Fixing Invalid CVE-CWE Mappings in Threat Databases

Şevval Şimşek
Boston University
Boston, MA, USA
sevvals@bu.edu

Howell Xia
Boston University
Boston, MA, USA
howellx@bu.edu

Jonah Gluck
Boston University
Boston, MA, USA
jonahg@bu.edu

David Sastre Medina
Red Hat Inc.
Madrid, Spain
asastrem@redhat.com

David Starobinski
Boston University
Boston, MA, USA
staro@bu.edu

***Abstract*—Accurate root cause analysis plays a key role for developing mitigation strategies and understanding attack paths. Many security analysis tools rely on threat databases to accurately report information related to vulnerabilities, such as root cause weaknesses or affected platforms. However, these databases are not entirely correct, with many instances of missing or erroneous information linked to vulnerabilities. This paper presents a method for automated correction of invalid Common Weakness Enumeration (CWE) mappings of Common Vulnerability and Exposure (CVE) entries in the National Vulnerability Database (NVD), which can also be applied to other threat databases. We systematically investigate the prevalence of incorrect or missing root-cause mappings, revealing that more than half of CVEs are linked to invalid or insufficiently detailed CWEs, particularly those categorized as *Prohibited* or *Discouraged*. Through a longitudinal analysis of the NVD, we detect trends in manual updates to CVE-CWE mappings and show how these can inform predictions for future corrections. We develop and present `FixV2W`, an automated correction method that uses a Knowledge Graph embedding model to predict and rank best-fitting CWE matches for correcting previously invalid CVE-CWE mappings. We evaluate `FixV2W` using invalid mappings that were subsequently corrected by the NVD. Notably, focusing on the top-10 ranked answers for correcting prohibited mappings, we show that `FixV2W` finds the correct CWE in 65% of the cases, and a candidate within the same branch as the correct CWE in 93% of the cases. Moreover, most of the correct mappings appear at the first or second ranks.**

***Index Terms*—Vulnerability, NVD, Weakness, Knowledge Graphs, Embedding.**

I. Introduction

Many security analysis tools [1]–[3] identify vulnerabilities by scanning libraries and dependencies, providing reports that include detected vulnerabilities and their severity scores. The accuracy of vulnerability databases is critical for security analysis tools, which many open-source software (OSS) developers and security professionals rely on for risk assessment and mitigation. The National Vulnerability Database (NVD) [4] serves as a key resource by providing *enrichment* metadata -additional information regarding the vulnerability- for Common Vulnerabilities and Exposures (CVEs) [5], including mappings to Common Weakness Enumeration (CWE) [6], Common Platform Enumeration (CPE) [7] and CVSS (Common Vulnerability Scoring System) scores. Similar to the NVD, other databases [8]–[10] also provide additional information on the CVE records, such as purl [11], or their own severity scores, while also including the data provided by the NVD. These mappings enable automated security tools to assess vulnerabilities, identify trends, and recommend mitigation. In particular, CWE mappings provide highly valuable information on the root causes of vulnerabilities, thus contributing to better understanding on how to mitigate them.

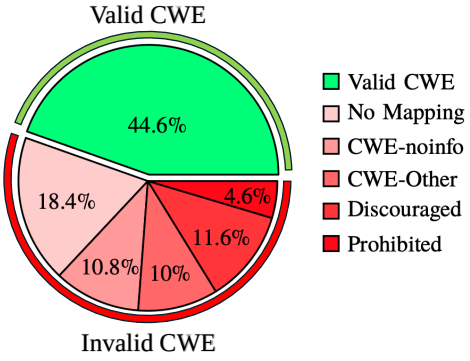


Fig. 1: The proportion of invalid mappings versus valid CVE-CWE mappings among 280,000+ CVE (as of December 17, 2024). The diagram shows that valid mappings currently represent less than half of the entries in the NVD.

Unfortunately, the NVD has been plagued with incomplete and inaccurate analyses [12], as well as significant delays in the analysis of newly reported CVE, which has left OSS developers and organizations in the dark for extended periods. NVD, although ensuring that it will go back to its processing pace before the huge drop in February 2024, is not even halfway done with more than 30,000 newly reported CVE in 2024 and first few months of 2025. Although recent problems have taken most of the spotlight, missing or inaccurate data has in fact been a long-lasting problem. Through our longitudinal analysis, our paper sheds further light into this issue.

Strikingly, our analysis of NVD entries as of December 2024, shown in Figure 1, determines that 55% of all CVE appearing in the NVD have been mapped to invalid or missing CWEs, that is CWE that are fully invalid (*Prohibited*), lacking sufficient detail (*Discouraged*), or just serving as placeholders for missing entries (*CWE-Other* and *CWE-noinfo*). Note that the *Prohibited* and *Discouraged* tags were introduced in 2019

by MITRE for certain entries in the CWE list [13], aiming to discourage CVE Numbering Authorities (CNAs) from using generic CWE entries containing no or insufficient information to conduct proper root cause analysis of vulnerabilities. While the use of Prohibited CWEs was discontinued after 2019, we detected that 5% of all CVE submitted since 2019 are still mapped to Discouraged CWE.

While incomplete or invalid CVE-CWE mappings represent the majority of mappings in the NVD, little research has been conducted on how this situation has emerged and evolved over time, and how to address this problem in a systematic way (see Section II for some of the related work on this topic). As such, the contributions of this paper are two-fold: (i) we perform an analysis of the evolution of the NVD over a long time period (from 2016 till now), to gain a thorough understanding of the prevalence and manifestations of this problem; (ii) we develop a rigorous method, based on the theory of knowledge graphs (KGs) [14], to fix this problem in a systematic and scalable manner.

A. Contributions

We summarize our contributions as follows:

- We systematically analyze the NVD to measure the amount of invalid and missing CVE-CWE mappings and their prevalence. Interestingly, we discover that only few invalid mappings have been corrected, even after the concepts of Prohibited and Discouraged CWEs were introduced in 2019.
- Our longitudinal analysis reveals that a significant portion of updates to CVE-CWE mappings occur within the same branch on the CWE hierarchy [15], generally one hop or two hops away. We leverage this finding in the design of our automated correction method.
- We propose and thoroughly evaluate `FixV2W`, a novel method for correcting CVE-CWE mappings using Knowledge Graph Embeddings (KGE) on an ontology consisting of the relationships between CVE, CWE, and CPE, representing vulnerability data in the NVD. Remarkably, our model does not require additional data or semantic input, such as LLMs.
- We demonstrate that our methods are able to correct Prohibited and Discouraged CVE-CWE mappings with a mean rank of 1.86 and 1.03, respectively, among the predictions where the correct CWE is found within the top-10 answers. Furthermore, our validation data set is distinct from our training data set. Namely, our model is validated under an open-world assumption (OWA) which is generally considered more realistic than the closed-world assumption (CWA) predominantly used in the evaluation of knowledge graphs [16].

