inconsistencies in these vulnerability databases. For instance, vulnerable software versions affected by a vulnerability could be inconsistent across popular databases [9]. Within NVD itself, researchers have uncovered a range of reporting inconsistencies including incorrect labeling of software and vendor names across entries (sometimes due to typos), invalid publication dates [10], incomplete list of affected products and reference URLs [11].

In this paper, we bring attention to another form of inconsistency within a single vulnerability database involving the assigned severity score (CVSS base metrics) with respect to the free-form vulnerability description, which, to the best of our knowledge, has not been systematically studied by the community. Intuitively, vulnerabilities described in the same way should receive similar severity scores (if not the same). Indeed, as per the instructions for creating descriptions [12], a CVE description should encapsulate comprehensive details, uniquely distinguishing each vulnerability. Those utilizing the database depend on these descriptions not only to grasp the vulnerability's essence but also to correlate it with real-world instances for effective remediation and patch application. Analysts also rely on descriptions to assign CVSS base metrics and a corresponding score.

However, our preliminary study shows that 4.4% of CVE entries in NVD that share the same description with another CVE are scored differently.[1] For instance, the three entries CVE-2017-(5807, 5808, 5809) share exactly the same textual description yet received significantly different scores, ranging from medium to critical; see an illustration in Figure 1. In addition to identical CVE descriptions, we also noticed CVE entries with very similar descriptions but, unexpectedly, different CVSS base metrics. Such discrepancies harm the credibility of the database and the accuracy of any work derived from it. The security team of an enterprise trying to prioritize patching of severe vulnerabilities can be misled into under-prioritizing a medium-severity vulnerability although it may exhibit the same characteristics as a critical vulnerability.

While identifying entries with identical descriptions is easy, the true extent of such inconsistencies may only be revealed after finding all descriptions that carry the same meaning. In turn, simple textual solutions (e.g., edit distance) cannot cope with the richness of natural language. Conversely, generic Natural Language Processing (NLP) techniques are not tailored to a cybersecurity-related vocabulary (e.g., consider "attack" as a "cause of death" [13]).

In this work, we aim at detecting severity score inconsistencies between vulnerability entries with semantically similar descriptions at a large scale, study their nature and prevalence, and identify possible causes. We propose to apply a set of machine learning-based techniques to divide the vulnerability entries into semantically similar clusters. We argue that vulnerabilities within in each cluster should receive similar severity scores, default of which would indicate score inconsistency.

[1] Identical description percentage = $\sum_{i=1}^{k} E_i \Big/ \#\text{NVD\_Entries}$, where k is the number of clusters containing more than one identical description in the entire NVD database, E_i is the number of entries in the ith cluster.

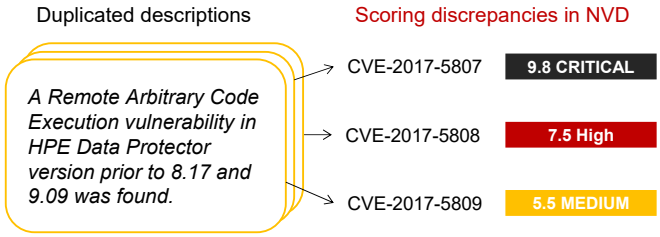


Fig. 1: Motivating Example

In the context of describing vulnerabilities, we define *semantic similarity* as the property when the descriptions of multiple vulnerabilities possess the same essential information sufficient to lead to the same severity score. This similarity naturally covers identical (exactly the same) descriptions.

We first transform the human-readable descriptions into machine-readable representations (features), based on multiple models, e.g., pre-trained BERT [14], TF-IDF [15], and customized vulnerability information fine-tuned word2vec [13]. Then, we combine clustering algorithms (K-Means clustering and Agglomerative clustering) to generate semantically similar clusters based on the obtained features. Clusters are further broken down by vulnerability type. We leverage *VIET* [13] to extract essential entities, e.g., privileges, from the descriptions. A cluster is deemed inconsistent when vulnerabilities therein that share identical entities are assigned different CVSS scores.

Our contributions are as follows:

1) We bring attention to the existence of severity scoring inconsistencies based on descriptions in the NVD.
2) Using clustering approaches and our adapted name entity recognition (NER) to process the vulnerability descriptions, we develop a framework that can automatically divide vulnerability entries into semantically similar clusters, which can serve as a foundation for further inconsistency analysis or security assessment.
3) We conduct a large-scale analysis of 124,034 CVE descriptions and accompanying CVSS v3 scores from the NVD with our framework, and found that 10.44% of vulnerability entries are affected by inconsistent metrics and/or descriptions with respect to other similar vulnerabilities.
4) We perform attribution analysis to identify error-prone CVSS metrics and have uncovered new types of clerical errors made by analysts while populating CVE entries, including mismatching textual attributes with CVSS metrics and copy-pasting oversights, and found that the privilege metric is associated with the most frequent discrepancies.

