Vulnerability Intelligence Alignment via Masked Graph Attention Networks

Yue Qin
Indiana University Bloomington
Bloomington, Indiana, USA
qinyue@iu.edu

Yue Xiao
Indiana University Bloomington
Bloomington, Indiana, USA
xiaoyue@iu.edu

Xiaojing Liao
Indiana University Bloomington
Bloomington, Indiana, USA
xliao@indiana.edu

ABSTRACT

Cybersecurity vulnerability information is often sourced from multiple channels, such as government vulnerability repositories, individually maintained vulnerability-gathering platforms, or vulnerability-disclosure email lists and forums. Integrating vulnerability information from different channels enables comprehensive threat assessment and quick deployment to various security mechanisms. However, automatic integration of vulnerability information, especially those lacking decisive information (e.g., CVE-ID), is hindered by the limitations of today's entity alignment techniques.

In our study, we annotate and release the first cybersecurity-domain vulnerability alignment dataset, and highlight the unique characteristics of security entities, including the inconsistent vulnerability artifacts of identical vulnerability (e.g., impact and affected version) in different vulnerability repositories. Based on these characteristics, we propose an entity alignment model, CEAM, for integrating vulnerability information from multiple sources. CEAM equips graph neural network-based entity alignment techniques with two application-driven mechanisms: *asymmetric masked aggregation* and *partitioned attention*. These techniques selectively aggregate vulnerability artifacts to learn the semantic embeddings for vulnerabilities by an asymmetric mask, while ensuring that the artifacts critical to the vulnerability identification are always taken more consideration. Experimental results on vulnerability alignment datasets demonstrate that CEAM significantly outperforms state-of-the-art entity alignment methods.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**;
• **Security and privacy**;

KEYWORDS

Entity Alignment; Vulnerability Intelligence; Knowledge Graph Alignment; Graph Attention Networks; Vulnerability Repository Inconsistency

ACM Reference Format:
Yue Qin, Yue Xiao, and Xiaojing Liao. 2023. Vulnerability Intelligence Alignment via Masked Graph Attention Networks. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3576915.3616686

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3616686

1 INTRODUCTION

There is a wealth of cybersecurity vulnerability information available through various channels: public vulnerability databases (e.g., the National Vulnerability Database [71]), individual-maintained vulnerability-gathering platforms (e.g., SecurityFocus [14]), security advisories of different vendors (e.g., Palo Alto Networks Security Advisories [41], Android Security Bulletins [30]), vulnerability disclosure email lists [15, 17–19] and forums [42], and many others. These vulnerability repositories contain a range of vulnerability artifacts, such as vulnerability type, affected device information, vulnerability severity score. This contextual information is essential for security practitioners to prioritize vulnerability remediation and preventing attacks.

However, the vulnerability artifacts of an identical vulnerability typically vary across different vulnerability repositories. As an example, the Linux kernel evolves quickly and has a large number of versions and derivatives such as Android, Ubuntu, Red Hat, and various IoT systems. However, for the Linux kernel-related vulnerabilities, the NVD often only records the vulnerability artifacts associated with limited Linux kernel versions and derivatives, while for identical Linux kernel vulnerabilities, the vulnerability artifacts (e.g., affected device and version) recorded in the security advisories of mobile device vendors can supplement those in the NVD [25]. Given an organization with both Red Hat and Android devices, without comprehensive vulnerability artifacts from both NVD and the Android security advisory, once a Linux kernel vulnerability was found, the organization cannot fully respond in a timely manner. Hence, integrating vulnerability information from different channels is essential for an organization to gain comprehensive and credible vulnerability information associated with different devices and OS derivatives, identify early signs of cybersecurity risk, and effectively contain the threat with proper means.

Challenges in vulnerability alignment. However, it is non-trivial to link the same vulnerabilities among different sources, especially for those newly-reported vulnerabilities without unique identifiers (e.g., CVE-ID), or those from vulnerability reports with less-structured format. For instance, news reports [4, 7] indicate that over 42% of vulnerabilities listed in VulnDB [5] do not have CVE-IDs. This creates a situation where vulnerable software products may remain unmaintained and untracked, making it challenging for security engineers to detect and fix vulnerabilities that might impact their products. IT professionals have also expressed concerns [10, 20] about the lack of CVE-IDs, which hinders them from

cross-referencing vulnerabilities documented by other repositories. Moreover, widely used scanning tools rely on vulnerability databases indexed by CVE-IDs to determine if products contain vulnerable software [6]. Additionally, as reported in [54], the vulnerabilities in the IoT vulnerability disclosure forums were typically released before obtaining CVE-IDs. Despite the lack of CVE-IDs, these forums often contain valuable vulnerability artifacts, such as proof of concept (PoC), which is crucial for efficient vulnerability assessment and patch generation. However, multiple commonly-used public vulnerability repositories, such as the National Vulnerability Database (NVD), do not provide such detailed information. To this end, organizations have expended significant human efforts to establish links between those IoT vulnerabilities and those in the NVD.