The significance of our results is as follows. Correcting errors in the root cause mappings of CVE and automating the CVE-CWE mapping correction process can significantly improve the quality of security and threat databases. This improvement is expected to yield higher accuracy in the numerous security analysis tools, software security practices and mitigation plans relying on data provided by the NVD [17] and other vulnerability databases. When CVEs are mapped to generic, outdated, or discouraged CWE, security tools may fail to provide meaningful insights, weakening the overall cybersecurity posture of organizations and OSS ecosystems. Enhanced accuracy also boosts confidence in security tools, reducing fatigue from false reports and increasing the adoption of security measures in the software development life-cycle. Beyond accuracy, improved mappings can streamline and speed-up vulnerability assessments, allowing analysts to focus on actual threats rather than false reports. This allows organizations to prioritize remediation efforts effectively, addressing critical vulnerabilities first.

The remainder of this paper is organized as follows. In section II, we discuss related work. In Section III, we detail the methodology we followed for conducting our longitudinal analysis, and explain how we built our knowledge graph and implemented our automated correction method, `FixV2W` based on this KG ontology. Next, in Section IV, we present results from the longitudinal analysis and the accuracy of our correction method using real data sets collected from the NVD in 2021 (for training) and in 2024 (for validation). Lastly, we present a conclusion for our research in Section VI.

II. Related Work

Analyzing the NVD has been in the spotlight of many researchers and we see that different issues have been addressed by different studies. Some studies focus on the CVSS accuracy, or missing metadata, whereas others focus on the viability of the CVE data, and completeness of related information. On the other hand, methods to automate and correct related fields such as CWE have been of interest to a smaller audience. We next discuss several studies that use machine learning methods to automate CVE to CWE mappings using natural language processing (NLP), Neural Networks, Knowledge Graphs and Large Language Models (LLMs).

We first review studies on measuring data quality of vulnerability databases. Zhang et al. [18] present a model to predict zero-day vulnerabilities based on the NVD. They find that data in the NVD presents many limitations, including missing version information, data errors and late vulnerability release times. We deduct from their study that correcting errors in the NVD is the first and critical step to using the database for such predictive models. Anwar et al. [12] assess the quality of NVD, addressing incorrect vendor information and vulnerability publication dates that affect CVSS scores. They develop an automation tool to correct these errors. While their work examine missing or incorrect CVE entry information, we focus on *relationships* between CVE and CWE entries. Nguyen et al. [19] study vulnerabilities in Google Chrome, revealing that many originated from early versions and highlighting errors in vulnerability data. This supports our assertion that CVE-CPE mappings can be flawed, leading to incorrect enrichment metadata. Kühn et al. [20] apply machine learning and NLP to improve the NVD's information quality, focusing on the atomic accuracy of CVE entries (e.g., security relevant tags and the

CVSS score). In contrast, we analyze the NVD focused on the root causes of the vulnerabilities (namely their weaknesses), and correcting mappings between CVEs and CWES.

Next, we discuss related work on automating CVE to CWE mappings. ThreatZoom [21] employs an adaptive hierarchical neural network for CVE to CWE matching, achieving 92% coarse-grain accuracy with the NVD and 75% with MITRE's CVE database. Their method focuses on unclassified CVE, where the validation set is unavailable and generated by domain experts. In contrast, we focus on correcting erroneous mappings across the entire CVE database, and we provide exact match predictions for 63% of the test set. Moreover, our model is lightweight and does not require any semantic information. CVE2CWE [22] uses NLP and a TF-IDF algorithm to assess semantic similarity between CWEs and unseen CVEs, reporting 70% and 57% similarity for the top 25 and 50 CWEs, respectively, without validation. Notably, only 50 of the 130 CWEs in the CWE-1003 list were considered, and this test set's accuracy is significantly lower than ours. VulnScopper [23] uses ULTRA, a KG foundation model and expanding on it using OpenAI's Ada LLM to create the description-level representation. Our model achieves similar accuracy with higher MRR, and is only based on KG embeddings without the need for additional semantics-processing. Z. Shi et al. [24] focuses on predicting candidates for future associations, such as CVE-CWE mappings, using KGs. On an open-world evaluation, Shi [24] achieves a 0.443 mapping accuracy for top 10 returned CWE-CVE mapping results. Lastly, although not published, CyberSecAI developed a closed grounded system based on ChatGPT called CWE GPT [25] as part of MITRE's CWE working group. This tool aims to reduce hallucinations exiting in such LLM models, via supplying the model with CWE specifications, to help the model make more informed decisions. While initial results appeared to seem promising, the study shows that the model cannot eliminate hallucinations.

To the best of our knowledge, this is the first work that focuses on specifically on correcting the incorrect data within the NVD. While others focus on mapping new CVEs or correcting the labeling of existing mappings, our model focuses on correcting the CVE-CWE mappings themselves, leveraging the longitudinal trends revealed in our work.

III. Methodology

In this section, we describe our methodology in two parts: first, we detail our longitudinal study to systematically analyze the NVD to measure the amount of invalid and missing metadata, and to identify key trends; second, we detail our knowledge-graph prediction method `FixV2W` for correcting invalid CVE-CWE mappings.

A. Longitudinal Analysis of NVD data

We start with providing a brief background on common concepts and key information needed to understand the motivation of our work. Next, we explain the process for our longitudinal analysis along with the significance of the data.

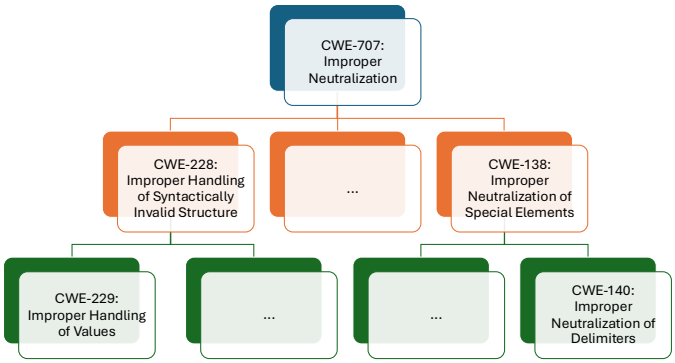


Fig. 2: A slice of the CWE Hierarchy, showing CWE-707 is Parent of CWE-228 and other CWE, CWE-228 is parent of CWE-229 and other CWE. CWE-707 is labeled *Discouraged* for CVE root cause mapping, due to lack of detail. At each level, the detail for the weakness description increases.

1) Background and Terminology: Common Vulnerabilities and Exposures (CVE) is a system, created by MITRE, for referencing publicly known vulnerabilities. Each new vulnerability submitted to the CVE list gets assigned a CVE ID. A vulnerability is submitted by a Certified Numbering Authority (CNA), which is usually the vendor of the affected product.