The remainder of this paper is organized as follows. Section II introduces necessary background information. Section III details the methodology of clustering and inconsistency detection. Section IV presents the experimental details, evaluation and results. Section V conducts a case study with further discussions. Section VI reviews the related work with Section VII concluding the paper.

II. Preliminaries

NVD [1] and CVE [16] are related public databases of software security vulnerabilities. Individuals or organizations report vulnerabilities to a CVE Program participant, who then reserves a CVE ID and provides details about the vulnerability, such as its type, triggering condition, affected versions, and a summary description [12]. Each CVE entry also has CVSS score to gauge its severity. While older entries are assigned with CVSS v2, the newer CVSS v3 metrics consider 8 indicators, namely: attack vector (AV), attack complexity (AC), privileges required (PR), user interaction (UI), scope (S), confidentiality (C), integrity (I) and availability (A). The scores are derived using version-specific formulas. The final CVSS score of a vulnerability ranges from 0 to 10. For CVSS v3, scores are categorized into four severity levels based on CVSS rating [17]. Scores of 9.0-10.0 are classified as *Critical*, 7.0-8.9 as *High*, 4.0-6.9 as *Medium*, and 0.1-3.9 as *Low*.

III. Methodology

In this section, we explain how we detect the severity score inconsistencies in the NVD by applying several methods to semantically divide vulnerabilities into smaller clusters based on their descriptions. Note that at the high level, we resort to clustering instead of pair-wise comparison because 1) pair-wise similarity involves complications and is not useful in practice, e.g., when A is similar to B, B is similar to C but C is not necessarily similar to A (i.e., no transitivity). 2) handling the number of combinations (pairs) out of hundreds of thousands of records is computationally too expensive. Each entry needs to be compared with every other entry (time/storage overhead). With our clustering approach, entries in each cluster can naturally be considered similar and this similarity is global, as opposed to pair-wise.

A. Semantically Similar Clusters

We utilize clustering algorithms to generate semantically similar CVE entries based on human-readable vulnerability descriptions. Clustering algorithms are unsupervised machine learning techniques that are used to group similar data points together, resulting in more similar data points in the same cluster and distant data points in different clusters.

More specifically, we first utilize techniques such as Word2Vec [18] and TF-IDF [15] to generate numeric representations of the vulnerability descriptions. Then, the clustering algorithm, e.g., *K-Means* clustering, would utilize *Euclidean* distance metric to measure the distance between CVE descriptions. The formed clusters include CVE descriptions that share similar features. The effectiveness of the clustering results relies on both the chosen clustering algorithm and the selected features. Although with the development of NLP, pre-trained models, e.g., BERT [14], can capture the semantic meaning of human-readable text, the specific cybersecurity context may still be missing. Section IV-B shows the detailed comparison of two clustering algorithms with seven features.

To better capture the cybersecurity meaning/context and enhance precision, we also encode vulnerability types (e.g.,

TABLE I: Running Example (all the CVEs are in the same semantically similar cluster)

CVE ID	Description	Score	Type	Vulnerability Entities
CVE-2018-21166	Certain NETGEAR devices are affected by **denial of service**. This affects R6100 before 1.0.1.22, [...] WNR2000v5 before 1.0.0.64.	4.9	*Denial Of Service*	denial of service
CVE-2018-21167	Certain NETGEAR devices are affected by **stored XSS**. This affects D6100 before 1.0.0.57, [...] and WNR2000v5 before 1.0.0.64.	5.5	*Cross Site Scripting*	stored xss
CVE-2018-21170	Certain NETGEAR devices are affected **by a stack-based buffer overflow by an unauthenticated attacker**. This affects EX2700 before 1.0.1.28, [...] and WN3100RPv2 before 1.0.0.56.	8.8	*Overflow*	stack-based buffer overflow, unauthenticated attacker
CVE-2018-21171	Certain NETGEAR devices are affected **by a stack-based buffer overflow by an authenticated user**. This affects D6100 before 1.0.0.57, [...] 1.0.2.98.	6.8	Overflow	stack-based buffer overflow, authenticated user
CVE-2018-21174	Certain NETGEAR devices are affected by **a stack-based buffer overflow by an authenticated user**. This affects D6100 before 1.0.0.57, [...] 1.0.0.62.	7.2	Overflow	stack-based buffer overflow, authenticated user

"Buffer Overflow", "Execute Code") for further cluster refinement. The vulnerability types are assigned by the analysts, and it is known that each vulnerability type carries its own characteristics in the descriptions [9].

Table I shows a running example for our approach. First, the descriptions for the CVEs are embedded into features for clustering. The format of the descriptions is the same for the selected five vulnerabilities. Each description starts with a common introduction to "NETGEAR devices", followed by an introduction of the problem type, culminating with details about impacted products and versions.

Due to the similar structure and wordings in the description, the features corresponding to them will be naturally close, facilitating the clustering algorithm to group them into the same cluster. Our approach then will further categorize entries in a cluster based on vulnerability types. The first three vulnerabilities are associated with different vulnerability types, *Denial of service*, *Cross site scripting*, *Overflow*, and will be further separated into different clusters; leaving only CVE-2018-21170, CVE-2018-21171, and CVE-2018-21174 within the same group.