The critical aspect of vulnerability information integration and management is linking security entities (i.e., vulnerabilities) that pertain to the same real-world entity across various data sources. This is a form of entity alignment (EA) problem [79] that arises in vulnerability repositories, when the repositories can be represented as vulnerability knowledge graphs (KG) that programmatically constructed from structured or semi-structured vulnerability reports. However, to the best of our knowledge, no previous research has explored cybersecurity entity alignment techniques tailored to the application of vulnerability artifact integration.

CEAM: design and implementation. Traditional entity alignment methods, which rely on hand-crafted rules, are often less effective in aligning security entities across different KGs due to variations in their structures and textual features [92]. Recent work [59] uses Knowledge Graph Embedding (KGE) models, trained to measure triple plausibility (e.g., TransE [46]) to align equivalent entities into a unified vector space based on a few seed alignments. However, such methods are not suitable for security entity alignment as KGE models cannot generate embeddings for new entities added to the KG after training. In the context of vulnerability information, timely updates are crucial, and it is impractical to retrain the KG embedding model on the entire augmented graph each time new security entities, such as vulnerabilities, are discovered. Mao et al. [67] consider entity alignment as an assignment problem between two isomorphic graphs by reordering the entity node indices. However, adapting such techniques for security entity alignment is challenging due to the significant differences in graph topology between security KGs constructed from different repositories, as observed in our measurements in §3.4.

In recent years, Graph Neural Network (GNN) has shown great success in open-domain entity alignment tasks [68, 70, 88]. This mechanism allows for recursive information propagation among neighbors to learn structure-aware entity representations. However, their core assumptions that identical entities have similar attributes and neighbors and vice versa do not hold for cross-platform security entities. In our study, we observe *identical vulnerability shows inconsistent attributes (vulnerability artifacts, e.g, impact and affected version) in different vulnerability repositories (see §3.4)*. The main reason is that a vulnerability is sometimes assessed considering different execution environments such as operating systems, architectures, configurations, and organization policies, which can lead to different vulnerability artifacts. In addition, different repositories provide vulnerability artifacts in different granularity (e.g., level of details), especially for vulnerabilities disclosed in maillists and forums. We also observe that different vulnerabilities can be associated with a considerable number of identical artifacts, leading to false positives of the alignment (see §3.4).

Given the aforementioned observations in the application of vulnerability alignment, we propose an entity alignment model, CEAM, tailored for the cybersecurity domain. It equips GNN-based entity alignment model with two application-driven designs: *asymmetric masked aggregation* and *partitioned attention*, to address the above challenges. We first aggregate *selective* attribute information to learn the semantic embeddings for security entities by an asymmetric mask. It computes similar representations for the same vulnerabilities with inconsistent artifacts in different repositories, by scaling down a partial representation of the inconsistent relation (e.g., has_product). This will alleviate the false negatives caused by inconsistencies between positive pairs. Further, we use GNNs to update entity embeddings with structural information based on graph topology, where the partitioned attention mechanism ensures that the artifacts critical to the vulnerability identification are always taken more consideration during the propagation, which cannot be guaranteed by traditional neural networks. Finally, we use two-layer MLP (Mutilayer Perceptron) to decide whether two entities are identical according to the discrepancy between entity embeddings learned by GNNs.

We have implemented CEAM and evaluated it on two annotated entity alignment datasets. We found that CEAM achieves the precision of 73.4%, the recall of 91.7% and the F1 score of 81.5%, which outperform the state-of-the-art entity alignment models CG-MuAlign [95], PARIS [79] and PRASE [75]. Particularly, our experiments show that the proposed two innovative mechanisms *asymmetric masked aggregation* and *partitioned attention* collectively improve alignment quality by 10.3% of F1 score on average.