The Common Weakness Enumeration (CWE) list is created and maintained by MITRE. The CWE list is a categorization system for hardware and software weaknesses, that under certain circumstances, could contribute to the introduction of vulnerabilities. A *CWE ID* can refer to weaknesses, categories or views. A category is a collection of weaknesses based on some common characteristic or attribute. Views are similar, and they group weaknesses for other types of reasons, such as existing in the same Top 25 CWE list of a year. Weaknesses are connected to each other in a hierarchy where one CWE is the *Parent* or *Child* or *Peer* of another. The hierarchy often starts with a category, followed by a class/base weakness (which is very general), and ends with a variant weakness (much more detailed). An example CWE branch depicting the hierarchy is shown in Figure 2.

CWE is highly valuable resource for understanding the root cause of vulnerabilities, and represents a key component of the metadata associated with a vulnerability. Until 2019, all kinds of CWE were used for root cause mapping of vulnerabilities. Since 2019, some of these CWEs have been labeled by MITRE as *Prohibited* or *Discouraged*. The Prohibited tag directly forbids the use of CWEs such as categories and views, which are not valid CWE ID. These CWEs are simply lists of other CWEs which relate to a concept or share a common feature. See Figure 3 for an example CWE category and its members. On the other hand, CWE with Discouraged tag are valid weaknesses, but they do not provide enough detail to represent a clear root cause. When possible, CNAs are encouraged to find a more detailed CWE that more explicitly describes how the vulnerability came to life.

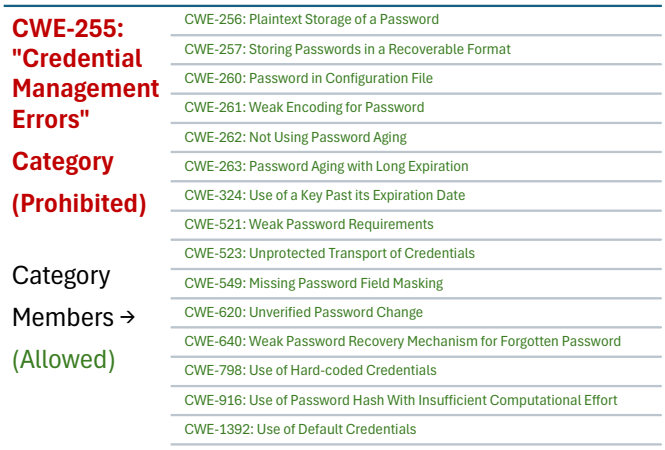


Fig. 3: An example of CWE category (CWE-255), which is Prohibited. CWE Category members are allowed for mapping and considered in our prediction.

The National Vulnerability Database (NVD) was designed to provide enrichment data on each CVE record that is submitted to the CVE list, since 2005. If relevant information is not provided by the CNA, the NVD "enriches" the CVE record via providing relevant information associated with the vulnerability, such as its root cause weakness (CWE), affected platforms (CPE), and a severity score (CVSS).

The *CWE-1003* view, also known as *Weaknesses for Simplified Mapping of Published Vulnerabilities*, consists of 130 CWE. NVD uses the CWE-1003 view for providing root cause information to CVEs, but this list has obvious shortcomings. While it is inevitable that this view is not completely representative of CWE list with 940+ unique weakness records, NVD uses two placeholder CWE, *CWE-Other* and *CWE-noinfo*, when the root cause is not found among the CWE in the CWE-1003 view. As seen in Figure 1, more than a quarter of all CVE are mapped to either of those placeholder CWEs, when a good match certainly exists in the CWE database.

2) Analysis of the NVD: In a large dataset where entries are connected in an intricate way, it is important to analyze trends within a bigger picture. For instance, one can imagine that 10 CVEs being mapped to the wrong CWE would not create much of a problem, whereas 10,000 CVEs being mapped to the wrong CWE would degrade the data accuracy significantly.

Therefore, we analyzed the CVE data in the NVD along with the history of updates, cumulatively and focusing on the updates on CVE-CWE relationships (i.e the *enrichment metadata*). We select the NVD for our analysis and methods due to its public availability and popularity, as it is being relied on by many security tools and databases. Apart from the NVD, there are public and private vulnerability databases, e.g., [8]–[10], and security advisories maintained by each CNA individually. We envision that our methods should be applicable to these databases as well.

CVE-CWE mappings are updated for many reasons, and the updates in the NVD are categorized into several groups (e.g., *CWE Remap*, *Modified Analysis*, etc.) to make it easier for processing the tens of thousands of update data. For instance, a *CWE Remap* event refers to a change in the root cause mapping of the CVE record, whereas *Modified Analysis* may mean updates to the description, or other fields, such as *CAPEC* (Common Attack Pattern Enumeration and Classification)and *KEV* (Known-Exploited Vulnerabilities) [26].

The NVD CVE History API is our primary resource. Quite simply, one can filter with CVE name or modification start and end date for fetching all the updates. We use both functionalities for obtaining the following three datasets (which were saved into .json files):

- *DATASET #1*: This dataset is a snapshot of the NVD as of August 4, 2021. This dataset is used to build our knowledge graph and train our prediction algorithm `FixV2W`.
- *DATASET #2*: This dataset is a snapshot of the NVD as of December 17, 2024, and includes every CVE record from 2002 until December 2024. In the absence of other ground truths, we assume that valid mappings in this datasets are correct (cf. Section V for further discussion). Combining Datasets #1 and #2, we extract all updates from invalid to correct mappings, which we use as a *test set* to evaluate the performance of `FixV2W`. Dataset #2 also reveals how many CVEs are still left without a root cause (CWE) mapping (that is, their mappings are invalid). We use this dataset to obtain the results shown in Figures 1 and 5.
- *DATASET #3*: This dataset records each and every update made to any CVE, between July 2016 (when the CWE-1003 list was introduced) and December 2024. We use this dataset to obtain the results shown in Figures 6 and 7.

B. Knowledge Graph Construction

We start with a background on knowledge graphs construction and evaluation terminology. Then we explain the construction and use of the knowledge graphs (KG) and embedding process and lastly, we present the automated correction process of the invalid CWE-CVE mappings, using our `FixV2W` algorithm.

1) Background and Terminology: A *Knowledge Graph* is a graph-structured representation of knowledge that captures the relationship between different entities. These graphs organize data in a machine-readable format and allow the user to connect and retrieve data as needed.

Entities are basic building blocks in a knowledge graph. They can represent any object or concept. In our knowledge graph, entities are CVE, CWE and CPE entries.

Relationships define connections between entities in a knowledge graph. This connection describes how two entities are related to each other, and in our work the relation types are `MatchingCVE`, `MatchingCWE`, `RelatedTo` (encompassing HasMember, ParentOf etc. CWE relationships). Figure 4 demonstrates these relationships.