B. Inconsistency Detection

After obtaining semantically similar clusters, we notice that a seemingly minor word may create a considerable impact on scoring. For example, in Table I, the exploit in CVE-2018- 21170 is permissible by an "unauthenticated attacker", whereas in CVE-2018-21174, only an "authenticated user" can execute the attack.

These kinds of subtle differences can be challenging to be captured by the generated sentence embeddings. However, these words hold paramount importance in the process of assigning CVSS scores. Specifically, the contrast between an "unauthenticated attacker" and an "authenticated user" carries profound implications that significantly influence vulnerability assessments, e.g., the former does not require privilege, while the latter does.

To address this challenge, we utilize a tool called *VIET* [13] to extract cybersecurity-significant entities for each description within the same cluster. These extracted entities provide essential information such as privilege, vulnerability vector, and vulnerability type, which can be loosely correlated with the metrics of PR, AV, and AC in CVSS, respectively. For instance, the "unauthenticated attacker" and "authenticated user" mentioned in Table I identified as privileges, can be aligned with the PR metric; and "stack-based buffer overflow" identified as vulnerability types, can be utilized for the AC metric.

Using VIET to compensate generic NLP-based clustering with cybersecurity significance, there are also several design choices to consider. For instance, VIET classifies the entities into 5 essential entities and 9 supplementary entities. The latter cover computer-related terms (e.g., version and vendor), which is not important for determining severity. The 5 essential entities are determining factors for severity (e.g., privileges and vulnerability types mentioned above), and thus we further refine the clusters as follows: all the essential entities have to match for descriptions to be considered similar. Note that the actual number of extracted essential entities may vary from 0 to 5. After the processing with VIET, we will have the semantically similar clusters ready as we defined.

In this paper, an inconsistent cluster is a semantically similar cluster of CVEs obtained through our clustering and VIET, where at least one CVE has a different CVSS score than the rest in the same cluster. As demonstrated in Table I, the descriptions of CVE-2018-21171 and CVE-2018-21174 are similar and share the same essential entities, "stack-based buffer overflow" and "authenticated user" but with different CVSS scores; the former is 6.8 and the latter is 7.2. Those two entries are considered a case of inconsistency.

IV. Experiment

In this section, we first provide the experimental setting and then present our experimental results and evaluation.

A. Experiment Details

The experiments were all developed in Python 3.9 and executed on a MacBook Pro running macOS Monterey, with Apple M1 Pro chip and 16GB of RAM.

Dataset. We collected 209,842 vulnerability descriptions in the NVD from 1999 to March 2023. We discarded entries listed as *REJECT*, and kept only those associated with CVSS v3 metrics, resulting in 124,034 vulnerabilities. Note that our methodology is also applicable to entries with only CVSSv2, though note that CVSSv3 metrics are already assigned to all CVEs from 2017.

Ground-Truth Dataset. To evaluate the effectiveness of the clustering methods, we randomly sample a subset of 1,000

CVE entries from the past six years (2017-March 2023) and manually classify them into clusters. This manual classification requires expert knowledge to identify semantically similar vulnerabilities based on human-readable descriptions, with respect to essential information that may affect the CVSS severity scoring. Two authors independently labeled the data and labels in disagreement were discussed and resolved to minimize human errors. The results, 1000 entries manually divided into semantically similar clusters, are used to evaluate the correctness of our inconsistency detection.

B. Effectiveness of Clustering Methods

To select an optimal combination of clustering algorithms and input features to cluster the CVEs effectively, we evaluate two popular clustering algorithms, namely *K-Means clustering* [19] and the *Agglomerative clustering* [20] methods, using seven features (detailed in Table II). Generally, *K-Means clustering* demonstrates efficient performance, but it requires users to pre-define the number of potential clusters (k) for the dataset, which can be challenging to determine. Agglomerative clustering performs slower; however, it offers the advantage of not requiring users to specify the number of clusters in advance. To enhance the efficiency of Agglomerative clustering, we adopted Fast Agglomerative clustering [21], a variant that utilizes local community search to accelerate the clustering speed. This approach reduces computational overhead, allowing faster clustering on larger datasets without compromising clustering quality, making it ideal for our experiments.

Features. The features we considered in this experiment include vulnerability linguistic model embedding (a customized Word2Vec embedding trained on vulnerability descriptions, referred to as *V-CBOW* as the model is trained based on the CBOW variant) [13], *TF-IDF* [15], and BERT [14]. The V-CBOW features contain vulnerability-specific information within the vulnerability descriptions, while the TF-IDF features only consider word frequencies in a description. Features from the pre-trained BERT can generate sentence-level embedding that expects to capture the most comprehensive human language semantics and complex contextual relationships.