Contributions. The contributions of this paper are as follows:

- We proposed an entity alignment model, named CEAM, tailored for the application of vulnerability inconsistency identification across different vulnerability repositories.
- We released the *first* annotated datasets for cybersecurity-domain entity alignment, and unveil their characteristics that challenge the assumption made in traditional entity alignment tasks [45, 88, 95]. These characteristics include inconsistencies in vulnerability artifacts for identical vulnerabilities across different repositories, as well as similarities in artifacts for different vulnerabilities.
- We discussed two potential applications of CEAM: (1) supplementing vulnerability artifacts across repositories and (2) debunking erroneous vulnerability artifacts.
- Our code, datasets, and full-version paper with Appendix are available at [40].

2 BACKGROUND

2.1 Vulnerability Artifacts

In Table 1, we present examples of common vulnerability artifacts used in our study. Specifically, *CVE* is the common identifiers of cybersecurity vulnerabilities. *Weakness* characterizes the category of the vulnerability, and *CWE_ID* is the identifier of Weakness. *Product* and *Vendor* are the name and the provider of the affected products (e.g., software, hardware, device, etc). *Version* is short for affected

versions. *Impact* is the consequence of exploiting the vulnerability. *Discoverer* is the name of the person or the organization who reported the vulnerability. Note that there exists relation between vulnerability artifacts. For example, the relation between a *Vulnerability* entity and a *Discoverer* is *hasDiscoverer*, which can be denoted as a triplet (h, r, t) where the relation r is decided by the types of the head entity h and the tail entity t and is in the form of *hasTail*.

CVSS and CVSS metrics. The Common Vulnerability Scoring System (CVSS) is an open and widely-adopted vulnerability severity scoring standard [34], which suggests various kinds of critical vulnerability artifacts associated with vulnerability severity. More specifically, in CVSSv3.1, vulnerability artifacts consist of eight dimensions in the base metrics: *Attack Vector* (AV), *Attack Complexity* (AC), *Privileges Required* (PR) and *User Interaction* (UI), *Scope* (S), *Confidentiality* (C), *Integrity* (I) and *Availability* (A). For instance, the artifact of AV reflects the context by which vulnerability exploitation is possible (e.g., remotely exploit); and the artifact of AC describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability.

There also exist artifacts which measure the current state of exploit techniques or code availability, namely temporal metrics, i.e., *Exploit Code Maturity* (E), the existence of any patches or workarounds, i.e., *Remediation Level* (RL), or the confidence in the description of a vulnerability, i.e., *Report Confidence* (RC). A *CVSS metric value* (e.g., *AV:N*) represents a pair of a CVSS metric and its value, i.e., the *Attack Vector* to be *Network*. The artifact of the CVSS score is calculated according to a corresponding *CVSS vector* aggregating CVSS metric values. For example, a CVSSv3.1 base score of *7.5* is calculated given the CVSS vector *(AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H)*. The formulas to calculate the base and temporal CVSS scores can be found in [35]. Note that while the current version of CVSS score is v3.1, many vulnerability reports still use v2 and v3.0. Despite this, both v3.1 and v3.0 share the same CVSS metrics and score calculation formula. However, v3.1 provides additional assessment guidance by updating the CVSS document specification.

Vulnerability profiling standard. According to CNA [33], to request an identifier for a newly-found vulnerability, certain information is *required*, such as the vulnerability type, vendor, and affected equipment. Additional information, such as impact, attack patterns, and discoverer, are also *encouraged*. In our research, we refer to these required and encouraged pieces of information as *profiling* artifacts (i.e., vendor, affected equipment, weakness, etc.). Other vulnerability artifacts like CVSS score, CVSS vector are considered as *non-profiling* artifacts. In our proposed system, the profiling artifacts are emphasized in a mandatory way by a partitioned attention mechanism during the aggregation of neighborhood information to update the representation of a target entity (§5.3).

2.2 GNN-based Entity Alignment

Most GNN-based entity alignment methods [45, 68, 95] are subject to the following framework: (1) a GNN to learn node representations from graph structure and (2) a margin-based loss to rank the distance between entity pairs. The loss function

$$L = \sum_{(i,j)\in\mathcal{P}} \sum_{(i',j')\in\mathcal{P}^-} \max\left\{\mathrm{d}(h_i, h_j) - \mathrm{d}(h_{i'}, h_{j'}) + \gamma, 0\right\}$$

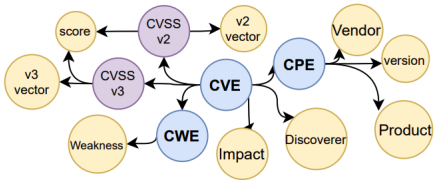

Figure 1: KG schema. Blue: entities; Yellow: literal artifacts; Purple: intermediate nodes.