A *triple* is the basic unit of information in a knowledge graph, consisting of a subject, a predicate (relationship) and

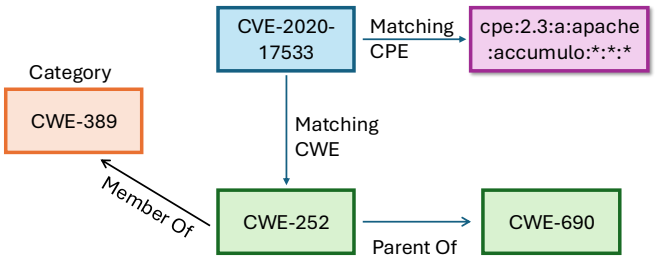


Fig. 4: Knowledge Graph slice, representing relationships between CVE (blue cells), CPE (pink cells) and CWE (green cells). Orange cells are CWE categories.

an object. We often refer to triples as mappings. An example triple as shown in Figure 4 can be given is (CVE-2020-17533, Matching CWE, CWE-252).

It is common practice to use knowledge graphs to train *embedding models*, to predict information such as unseen relationships between entities. Embedding models train with existing data, as well as via creating a set of false triples that the data is evaluated against. Embedding models train by learning the semantic architecture of the graph, the process consists of minimizing the loss function [27] over a set of triples. If the embedding model can succeed in ranking the positive (correct) triple higher than the negative (incorrect) triples (i.e. a smaller rank value), we can say that the model is successful.

The *Mean Rank (MR)* and *Mean Reciprocal Rank (MRR)* are rank-based metrics that are used to evaluate the performance of Knowledge Graph Embeddings. MR is the average of all the ranks for the triples in the test set we are evaluating. This metric is a way to get an idea about the performance of the embedding model. The ideal case is that the positive triples rank in the top positions (with smallest ranks) and the negative triples follow the positive ones. MRR is the mean of all reciprocal ranks for the triples in the test set. This metric is particularly important for understanding the distribution of ranks, namely, a perfect MRR of 1 means that all the triples in the test set are ranked at the first position, whereas a low MRR indicates that many triples have high ranks.

For a set V^T representing *the test set of CVEs*, we obtain $\text{rank}(v)$, the rank for each entry $v \in V^T$. The MRR and MR values are then calculated as follows:

$$\text{MRR} = \frac{1}{|V^T|}\sum_{v \in V^T}\frac{1}{\text{rank}(v)} \textbf{ and } \text{MR} = \frac{1}{|V^T|}\sum_{v \in V^T}\text{rank}(v).$$

2) Knowledge Graph for Correcting Vulnerability Metadata: Our KG consists of vulnerabilities (CVE) and their enrichment metadata, CWE and CPE. The CWE list itself is connected in a hierarchical way, using relationships such as `Parent Of`, `Child Of`, `Peer Of`, and using views and categories to which each CWE belongs. An example slice of this KG can be seen in Figure 4, where multiple CVE are connected to the same CPE, and multiple CPE are connected to the same CVE, via the `Matching CVE` predicate. Similarly, a CVE can have multiple `Matching CWE` predicates, and CWE can be connected to several other CWE via `Related To` or `Has Member` predicates.

We use the TransE model for predictive tasks in our work, as prior work [24] shows that this embedding model performs best in an ontology similar to ours. The TransE embedding model uses a scoring function that computes the distances between each node pair, then over several epochs, it aims to minimize the loss function that is computed using these scores. The TransE scoring function computes a similarity between the embedding of the subject translated by the embedding of the predicate and the embedding of the object, using the $L1$ norm (Manhattan Distance) [28] or $L2$ norm (Euclidean Distance) [29]. Such scoring function is then used on positive and negative triples in the loss function [30].

In our training process, we use the Adam optimizer, Multiclass NLL (Negative log-likelihood) Loss and LP regularizer [30].

3) Knowledge Graph prediction for CVE-CWE mappings: As indicated on the CWE website, we know that some CWEs are not suitable for use in root-cause mapping of vulnerabilities. These CWE are labeled *Prohibited*; and some CWE are not detailed enough for use in root cause mappings, which are labeled *Discouraged*. Prohibited CWE are categories or views that group similar weaknesses under a concept or a common element. Until recently, these were actively being used in enrichment metadata of new vulnerabilities in the NVD. Among invalid entries, Prohibited CWEs are of specific interest as it is definitely known that they are unacceptable. Similarly, in most cases the Discouraged CWEs are not acceptable due to lack of clarity and detail, therefore we demonstrate our method on both sets.

Without prior knowledge, one would use all possible CWE mappings (CWE-1003) and score those to obtain the most likely matches. However, since we know that the categories and views include relevant and valid CWE and the intended root cause mapping likely can be found among the CWE in the category/view, we narrow down the possible CWEs and run the prediction function focused on this set. Moreover, we also evaluate our model for *fine-grain* (direct neighbor) and *coarse-grain* (CWE in the same branch) predictions among the top 10 candidates returned by the embedding model.

For the rest of this section, the list of CVEs in Dataset #1 (the training set) is denoted as V^1 and the list of CVEs in Dataset #2 (the validation set) is denoted as V^2. CVE elements in these sets are denoted as $V^1 = \{v_1^1, v_2^1 \ldots, v_n^1\}$ and $V^2 = \{v_1^2, v_2^2, \ldots, v_m^2\}$ where $n = |V^1|$ and $m = |V^2|$.

The CWE list is denoted as $W = \{w_1, w_2, ..., w_k\}$ and represents the set of all existing CWEs, where $k = |W|$. The NVD's CWE slice (i.e., the CWE-1003 list) is defined as W^{NVD}. Clearly, $W^{NVD} \subset W$. We also define weaknesses in the prohibited CWE list, as W^P (note that similar sets can be defined for discouraged and other invalid CWEs). The members of this list are $w \in W^P$.

For ranking the possible matches, the KG embedding model is queried with two elements of a triple, in our case the subject

Algorithm 1 FixV2W

1: Determine $V^P = \{v \in V^1\}$ such that v is mapped to a prohibited CWE.
2: **procedure** FixV2W(V^P)
3: **for all** $v \in V^P$ **do**:
4: Determine set of allowed CWEs $W^A(v)$
5: **for all** $w \in W^A(v)$ **do**:
6: Form triple $M = (v, w)$
7: Calculate score s = $Predict(M)$
8: **end for**
9: Reorder elements in $W^A(v)$ in descending order of the score s found for each $w \in W^A(v)$
10: **end for**
11: **return** $W^A(v)$ for each $v \in V^P$
12: **end procedure**

Algorithm 2 Evaluate-FixV2W

1: Get test set $V^T = \{v \in V^1$ and $v \in V^2\}$ such that v is mapped to a prohibited CWE w_1 in Dataset #1 and to an allowed CWE w_2 in Dataset #2
2: Run procedure FixV2W(V^T)
3: **for all** $v \in V^T$ **do**:
4: Scan the sorted candidate list of weaknesses $w \in W^A(v)$, starting from the first element
5: **if** $\exists w$ such that $w = w_2$ **then**
6: Exact match found
7: Return rank of w
8: **end if**
9: **if** $\exists w$ such that w is a direct neighbor of w_2 **then**
10: Fine grain match found
11: Return rank of w
12: **end if**
13: **if** $\exists w$ such that w is in the same branch as w_2 **then**
14: Coarse grain match found
15: Return rank of w
16: **end if**
17: **end for**
18: **return** Accuracy, MR and MRR of exact matches, fine grain matches, and coarse grain matches

and predicate. The model calculates the scores for all possible completions. When all possible matches are scored, these are sorted from highest score to lowest, and positions in which each triple are is called the *rank*. The triple with the highest score is returned at rank #1.