Evaluation Methods. After we extract all the features listed in Table II and Table III, we first perform clustering based on the Agglomerative clustering. Then, we use the cluster numbers as the predefined clusters (k) to perform the same experiment in the K-Means algorithm. Notably, different features may lead to different clusters; and we maintain the same number of clusters between both algorithms for fair comparison.

The Normalized Mutual Information (NMI) is used to evaluate the performance of each clustering method with different text features, word frequency, and semantics meaning of the text [13], [22]. The NMI measures the mutual information between clustering results and real labels, taking into account the uneven distribution of labels and clusters. Since our clustered dataset is unbalanced, it is suitable for evaluating our clustering results.

Clustering Features	K-Means Clustering	Agglomerative Clustering
V-CBOW	0.6934	0.6880
TF-IDF	0.7024	0.7781
BERT	0.7333	0.7329
V-CBOW + TF-IDF	0.7892	0.8051
BERT + TF-IDF	0.7957	0.7948
BERT + V-CBOW	0.8158	0.8050
BERT + V-CBOW + TF-IDF	0.8227	0.8100

TABLE II: Evaluation of clustering 1000 CVE entries without Vulnerability Type Information

Clustering Features	K-Means Clustering	Agglomerative Clustering
V-CBOW	0.8179	0.8135
TF-IDF	0.8047	0.8358
BERT	0.8867	0.8854
V-CBOW + TF-IDF	0.8512	0.8399
BERT + TF-IDF	0.8392	0.8413
BERT + V-CBOW	0.8762	0.8739
BERT + V-CBOW + TF-IDF	0.8538	0.8418

TABLE III: Evaluation of clustering 1000 CVE entries with Vulnerability Type Information

Evaluation Results. The results of both clustering algorithms based on seven different features, with and without vulnerability type information, are presented in Table II and Table III. We observe that, in general, the inclusion of vulnerability type information improves the NMI score for both algorithms. For instance, in Table II, the best feature combination, *BERT + V-CBOW + TF-IDF*, achieves NMI scores of 0.8227 and 0.81 without vulnerability type information. However, when vulnerability type information is introduced, the NMI scores increase to 0.8538 and 0.8418, respectively. Interestingly, we notice that this feature combination performs worse than using only the BERT feature when vulnerability type information is added. It might be due to the clustering algorithms giving more importance to the differences in data types, rendering semantic and lexical information less necessary for distinguishing data. Additionally, the classification feature may contain security-related information, making other features such as V-CBOW and TF-IDF redundant and resulting in a less optimal outcome compared to using BERT alone. Overall, both algorithms demonstrate improved performance when vulnerability type information is utilized, which validates our choice of including vulnerability type in the clustering process. Thus, in the later experiment, Agglomerative clustering with BERT feature is chosen to cluster the vulnerabilities (although K-Means algorithm performs slightly better, it is challenging to define the k value without the help of another algorithm).

C. Effectiveness of Inconsistency Detection

Across all 124,034 CVE entries, after applying the clustering method with the best performance (Agglomerative clustering with BERT embedding and vulnerability type information) and classification of vulnerability type to obtain 9,057 clusters; in total, resulting in 54,018 CVE entries (we remove the clusters that contain only one entry as they can not form any

inconsistency). We first perform semantic clustering and literal clustering on those entries. After utilizing the literal clustering that screens out the identical descriptions in each cluster, there are 8034 CVE entries (2,234 identical clusters) left for check consistency. Those clusters should naturally have the same severity levels, while we detect that 528 clusters (2,350 CVE entities) have inconsistent scores.

The semantic clusters consist of the CVE descriptions with similar sentence structure/meaning. After extracting all the essential entities in the same cluster, we compare the CVSS score among the entries sharing identical essential entities. We remove the clusters with only one entry. In the end, we identify inconsistencies in 1,574 clusters out of 33,120 clusters, accounting for a total number of 12,866 entries out of 53,711 entries. We provide further in-depth study in Section IV-D.

Evaluation Methods. Our ground truth dataset involves multi-classification challenges, resulting in multiple sets of confusion matrices. For evaluating global performance, we employ both macro average and weighted average metrics. The macro average computes the overall average for all classes, while the weighted average incorporates class-specific weights, which is especially useful for addressing significant data imbalances. These weights can be calculated based on the percentage of the data ratio for each class. We calculate accuracy, precision, recall, and F1-score based on macro and weighted averages.

Evaluation Results. Table IV shows our evaluation results. The macro average and weighted average have the same accuracy, 0.794. The weighted average shows much higher results than the macro average in precision, recall, and F1-score. This may be because the weighted average focuses more on overall performance, while the macro average emphasizes average performance across classes. Since the number of samples in different classes varies a lot, the weighted average gives more importance of the classes with larger sample sizes, resulting in a relatively high weighted average value.