aims at making equivalent entities (i, j) close to each other while maximizing the distance between negative pairs (i', j'). Here h_i is the embedding of entity i updated by a GNN layer in the form of:

$$h_i^{l+1} \leftarrow \sigma\left(\text{Aggregate}\left[W^l \cdot h_k^l, \forall k \in (i \cup N_i)\right]\right)$$

where N_i is the set of neighboring nodes around node i, W^l is the transformation matrix in layer l, and σ is a non-linear activation function. Instead of W^l, the relation-aware GNN learns a specific transformation matrix W_r^l for each relation r. GNN variants serve the purpose of `Aggregate` by different operations such as normalized mean pooling [45] and weighted summation [83]. In this paper, we propose *masked aggregation* and *partitioned attention* in `Aggregate` operation to infuse security domain knowledge into the learning of entity embeddings.

3 VULNERABILITY KNOWLEDGE BASE

In our study, we *first time* annotated and released three vulnerability knowledge graphs (KGs) based on two governmental vulnerability repositories, i.e, National Vulnerability Database (NVD) [71] and ICS-CERT Advisories (ICS-CERT) [16], and one security information portal, i.e., SecurityFocus (SF) [14]. Given those vulnerability knowledge graphs, we also generated and released two cybersecurity-domain entity alignment (EA) datasets by linking entities from ICS-CERT and SecurityFocus to NVD. The annotated dataset is available at [40]. Note that the crawled data are for informational purposes only, following all repositories' terms of service.

Below we explain the annotation process of the vulnerability KGs (§3.1, §3.2) and the EA dataset (§3.3). A quantitative study in §3.4 showing the particularity of the data demonstrates the challenges of aligning security entities.

3.1 KG schema

We design the vulnerability KG schema by summarizing the common artifacts and their relations provided by vulnerability repositories. Table 1 illustrates the common artifacts provided by the three vulnerability repositories investigated in our study, i.e., NVD, ICS-CERT and SF. In our study, we selected the vulnerability repository considering its popularity, vulnerability report format (including both structure and semi-structure). We also include one vulnerability repository which has special focus (i.e., ICS-CERT which focuses on the vulnerabilities of Industrial Control Systems).

Figure 1 illustrates a general schema (i.e., entity types and relations) of the proposed security KGs. These vulnerability artifacts can be interlinked by the concepts in the following standard security databases: Common Vulnerabilities and Exposures (CVE) for

Table 1: Common artifacts provided by security repositories. Profiling artifacts are in bold.

Source \ Artifact	CVE	CWE		CVSS v2&v3			CPE			**Impact**	**Discoverer**
		weakness	cwe-id	vector	score	metric value	**vendor**	**product**	affected version		
ICS-CERT	⊕	⊕	⊕	⊕	⊕	⊕	✓	✓	✓	✓	⊕
SF	✓	⊖				⊕	✓	✓	✓	✓	✓
NVD	✓	✓	✓	⊕	⊕	⊕	✓	✓	✓	✓	

† (✓: all reports, ⊕: large fraction of reports, ⊖: small fraction of reports).

descriptions of publicly known vulnerabilities, Common Weakness Enumeration (CWE) for categorizing software security weaknesses, Common Platform Enumeration (CPE) for encoding names of IT products and platforms, and Common Vulnerability Scoring System (CVSS) for evaluating vulnerability severity.

3.2 KG Construction

For NVD, we construct the vulnerability KG by retrieving vulnerability artifacts from an NVD database [63] and linking them according to our KG schema. For the other two semi-structured vulnerability repositories (i.e., ICS-CERT, SF), we collected all available reports and annotated a subset of them. The annotation process are detailed as below.

Gazeteer. We build a vocabulary for vulnerability-related artifacts based on the Common Platform Enumeration (CPE) [32] and the Common Weakness Enumeration (CWE) [36] to help us recognize mentions of vulnerability artifacts in text. Specifically, we adopted the CWE list as a dictionary of weaknesses, and took advantage of the tree structure to determine whether two vulnerability types are correlated with each other; and the CPE as a dictionary of vendor, name, and versions of IT products. In total, the dictionary contains 26,461 product names, 8,738 vendors, and 187,711 versions from CPE, and 1,029 weaknesses from CWE.