For the task of predicting the correct CWE mapping for a CVE, we define V^P as all CVEs that were mapped to an invalid CWE in Dataset #1. For each $v \in V^P$, we consider a candidate list $W^A(v)$ of allowed CWEs and use this set for the prediction function using the KG embedding model. Note that $W^A(v) \subset W^{NVD}$ (e.g., we can restrict to CWEs that are within the same branch or are children of the Prohibited CWE used in Dataset #1). The prediction function runs for each CVE v in V^P and each CWE $w \in W^A(v)$ and calculates the score s for each mapping $M = (v, w)$. Last, the algorithm returns the reordered list of weakness candidates $W^A(v)$ sorted in descending order, where the CWE with the highest score is returned at the first position, the next largest at the second and so on.

The $Predict(M)$ function is a method that the KG embedding uses to calculate the score of a given mapping (triple). Given a subject and predicate, we can calculate the score for triple completion run on the object. Similarly, if the predicate was unknown, we can run the same function (knowing subject and object) to obtain a score for the test triple. We used Ampligraph [27] Python package for the implementation of our method, which provides the embedding model functions.

a) Example run of Algorithm 1: Consider v = CVE-2013-1913, a vulnerability that was mapped to an invalid category CWE-189 *Numeric Errors* in Dataset #1. In Dataset #2, this CVE mapping was corrected to CWE-190 *Integer Overflow or Wraparound*, which is allowed and the accurate root cause for this vulnerability.

We start with extracting the members of the allowed category (line 4), which returns the following list: $W^A(v)$ = {CWE-128, CWE-190, CWE-191, ..., CWE-1389}. For each item in this list, we form a triple, and calculate the scores s for each of these triples using the embedding model (line 8):

- $Predict$(CVE-2013-1913, CWE-128) $= -11.15623$
- $Predict$(CVE-2013-1913, CWE-190) $= -10.707506$
- ...
- $Predict$(CVE-2013-1913, CWE-1389) $= -12.5468$

Lastly, we order the scores in descending order (line 9), with the first item having the highest rank and being strongest candidate. Since `FixV2W` determines that CWE-190 is the triple with the highest score, we mark this prediction as an exact match of top rank, following Algorithm 2.

IV. Results and Evaluation

In this section, we present our results and key takeaways obtained by the longitudinal analysis and the methods for fixing invalid mappings.

A. Longitudinal Analysis of CVE data

Our analysis of CVE-CWE mapping updates reveals that the NVD has many erroneous data, which still requires fixing. in particular, we detect many CVEs with *Discouraged* or *Prohibited* CWE mappings. Although prioritizing new updates is understandable and undeniably more urgent, we discuss in this section that many of these errors can be easily corrected with the help of our algorithms.

a) Invalid or missing CWEs: Using Dataset #2, we classify CVEs according to the years they were submitted. Figure 5 shows that between 2008 and 2024, between 15% and 30% of CVE reported each year ends up without a valid CWE mapping. Moreover, among the 280,000+ CVE that currently exist in the database, 18.4% have empty CWE metadata, while 10.8% are mapped to CWE-Other and 10% are mapped to

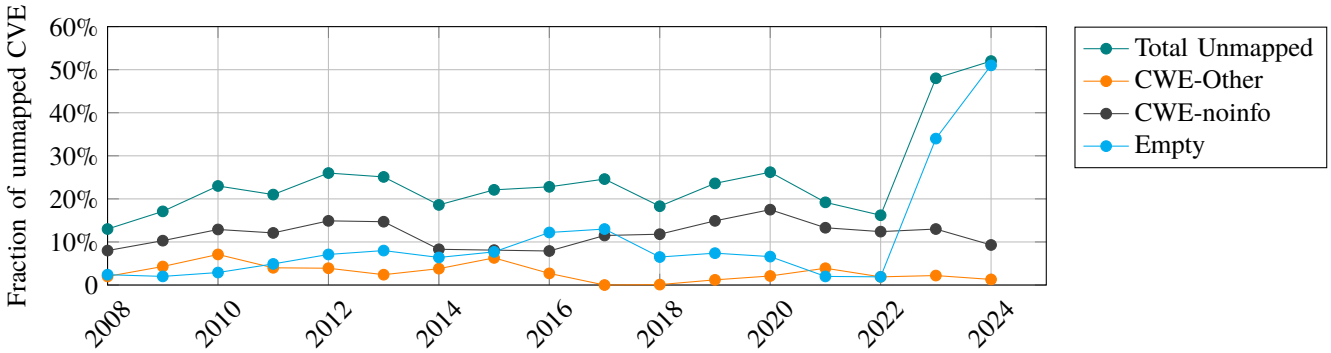


Fig. 5: The ratio of CVE without a specific CWE mapping for CVE published between 2008 and 2024.

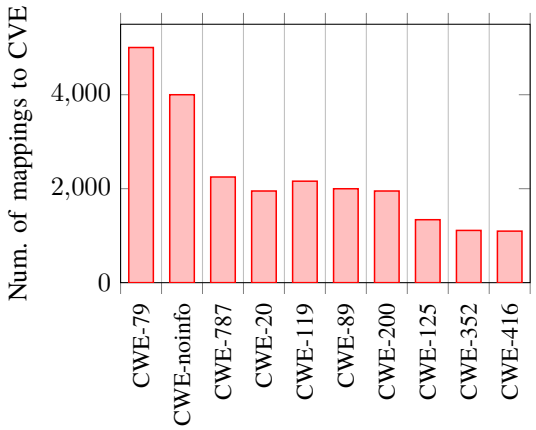


Fig. 6: Most occurring newly mapped CWEs between 2016-2024. From left to right, top 3 are CWE-79: Cross-Site Scripting, CWE-noinfo and CWE-787: Out-of-bounds Write.

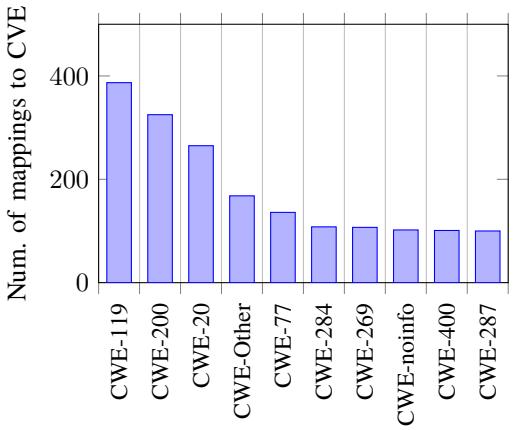


Fig. 7: Most removed CWEs between 2016-2024, from left to right. The top 3 are CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer, CWE-200: Exposure of Sensitive Information to an Unauthorized Actor, and CWE 20: Improper Input Validation.