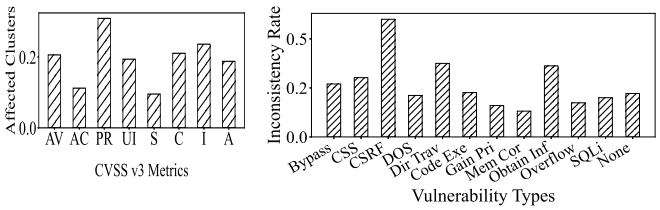
	Accuracy	Precision	Recall	F1-Score
Macro Average	0.794	0.695	0.646	0.661
Weighted Average	0.794	0.996	0.794	0.852

TABLE IV: Effectiveness of inconsistency detection

D. Inconsistency Analysis

We provide an analysis of the discovered inconsistencies in literal and semantic clusters. The analysis focuses on discovering inconsistent CVSS metrics, the inconsistencies in different vulnerability types, and the trend of inconsistencies over the past ten years.

Inconsistency in the Literal Match. Figure 2a shows the inconsistent metrics (the CVSS metric that caused inconsistency in the inconsistent clusters) within the literal matched clusters. We first obtain the CVSS v3 metrics for each CVE ID (metrics: AV, AC, PR, UI, S, C, I, A) in the cluster. Then, we compare each metric value within them. If the value of a metric



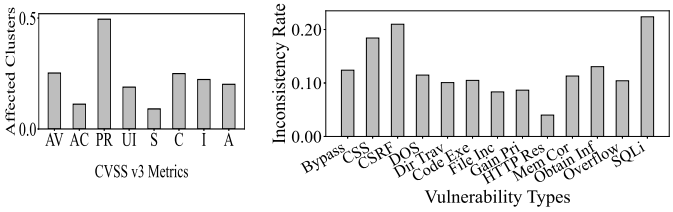
(a) Inconsistent metrics (b) Inconsistency rate vs. types

Fig. 2: Inconsistencies in different CVSS metrics and vulnerability types (literal)

differs from any other metric values in the cluster, we consider this particular metric causes inconsistency. Once we finish comparing the metric values over the 528 inconsistent clusters, we calculate the occurrence percentage of each metric. This percentage corresponds to the number of clusters affected by a particular metric over the total count of inconsistent clusters.

Figure 2b shows the inconsistency rate for different vulnerability types. There are 13 vulnerability types in all the entries, considering some entries with no type. We calculate the type-specific inconsistency rate by dividing clusters with that type inconsistency by all clusters of that type in question.

Result and Implications. The result in Figure 2a indicates that the Privileges Required (PR) metric is the most error-prone metric (38.64%) in the literally matched clusters, followed by the impact metric I (29.55%). In contrast, analysts rarely generate inconsistent CVSS scores based on the Scope (S) metric. The impact metrics (C, A) and the attack vector metric (AC) share similar rates (∼23%-25%). The result in Figure 2b shows that the minimal occurrence of inconsistency cases is in the vulnerability type "Memory Corruption" (Mem Cor) (13.2%), while "CSRF" has the highest inconsistent scores (60%). We also observe that the majority of the vulnerability types contain around ∼20% of inconsistencies.



(a) Inconsistent metrics (b) Inconsistency rate vs. types

Fig. 3: Inconsistencies in different CVSS metrics and vulnerability types (semantic)

Inconsistency in the Semantic Match. Figure 3a shows the inconsistent metrics within the semantically matched clusters. We have obtained the original clusters by applying the clustering method with the vulnerability type. Based on the original clusters, we extract the essential entities from the vulnerability description of the CVE ID within the same cluster. Similar

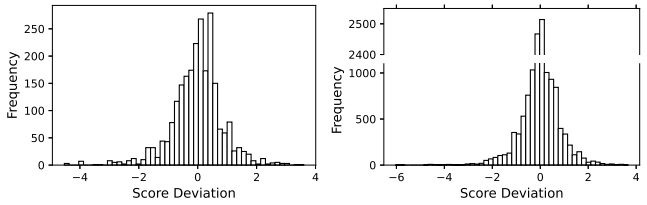
Year	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
#Inconsistent CVE (Percentage among CVSSv3-assigned CVEs)	99 (8.49%)	186 (7.12%)	1068 (11.80%)	1659 (11.34%)	1717 (10.94%)	1649 (10.66%)	1418 (7.69%)	1623 (7.78%)	2896 (13.23%)	469 (17.87%)

TABLE V: Number of descriptive inconsistencies per year in the NVD database (between January 2014 – March 2023)

to the literally matched clusters, we calculate the number of inconsistency clusters caused by eight CVSS v3 metrics. We also calculate the inconsistency rate per-type as in the literally matched clusters.

Result and Implications. The results in Figure 3a share a similar trend to that obtained through literally matched clusters. The inconsistent CVSS v3 scores are most affected by the Privileges Required (PR) metric in vulnerability descriptions (49.5%). This rate is much higher than the ones in the literal clusters. The inconsistency rate is relatively low in the Attack Complexity (AC) and Scope (S) metrics. The results in Figure 3b show that there are more clusters with inconsistent CVSS V3 scores under the vulnerability types "CSRF", and "SQL Injection" (over 20%), and the rate of the "Http response splitting" type is only 4%.