Annotation Process. The entities and relations of vulnerability KGs are annotated by three cybersecurity-major students using the qualitative open-coding technique [78]. Specifically, the annotation standard is summarized through five preliminary rounds of annotation, covering 320 reports. In the first round, annotators discuss the context and the corresponding data entry of each vulnerability artifact in both repositories to summarize a set of patterns to extract entities and assign relations between them. For example, SecurityFocus does not provide a specific data entry for the artifact *weakness*, while through our observation in the preliminary annotation, we could extract the weakness *stack-based buffer overflow* from the text "... is prone to a stack-based buffer overflow vulnerability". In each following round, annotators resolve disagreements and refine the patterns. After the adjustment through five rounds, annotators programmatically parse all reports and manually verify a subset of them. The three annotators amend the annotation of each report in this subset, respectively, as shown in Table 2[1]. These annotations are merged into a final version by majority vote. The statistics of three vulnerability KGs are shown in Table 3. The data is released with the consent from all annotators.

[1]To calculate Fleiss's kappa, we take triples recognized by three annotators as samples, and the types of triples (including *NotRecognized*) as categories.

Table 2: Annotation statistics. The agreement is measured with Fleiss's kappa coefficient [55].

Repository	# Reports	# AnnotatedReports	Agreement
ICS-CERT	1,324	1,000	0.824
SecurityFocus	68,785	7,650	0.873

Table 3: Statistics of security KGs. #Vuls represents the number of vulnerabilities. #Ntypes represents the number of node types.

Security KG	# Nodes	# Vuls	# Edges	# NTypes	# Relations
NVD	40,639	12,000	109,812	11	13
ICS-CERT	4,700	1,136	21,996	12	14
SecurityFocus	22,579	6,000	70,488	7	8

3.3 Construction of Alignment Data

We build two alignment datasets (i.e., CERT-NVD, SF-NVD), which consist of positive and negative alignment pairs. We consider NVD knowledge graph as the target KG and align the other two. This is because NVD is the standard database with the highest coverage of vulnerabilities. Each alignment dataset includes vulnerability artifacts that are *jointly* presented by both two KGs, except for the vulnerability identifiers, i.e., CVE-ID. Records with two and above missing artifacts are dropped in this process. Here we use the same data format with public entity alignment dataset [95].

We label the identical entities (i.e., vulnerabilities) in different sources based on the same reference of common identifiers if they exist (i.e., CVE-ID, CWE-ID, CPE serial number) and security checklist references (i.e., hyperlinks of the same vulnerability to the other repositories) in the NVD database. Table 4 shows the statistics of the KGs to be aligned, including the number of co-occurred vulnerabilities and artifacts across two KGs and the total number of artifacts. In our study, we annotate 987 and 5,769 identical vulnerabilities (i.e., positive alignment pairs) when aligning CERT and NVD, SF and NVD, respectively. To construct negative alignment pairs, we randomly sample 10 vulnerabilities from the other vulnerability repository for each vulnerability in a pair of positive alignments, resulting in 20 negative alignment pairs corresponding to each positive alignment pair. In total, the alignment dataset contains 6,756 positive pairs and 13,5120 negative pairs. In our study, we use this dataset to evaluate CEAM (§6.3).

Table 4: Statistics of KGs for Alignment. #AlignedEnt and #SharedArt stand for the number of co-occurred vulnerabilities and artifacts across two KGs, respectively, #TotalArt stands for the total number of artifacts among two KGs.

Dataset	#AlignedEnt	#SharedArt	#TotalArt
CERT-NVD	987	514	19,721
SF-NVD	5,769	6,668	56,550

3.4 Findings

Based on the three annotated vulnerability KGs and the associated two alignment datasets, we conducted a measurement study to understand the characteristics of those alignment pairs. In our study, surprisingly, we observe the attribute inconsistency between positive alignment pairs and artifact similarity between negative alignment pairs, which is distinct from entity alignment datasets in other domains [95]. We elaborate our findings as below.

Inconsistencies in positive pairs. We denote that a pair of entities encounters an attribute inconsistency if there is no match between all of their artifacts of a certain attribute type, e.g., `Siemens` and `Delta Electronics` are considered as inconsistent in the attribute type of `Vendor`. Here, in our study, a match is defined as either identical tokens or a substring match. Among all the positive pairs in our dataset, 56.0% of vulnerability pairs have artifact inconsistencies. This is a much higher rate compared to other entity alignment datasets such as music and movie datasets [95], where only 8.3% of positive entity pairs face the same problem. These inconsistencies make it challenging for the classifier to accurately identify positive matches. To address this issue, we propose a masked gate mechanism that selectively aggregates attribute information from the neighborhood (see §5.2).