CWE-noinfo (see Figure 1). Moreover, a total of 16.2% of all CVEs are mapped to either Prohibited or Discouraged CWEs.

b) The use of CWE-Other and CWE-noinfo: A total of 4000 updates between July 2016 and December 2024 were remaps into `CWE-noinfo`. In fact, Figure 6 that among the top 3 CWE used to provide root cause mappings to CVEs, `CWE-noinfo` is number 2. While `CWE-Other` indicates that the appropriate CWE mapping may exist, but is not found in the subset (`CWE-1003`), `CWE-noinfo` provides no information about the root cause of CVE whatsoever. This brings up the question if the CWE-1003 list is adequate. Considering that the list was formed in 2016, and last updated in 2019, an update may be in order for more accurate root cause mapping.

c) Commonly misused CWE: Figure 7 shows that certain CWEs are more often misused, from most often misused to less: `CWE-119`, `CWE 200`, `CWE-20` and so on. Mappings of CVEs to those CWEs are also updated more often. All of these CWE are now labeled `DISCOURAGED` for vulnerability mapping since they lack specificity, therefore cannot tell much about a root cause of a CVE. We present and evaluate our approach for fixing those in Section III-B3.

d) Updated CWE mappings that are related: For the rest of the CVE remaps to valid CWE, 60% of remaps end up at most two hops away from the old CWE value, whereas more than 90% are in the same branch. Some examples are shown in Table I. This means that the correct CWE can be reached from the old incorrect CWE, via searching the related weakness chains, and usually going from parent to child i.e. from less to more detailed CWE within the same chain [31]. Leveraging this clue, we designed our `FixV2W` algorithm on searching for the correct CWE mapping within the same branch.

e) CWE and CVSS score relationship: CVSS scores are generally updated along with the CWE mappings, and sometimes afterwards. Among the 1000+ updated CVE-CWE mappings that also had a CVSS score update, between 2016 and 2024, we see that 729 of them (68%) was the first time this CVE ever had a CVSS score. The average CVSS score is 7.4 (High) with more than 60 Critical severity score instances. For the remaining 328 pairs (31%), we saw that 250 CVSS scores (23%) decreased and 69 of them (6%) increased. Our results indicate that correct root cause detection can inform

TABLE I: The most observed Old CWE to New CWE pairs for the same CVE Record, that were updated between 2018 and 2022. Pairs are either in the same chain for valid CWE, or the new CWE may be a member of the old CWE category.

CWE ID Pair	Occurrences	Same Chain	Is Member
CWE-Any - CWE-noinfo	38%	-	-
CWE-119 - CWE-787	6%	Yes	-
CWE-77 - CWE-78	3%	Yes	-
CWE-119 - CWE-125	2%	Yes	-
CWE-264 - CWE-732	1%	-	No*
CWE-255 - CWE-522	1%	-	Yes
CWE-264 - CWE-269	1%	-	No*
CWE-416 - CWE-787	1%	Yes	-
CWE-190 - CWE-787	1%	Yes	-
CWE-284 - CWE-732	1%	Yes	-
CWE-399 - CWE-772	1%	-	Yes

* CWE-264: "Permissions, Privileges, and Access Controls" is a category with no members.

CVSS scoring, and is an important resource for understanding factors that affect the severity of vulnerabilities

To summarize, we see several trends in CVE change history data and in the NVD that can inform our predictions for correcting and automating CVE-CWE mappings. It is no surprise that there are errors within a database at this scale. Our analysis specifically sheds light into problematic root cause weakness mappings of CVE.

B. Knowledge Graph Predictions for CVE-CWE mappings

In this section we evaluate our methodology `FixV2W` for automatically fixing invalid CVE-CWE mappings. In Section III-B, we described the data and ontology for the creation of our knowledge graph. Using this knowledge graph, we next move on to evaluation and results.

a) Correcting Prohibited CVE-CWE mappings: After processing the results for the predictions provided by the embedding model, we calculated several accuracy metrics for the test set.

For evaluation, we detected 10026 CVE that were mapped to prohibited CWEs in the Dataset #1, and only 502 of them (2%) were updated to valid CWEs in Dataset #2. The updated CVE-CWE mappings are used as a test set to validate the predictions returned by `FixV2W`. We compare the updated (correct) CWE to the top 10 candidates returned by the embedding model, sorted by their scores. If the correct CWE is found among the candidate list, we flag this as the exact match, and later evaluate how good this match is via calculating MR, MRR and Hits@N metrics.

Otherwise, the candidates in the list are evaluated for their closeness to the correct CWE, depending on whether they are a direct parent/child or they are in the same branch. If fine-grain or coarse-grain matches were found, we calculate again the Mean Rank (MR) and Mean Reciprocal Rank (MRR) and Hits@N metrics to evaluate model's performance, as described in Algorithm 2.

Among the test set of 502 Prohibited CVE-CWE mappings, `FixV2W` is able to predict the exact match as the first prediction for 52% of CVEs, and among the top 10 results for 65% of the CVEs. In Table II, we show an example prediction result for an exact match, for CVE-2013-1913 which was previously mapped to CWE-189 (*Numeric Errors*) Category. This CVE later was updated to CWE-190 (*Integer Overflow or Wraparound*) weakness, which is a member of the CWE-189 category, and the model predicted the correct update at the top position. The significance of the exact match prediction is that these results require minimal to no manual analysis afterwards, considering that more than half of the correct mappings were predicted at the top rank. Among those CVE where the model has returned an exact match, the distribution of the ranks is shown in Figure 8(a).

TABLE II: CWE mapping prediction results for CVE-2013-1913. Higher score yields higher ranking. CWE-190 is the updated mapping for this CVE as of December 2024. `FixV2W` ranks this CWE at the top of its candidate list.

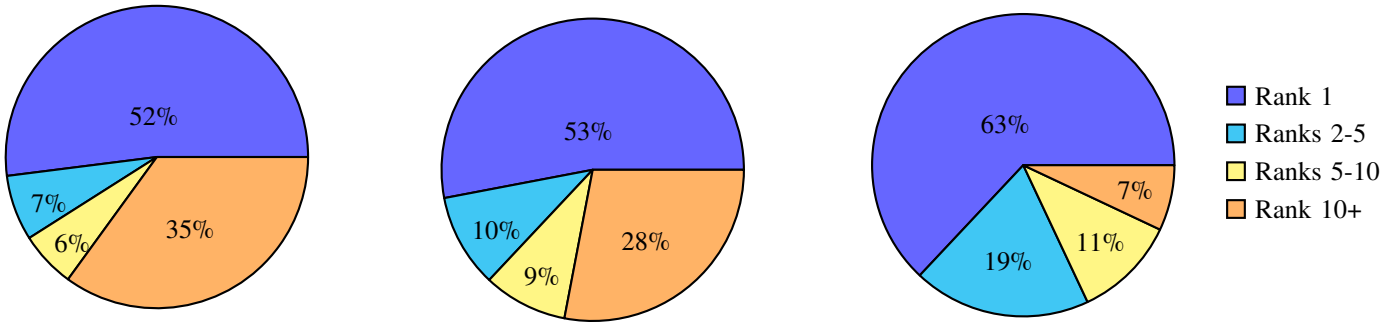
Rank	Predicted CWE	Score
1.	**CWE-190**	**-10.707**
2.	CWE-476	-10.844
3.	CWE-125	-10.949
4.	CWE-119	-11.075
5.	CWE-79	-11.156
⋮		