Inconsistencies over Years. In addition, we selected CVE entries from the past decade to analyze the number of inconsistent entries per year. In the past decade, there have been a total of 122,481 CVE entities, including 12,784 inconsistent entries. Then, we count the number of descriptive inconsistencies per year from January 2014 to March 2023 in the NVD database and calculate their ratio to the total number of CVE entries per year. We observe that the inconsistency ratio is stable over the years, around $\sim$8% to 12% in Table V. It also shows that the proportion of descriptive inconsistencies has increased in 2022 (13.23%) and exceeded 15% in 2023 (as 2023 is still partial as of this writing).



(a) Literal cluster (b) Semantic cluster

Fig. 4: CVSS score deviation in literal matched clusters and semantic clusters

The Score Deviation in the Inconsistency Clusters. Figure 4 shows the score deviation of both literal and semantic clusters. We first calculate the average CVSS score for each inconsistent cluster, then measure the distance between individual scores and this calculated average. The histogram of the deviation of the CVSS score in inconsistency clusters provides insight into the extent of variation that severity assessments can become within seemingly similar vulnerability groups.

Result and Implications. In Figure 4a we observe that in the literal matched clusters, the CVSS score demonstrates a more pronounced positive deviation from the average value. This indicates that inconsistencies in the literal clusters frequently result in higher ratings. This pattern, however, is not observed within the semantic matched clusters. In Figure 4b, a more significant cluster of entries is noticeable around the -2 and +2 marks, reflecting a substantial degree of deviation. This observation suggests that semantic matched groups might contain a broader range of distinctly rated values, signifying a higher degree of variability in severity assessments.

V. Case Study and Discussion

In this section, we conduct a case study to demonstrate the real-world impact of our inconsistency measurements. We present two groups, one selected from the semantic match clusters and the other one from the literal match clusters. The first group contains five unique CVE entries with semantically similar descriptions, and the first two CVE entries (CVE-2020-3406, CVE-2020-3591) have exactly the same description but different CVSS scores. The second group contains three CVEs with identical descriptions but assigned different scores with large discrepancies. CVEs in each cluster pertain to the same vulnerability type and also have exactly the same essential entities predicted by the NER model, which further indicates that their descriptions should be very similar semantically.

According to the information from the captured essential entities and their corresponding descriptions, we manually analyze the results to see which of the CVSS vectors are in error. We also utilize majority voting to derive a relatively reliable assessment of the metrics.

Observation from the Semantic Group. By applying VIET to extract the essential entities, we confirm that a few of the CVE entries are indeed human judgment errors that lead to inconsistencies appearing in the metrics. The semantic group in Table VI has five unique CVE IDs, all pertaining to the same vulnerability type with identical essential entities, but assigned different scores ranging from 4.3 to 6.4. As each entity could be mapped to relevant vectors, the authors checked each entity and description. We observed that "persuading a user to click a crafted link" means human interaction required, whereas CVE-2020-3591 and CVE-2020-3590 have designated this vector as "None". Moreover, the text "allow the attacker to execute arbitrary script code in the context of the interface..." also indicates that the confidentiality of CVE-2020-3591 should not be assigned "None".

Observation from the Literal Group. In Table VI, the second group, termed the literal group, consists of three CVE descriptions that are identical. All pertain to the vulnerability type "Execute Code" and share the same essential entities as extracted by VIET. Notably, despite identical descriptions, the

TABLE VI: The case study of two inconsistent clusters

Group Property	CVE ID	Vul. Type	Description	CVSS3	Vul. Entities	CVSS Metrics	Suggestion
Semantic Group (Similar CVE Descriptions)	CVE-2020-3406	XSS	A vulnerability [...] An attacker could exploit this vulnerability by persuading a user to click a crafted link. A successful exploit could allow the attacker to execute arbitrary script code in the context of the interface or access sensitive, browser-based information.	5.4	'allow an authenticated', 'remote attacker', 'cross-site scripting', 'xss validate user-supplied input', 'persuading a user', 'crafted link', 'allow the attacker', 'execute arbitrary script code'	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N	**Manual checking:** UI ⇒ required C ⇒ not "None" **Majority Voting:** AV: Network AC: Low PR: Low UI: Required S: Changed C: Low I: Low AV: None
	CVE-2020-3591	XSS		4.3		CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N	
	CVE-2020-3590	XSS	A vulnerability [...]. An attacker could [...] context of the interface or access sensitive, browser-based information.	6.4		CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N	
	CVE-2020-3523	XSS	A vulnerability [...]. An attacker could [...] context of the interface or access sensitive, browser-based information.	5.4		CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N	
	CVE-2021-1130	XSS	A vulnerability [...]. An attacker could [...] need to have administrative credentials on the affected device.	4.8		CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:C/C:L/I:L/A:N	
Literal Group (Identical CVE Descriptions)	CVE-2017-5808	Execute Code	A Remote Arbitrary Code Execution vulnerability in HPE Data Protector version prior to 8.17 and 9.09 was found.	7.5	'Remote Arbitrary Code Execution'	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	**Manual checking:** AV ⇒ not "Local" **Majority Voting:** AV: Network AC: Low PR: None UI: None S: Unchanged C: High I: None A: High
	CVE-2017-5809	Execute Code		5.5		CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	
	CVE-2017-5807	Execute Code		9.8		CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	