For the rest of the test set where 35% of the CVE-CWE mappings were not predicted within the top 10 positions, we performed a fine-grain and coarse-grain evaluation, as explained in Section III. We realized that most of the time, the direct parents/children or neighbors in the same CWE branch were predicted by the KG instead. In Figure 8(b), we see that among the remaining 35%, a direct parent or child was predicted for 7%, taking the unpredicted slice down to a 28%. Lastly, we see in 8(c) that for those CVE where an exact match or a fine-grain match cannot be found, a CWE in the same branch has been predicted for 28%, resulting in an overall 7% remaining CVE that our model could not return any related CWE at all.

To summarize, for the ability to predict the correct CWE within the top 10 positions, we report the exact match prediction rate at 65%, fine-grain at 72% and coarse-grain at 93%. Moreover, we see that most of the correct predictions are found at the first rank.

- For the exact match predictions in the top-10 ranks (327 cases), we achieve a MR of 1.862 and MRR of 0.848.
- For the fine-grain predictions in the top-10 ranks (327 exact match + 38 direct parent/child), we achieve a MR of 2.175 and MRR of 0.797.
- For the coarse-grain predictions in the top-10 ranks (327 exact match + 141 CWE in branch), we achieve a MR of 2.252 and MRR of 0.765.

When it comes to the rest of the 7% of CWE that the model was not able to predict, we find that all of the prohibited



(a) Exact match prediction (b) Fine-grain (c) Coarse-grain

Fig. 8: Distribution of ranks for CVE-CWE prediction of Prohibited mappings. At rightmost diagram (c), for the 7% remaining "unpredicted" mappings, old CWE category has no members in the NVD CWE slice.

CWE that CVE were previously mapped have *no candidate weaknesses among the members of the prohibited category or view that belong to the CWE-1003 list*. Since the only possible values for CWE mappings are in the CWE-1003 list, this leaves the model no possible CWE to choose from, therefore it cannot provide a prediction.

b) Correcting Discouraged CVE-CWE mappings: Similar to the task for correcting *Prohibited* CVE-CWE mappings, we calculated the model performance for exact match, fine-grain and coarse-grain predictions. Overall, 1591 *Discouraged* mappings were detected in Dataset #1 where the old mapping was updated by December 2024 (in Dataset #2). Among these, 877 were to other invalid mappings (nearly 50%). For the remaining 715 and considering again the top 10 ranked answers, `FixV2W` was able to find the exact match for 127 CVE, and coarse grain match for 247 of them. Thus, 331 unique records were successfully predicted by the model. We see again that the correct predictions are mostly returned at the top position especially for the exact match set.

- For the exact match predictions, we achieved a MR of 1.039 and MRR of 0.980.
- For the fine-grain predictions our results are the same as exact match, i.e. the model did not find any direct parent or children if the exact match was not found.
- For the coarse grain predictions (127 Exact match + 247 CWE in branch) we achieved a MR of 1.400 and MRR of 0.901.

Since the model could return a prediction for 374 CVE-CWE queries among 715, this puts the overall prediction rates to: 17% for exact match and fine-grain prediction and 52% for coarse grain prediction. However, when we examine the set of CVEs that the model cannot predict, we see that none of the CWE in the old invalid mappings had any children in the CWE-1003 list. In this case, it is a better idea to use the entire CWE-1003 list for prediction.

Excluding this set of CVEs, where the CWE options for the predictions are an empty set, we can report a prediction rate of 38% for exact matches and 100% for coarse-grain matches.

C. Performance & System Requirements

There exist set of requirements for easy replication of our model such as computational compatibility and resources. Our team worked on a shared computing cluster provided by Boston University, which includes 2 sixteen-core 2.8 GHz Intel Gold 6242 processors. In one of the processors, we used up to 8 cores in the parallelization of our code. We used Tensorflow 1.13.1 version and Python 3.7.10. Along with these, we used an older version of Ampligraph (1.4.0) since the 2.x versions did not include some of the functions that were needed in our methods. For scalability, our codes can be parallelized and quries can be run in parallel, if the system allows. For instance, for the test set of 773 Discouraged CVE-CWE mappings, FixV2W predictions were completed in 32.1 minutes on a single CPU, and when we utilized 32 parallel processors, the queries were completed in just 6.51 minutes. On an A1000 GPU (CUDA), the queries completed in 4.11 minutes and two parallel GPUs completed all 773 queries in 2.1 minutes.

We obtain our data from the NVD via API requests and process those for easy access in the code and to refrain from loading unnecessary data to the program memory. We processed CVE-CWE mappings, the query test set and CWE data, which are all included in our tutorial folder. The source codes and the files that we processed (and are required to run the codes) can be found in our GitHub page [32].

V. Limitations and Threats to Validity

a) Dataset Accuracy: For both our predictions and validations, we rely on the NVD. We begin our analysis with the assumption that the NVD contains errors, yet we still use it to train our model. To address this, we are working on integrating "corrected" data into our knowledge graph (KG) and re-training the models. This process is time-intensive and requires ongoing updates.

Additionally, the latest dataset from the NVD serves as our validation set (i.e., we assume that valid mappings in that dataset are correct). Since there may still be errors in the updated data, the accuracy of our results is closely tied to the accuracy of the NVD. Looking ahead, we plan to explore more

TABLE III: Example of a CVE which was updated by the NVD from a Discouraged CWE (CWE-269) to another Discouraged CWE (CWE-697). The FixV2W method predicts a more accurate CWE mapping at the first rank.

CVE	CVE-2021-23999	If a Blob URL was loaded through some unusual user interaction, it could have been loaded by the System Principal and granted additional privileges that should not be granted to web content. This vulnerability affects Firefox ESR <78.10, Thunderbird <78.10, and Firefox <88.
CWE-NVD	CWE-697: Incorrect Comparison	The product compares two entities in a security-relevant context, but the comparison is incorrect, which may lead to resultant weaknesses.
CWE-FixV2W	CWE-271: Privilege Dropping / Lowering Errors	The product does not drop privileges before passing control of a resource to an actor that does not have those privileges.

reliable validation methods, but for this study, we acknowledge this as a limitation.

b) Results Evaluation: The fine-grain and coarse-grain predictions are valuable results that greatly reduce the need for manual analysis. However, since the exact CWE was not returned by the model, it is necessary to find the exact match either manually or with another layer of automation. On the other hand, we are assuming that the updated CWE is the best match for the CVE, provided by the NVD or CNA. While we hope that this is the case, at least for the accuracy of the validation, we realize this is not always the case and a better mapping exists in the same branch that was not provided by the NVD. Consider CVE-2021-23999, the NVD updated its mapping from Discouraged CWE-269: *Improper Privilege Management* to another Discouraged CWE, CWE-697: *Incorrect Comparison*. As seen in Table III, CWE-271: *Privilege Dropping / Lowering Errors* is a better and more detailed CWE that fits the CVE description, since the System Principal does not drop privileges when passing control to the web content, and is returned as the best prediction from our model. While this is not always the case, we see instances where FixV2W appears to be more accurate than the current CWE mapping provided by the NVD, used as the ground truth.