associated scores vary considerably, spanning three severity levels: medium, high, and critical. The sole discernible metric is "Remote," correlating to the attack vector (AV). An inconsistency was identified in CVE-2017-5809, where the metric "Local" contradicts the textual information. For other metrics, namely privilege (PR), confidentiality (C), integrity (I), and availability (A), it is unclear how to populate the values due to missing information in the descriptions.

Potential Root Cause of Description Inconsistencies. We investigated the root cause of the discrepancies between the three CVE entries mentioned above and in our motivating example. According to the vulnerability write-ups referred in the CVE entries, we found that the first one is indeed related to an unauthenticated remote code execution vulnerability with SYSTEM privileges (Windows), which is indeed a critical security flaw and accurately ranked as AV:N/C:H/I:H/A:H. However, the nature of remaining vulnerabilities is different: the second one enables an unauthenticated remote attacker to cause a denial of service (correctly ranked as AV:N/C:N/I:N/A:H), leading to a high severity; the last one is a local privilege escalation vulnerability (again, correctly ranked as AV:L/C:H/I:N/A:N), leading to a medium severity. In summary, *the CVSS metrics agree with the true nature of the vulnerability but not with the descriptions*. We speculate that the three vulnerabilities were processed together (given the consecutive IDs) and the description of the first one was copy-pasted to the other CVE entries by an analyst without anyone noticing.

VI. Related Work

In this section, we briefly discuss several prior works that are close to our work or share similar methodologies.

A large body of research has explored to extract various features from the rich vulnerability information of NVD/CVE repositories to predict severity level, impact, and other consequences of vulnerabilities [7], [9], [23], [24], [25], [26], [27], [28], [29], [30], among which the use of the free-form description text has been prevalent.

NLP-based Scoring/Ranking using Descriptions. The severity level of vulnerabilities can help determine which system needs to be patched first (given limited resources), direct investments, serve as an evaluation factor, etc. Therefore, using NLP (as opposed to the context-less traditional static analysis) to classify vulnerabilities based on severity, or derive a severity score, has become an established research direction. For instance, Han et al. [23] and Sharma et al. [25] use word embedding and CNN to automatically capture discriminative words and sentence features of vulnerability descriptions, to predict the severity level of a vulnerability. A similar work done by Costa et al. [29] uses DistilBERT to process the descriptions. In addition to descriptions alone, Ni et al. [31] combine BERT's specific task layer with CNN to predict the severity of vulnerabilities by fine-tuning BERT to handle vulnerability descriptions and other information, including access permissions obtained, attack sources, and required authentication. Taking a further step forward, Binyamini et al. [22] derive the causal relationship between vulnerability exploits and construct potential attack paths from the vulnerability descriptions using NLP. Zhang et al. [13] propose to extract essential attack-related entities to facilitate vulnerability score rating from descriptions with fine-tuned cybersecurity-rich text embedding. In our work, we identify semantically similar descriptions for inconsistency comparison using their tool VIET [13] and clustering methods.

Inconsistency of Vulnerability Databases. Before the NLP-based severity scoring, there were already the CVSS metrics as our study aims to examine, which are assigned by human analysts. Unsurprisingly, there exist inconsistencies of various aspects of such assigned scores in the vulnerability databases. Noticing the inconsistencies between multiple vulnerability databases, Jiang et al. [24] use the vulnerability description in NVD as training input and conduct majority voting on the inconsistent scores, thereby simplifying the derivation of vulnerability severity and solving the problem of inconsistency. They also take into account structured machine-readable data, e.g., version and vendor. Dong et al. [9] focus on the two

widely used databases NIST NVD and MITRE CVE and detect the severity score inconsistencies on a massive scale. Anwar et al. [10] focus on NVD and uncover inconsistency errors within one entry, e.g., vulnerability publication dates and applications affected by the vulnerability. Our work focuses on the inconsistencies within semantically similar vulnerability descriptions; we observe that different CVE entries can contain the same description (or semantically similar descriptions), and after being evaluated by human analysts, got assigned inconsistent CVSS scores. This indicates, logically, there must be one or multiple entries assigned inaccurate severity scores, or it might be incorrect descriptions which should not have been the same/similar.

VII. Conclusion

Inconsistencies in vulnerability databases have recently been identified one after another, and we advanced this area further to consider a type of inconsistency where vulnerabilities with similar descriptions in a cybersecurity sense or even exactly the same description received different severity scores. We proposed an approach that can detect such inconsistencies on a large scale by adapting several methods/tools. Our analysis revealed around ten percent of the current vulnerabilities in the NVD potentially have this issue. We also discussed various implications of such inconsistency issues and believe that future research will be able to address each of such implications, e.g., erroneous data of the vulnerability database and inconsistent scoring.

References

[1] The National Institute of Standards and Technology, "NVD Data Feeds," https://nvd.nist.gov/vuln.

[2] M. Souppaya and K. Scarfone, "Guide to enterprise patch management planning: Preventive maintenance for technology," National Institute of Standards and Technology, NIST Special Publication NIST SP 800-40r4, Apr. 2022.

[3] "OpenSCAP," last visited Sep. 3, 2023. https://www.open-scap.org/.

[4] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, and Y. Zhong, "VulDeePecker: A deep learning-based system for vulnerability detection," in *Network and Distributed System Security (NDSS'18)*, 2018.

[5] 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 *USENIX Security Symposium (USENIX Security)*, Aug. 2022, pp. 1993–2010.

[6] Y. Mirsky, G. Macon, M. D. Brown, C. Yagemann, M. Pruett, E. Downing, S. Mertoguno, and W. Lee, "Vulchecker: Graph-based vulnerability localization in source code," in *USENIX Security Symposium (USENIX Security)*, Aug. 2023, pp. 6557–6574.

[7] A. Khazaei, M. Ghasemzadeh, and V. Derhami, "An automatic method for CVSS score prediction using vulnerabilities description," *Journal of Intelligent and Fuzzy Systems*, vol. 30, no. 1, pp. 89–96, 2016.

[8] L. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 1, pp. 1–20, 2014.

[9] Y. Dong, W. Guo, Y. Chen, X. Xing, Y. Zhang, and G. Wang, "Towards the detection of inconsistencies in public security vulnerability reports," in *USENIX Security Symposium (USENIX Security)*, Aug. 2019, pp. 869–885.

[10] 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 (TDSC)*, vol. 19, no. 6, pp. 4255–4269, 2021.

[11] M. Zhang, X. de Carné de Carnavalet, L. Wang, and A. Ragab, "Large-scale empirical study of important features indicative of discovered vulnerabilities to assess application security," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 14, no. 9, pp. 2315–2330, 2019.

[12] The MITRE Corporation, "How are the CVE Record DESCRIPTIONS created or compiled?" https://www.cve.org/ResourcesSupport/FAQs.

[13] S. Zhang, M. Zhang, and L. Zhao, "VIET: A tool for extracting essential information from vulnerability descriptions for CVSS evaluation," in *Conference on Data and Applications Security and Privacy (DBSec'23)*, vol. 13942, 2023, pp. 386–403.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT'19)*, 2019, pp. 4171–4186.

[15] A. Simha, "Understanding TF-IDF for machine learning," https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

[16] The MITRE Corporation, "CVE Website," https://cve.mitre.org/.

[17] Atlassian, "Severity levels for security issues," https://www.atlassian.com/trust/security/security-severity-levels.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations (ICLR'13)*, 2013.

[19] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *JSTOR: Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[20] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *Computing Research Repository (Corr)*, vol. abs/1109.2378, 2011.

[21] P. Fränti, O. Virmajoki, and V. Hautamäki, "Fast agglomerative clustering using a k-nearest neighbor graph," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 28, no. 11, pp. 1875–1881, 2006.

[22] H. Binyamini, R. Bitton, M. Inokuchi, T. Yagyu, Y. Elovici, and A. Shabtai, "A framework for modeling cyber attack techniques from security vulnerability descriptions," in *SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD'21)*, 2021, pp. 2574–2583.

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

[24] Y. Jiang and Y. Atif, "Towards automatic discovery and assessment of vulnerability severity in cyber-physical systems," *Array*, vol. 15, p. 100209, 2022.

[25] 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, 2021.

[26] S. Zhang, D. Caragea, and X. Ou, "An empirical study on using the national vulnerability database to predict software vulnerabilities," in *Database and Expert Systems Applications (DEXA'11)*, vol. 6860, 2011, pp. 217–231.

[27] S. Zhang, X. Ou, and D. Caragea, "Predicting cyber risks through national vulnerability database," *Information Security Journal: A Global Perspective*, vol. 24, no. 4-6, pp. 194–206, 2015.

[28] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'10)*, 2010, pp. 105–114.

[29] J. C. Costa, T. Roxo, J. B. F. Sequeiros, H. Proença, and P. R. M. Inácio, "Predicting CVSS metric via description interpretation," *IEEE Access*, vol. 10, pp. 59 125–59 134, 2022.

[30] J. Walden, J. Stuckman, and R. Scandariato, "Predicting vulnerable components: Software metrics vs text mining," in *IEEE International Symposium on Software Reliability Engineering (ISSRE'14)*, 2014, pp. 23–33.

[31] X. Ni, J. Zheng, Y. Guo, X. Jin, and L. Li, "Predicting severity of software vulnerability based on BERT-CNN," in *International Conference on Computer Engineering and Artificial Intelligence (ICCEAI'22)*, 2022, pp. 711–715.