VI. Conclusion

In this work, we introduced FixV2W, a simple, yet effective approach for correcting invalid mappings in the NVD, which can be applied to broader vulnerability database applications. We found that more than half of the CVE in the NVD database have CWE mappings that need to be fixed. FixV2W is based on a knowledge graph representation of the NVD database, on which we train an embedding model to predict hidden relationships. We leverage our findings from the longitudinal study to inform predictions for better CVE-CWE mappings. Namely, if a CVE was mapped to an invalid CWE, the children of that CWE (for Discouraged CWE) or members belonging to the same category or view (for Prohibited CWE) are good candidates.

We show that, after filtering, we can predict the correct mapping, either exact match or fine-grain or coarse-grain matches, with high accuracy. In many cases, the correct match appears at the first or second rank of the list returned by FixV2W.

We envision that the FixV2W method will be used as the first step in fully-automating the CVE-CWE mapping correction task, for increasing the accuracy of the database as well as many security analysis tools that rely on the NVD and other vulnerability databases. Since the coarse-grain prediction does not give an exact root cause prediction for the CVE, we will be working on combining other methods with our fine and coarse-grain predictions, for accuracy improvements and for applying our method to the rest of the CVE-CWE mappings that remain incorrect.

Acknowledgments

This work was supported in part by the Boston University Red Hat Collaboratory (awards numbers 2024-01-RH03 and 2025-01-RH05).

References

[1] Snyk, "Snyk Open Source," https://docs.snyk.io/scan-application-code/snyk-open-source, 2023.
[2] Mend.io, "Mend SCA," https://www.mend.io/sca/, 2023.
[3] The OWASP® Foundation, "OWASP Dependency Check," https://owasp.org/www-project-dependency-check/, accessed:2024-05-01.
[4] NIST. (2002) National Vulnerability Database. [Online]. Available: https://nvd.nist.gov/
[5] MITRE. (2024) Common Vulnerability and Exposures. [Online]. Available: https://cve.mitre.org
[6] MITRE, "Common weakness enumeration (CWE)," https://cwe.mitre.org, 2024, accessed: 2024-04-30.
[7] NIST. (2024) Common Platform Enumeration. https://nvd.nist.gov/products/cpe.
[8] Snyk OS, "Snyk vulnerability database," https://snyk.io/vuln.
[9] Red Hat Inc., "RedHat CVE Database," https://access.redhat.com/security/security-updates/cve, 2025.
[10] Mend.io, "Mend vulnerability database," https://www.mend.io/wp-content/media/2021/11/Mend-Vulnerability-Database.pdf, 2023.
[11] S. Schuberth, P. Ombredanne, J. Kowalleck, W. Bartholomew, S. Springett, and J. M. Horan, "Package-url," https://github.com/package-url/purl-spec, 2017.
[12] 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.
[13] MITRE, "CWE Resources & Entry Structure," https://cwe.mitre.org/documents/cwe_usage/guidance.html#resources.
[14] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.
[15] MITRE. (2024) CWE Research Concepts View. https://cwe.mitre.org/data/graphs/1000.html. Accessed = 09-01-2024.
[16] H. Yang, Z. Lin, and M. Zhang, "Rethinking knowledge graph evaluation under the open-world assumption," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 8374–8385. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/378226e5df7eded3e401de5c9493143c-Paper-Conference.pdf
[17] X. Wu, W. Zheng, X. Xia, and D. Lo, "Data quality matters: A case study on data label correctness for security bug report prediction," *IEEE Transactions on Software Engineering*, vol. 48, no. 7, pp. 2541–2556, 2022.

[18] 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*, A. Hameurlain, S. W. Liddle, K.-D. Schewe, and X. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 217–231.

[19] V. H. Nguyen and F. Massacci, "The (un)reliability of nvd vulnerable versions data: An empirical experiment on google chrome vulnerabilities," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 493–498. [Online]. Available: https://doi.org/10.1145/2484313.2484377

[20] P. Kuehn, M. Bayer, M. Wendelborn, and C. Reuter, "Ovana: An approach to analyze and improve the information quality of vulnerability databases," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ser. ARES '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3465481.3465744

[21] E. Aghaei, W. Shadid, and E. Al-Shaer, "Threatzoom: Cve2cwe using hierarchical neural network," *arXiv preprint arXiv:2009.11501*, 2020.

[22] M. Albanese, O. Adebiyi, and F. Onovae, "Cve2cwe: Automated mapping of software vulnerabilities to weaknesses based on cve descriptions," *In Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024), pages 500-507*, 2024.

[23] D. Alfasi, T. Shapira, and A. B. Barr, "Unveiling hidden links between unseen security entities," https://arxiv.org/abs/2403.02014, 2024.

[24] Z. Shi, N. Matyunin, K. Graffi, and D. Starobinski, "Uncovering cwe-cve-cpe relations with threat knowledge graphs," *ACM Trans. Priv. Secur.*, vol. 27, no. 1, feb 2024. [Online]. Available: https://doi.org/10.1145/3641819

[25] C. Madden, "ChatGPT CWE GPT," https://cybersecai.github.io/CWE_Assignment/cwe_gpt.

[26] NIST. (2024) Vulnerability APIs. [Online]. Available: https://nvd.nist.gov/developers/vulnerabilities

[27] L. Costabello, S. Pai, C. L. Van, R. McGrath, N. McCarthy, and P. Tabacof, "AmpliGraph: a Library for Representation Learning on Knowledge Graphs," Mar. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2595043

[28] P. E. Black, "Manhattan distance," *Dictionary of Algorithms and Data Structures*, 2019. [Online]. Available: https://www.nist.gov/dads/HTML/manhattanDistance.html

[29] Paul E. Black, "Euclidean distance," *Dictionary of Algorithms and Data Structures*, 2004. [Online]. Available: https://www.nist.gov/dads/HTML/euclidndstnc.html

[30] L. Costabello, S. Pai, C. L. Van, R. McGrath, N. McCarthy, and P. Tabacof, "Knowledge graph embedding models," https://docs.ampligraph.org/en/1.3.2/ampligraph.latent_features.html.

[31] MITRE. (2024) Chains and composites. https://cwe.mitre.org/data/reports/chains_and_composites.html#chains. Accessed = 09-01-2024.

[32] S. Simsek, "Threat knowledge graphs - FixV2W," https://github.com/nislab/threat-knowledge-graph.