Similarity in negative pairs. We observe that different vulnerabilities can be associated with quite many identical artifacts. More specifically, among all negative pairs in our alignment datasets, 4.04% differ in only one quarter types of artifacts. As a comparison, with the same negative sampling ratio, only 1.14% negative pairs in the music dataset in [95] have the same problem. Such circumstances make it difficult for the classifier to distinguish different-but-similar entities. Based on the vulnerability profiling standard and our observation on the datasets, to distinguish a vulnerability from a set of known vulnerabilities, at least one of its *profiling* artifacts should be different. Therefore, higher attention on the profiling artifacts helps preserve the distinction between similar entities. Accordingly, in our study, we propose a partitioned attention mechanism to assign higher importance weights to the profiling artifacts in a mandatory way.

4 PROBLEM FORMULATION

We represent the security knowledge graph as a heterogeneous graph G connecting nodes in types $\mathcal{A}$ by relations $\mathcal{R}$. We use a set of nodes $\mathcal{V}$ to denote security entities with a mapping $\psi : \mathcal{V} \rightarrow \mathcal{A}$ to the entity types. The security KG is modeled as a set of triples $G(\mathcal{V}, \mathcal{A}, \mathcal{R}) = \{t|t : (i, r, j), i, j \in \mathcal{V}, r \in \mathcal{R}\}$. A triple t describes the relation (i.e., edge) between the subject and the object (i.e., nodes). For example, the triple *(CVE-2021-40142, reported_by, Siemens)* represents the fact that `Siemens` is the discoverer of the vulnerability identified by `CVE-2021-40142`. Entity types and relations are aligned in advance: for a certain G^*, its entity types $\mathcal{A}^*$ and relations $\mathcal{R}^*$ are aligned to the KG schema in §3.1 during KG construction.

Security entity alignment. Given a source knowledge graph $G(\mathcal{V}, \mathcal{A}, \mathcal{R})$ and a target knowledge graph $G'(\mathcal{V}', \mathcal{A}', \mathcal{R}')$, entity alignment aims at finding the identical target entity $v' \in \mathcal{V}'$ for each source entity $v \in \mathcal{V}$. However, in the security domain, only a few entities on one KG has the same real-world reference on another. Therefore, we define the security entity alignment problem as follows: Given a set of entity pairs $S = \{(v_s, v'_s) \in \mathcal{V}_s \times \mathcal{V}'_s, \mathcal{V}_s \subset \mathcal{V}, \mathcal{V}'_s \subset \mathcal{V}'\}$ between two knowledge graphs G and G', security entity alignment aims to determine whether each entity pair in $\{(v_i, v'_i) \in (\mathcal{V} \times \mathcal{V}') \setminus S\}$ refer to the same real word objects with high precision and recall.

5 CEAM: DESIGN AND METHODOLOGY

5.1 Overview

CEAM applies a joint GNN framework to address the security entity alignment problem. CEAM consists of a 2-layer GNN along with masked, attentional aggregation that embeds structural information into node representation and a classifier that makes the alignment decision according to node distance. As in [95], we model the security entity alignment as a binary classification problem on a given set of positive and negative entity pairs. The entire model is jointly trained by the cross entropy loss: $L = -\frac{1}{N}\sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$, where y_i labels whether the two entities in pair i refer to the same object.

Architecture. Figure 2 shows the architecture of CEAM. The nodes are initialized with their textural representations (i.e., blue grids on Figure 2). The GNNs for graph G and graph G' share the same transformation matrix to calculate the hidden representation of each node (i.e., green grids on Figure 2). Then CEAM calculates node-level attention for each relation, according to the interactions between the hidden representation of the centering node (on the top of Figure 2) and its neighboring nodes (on the bottom of Figure 2). Each relation is then represented by the weighted sum of the hidden representation of all neighboring nodes connected with the centering node by this relation (i.e., yellow grids on Figure 2).

To aggregate selective artifact information to learn the semantic embeddings for vulnerability entities and also using the structural information of the vulnerability KG, a mask gate is calculated for each relation representation, based on the cross-graph similarity between centering nodes with joint neighbors (§5.2), and a relation-level attention, namely partitioned attention, is calculated based on both relation type and the importance of the neighboring node connected by this relation. The neighborhood representation of the centering node is the weighted sum with the partitioned attention after an element-wise production between the mask gate and the relation representation. The centering node representation is finally updated as the concatenation of its hidden representation and neighborhood representation (§5.3).

To determine whether a pair of nodes (i, i') refer to the same entity, CEAM considers i and i' as the centering node to update their

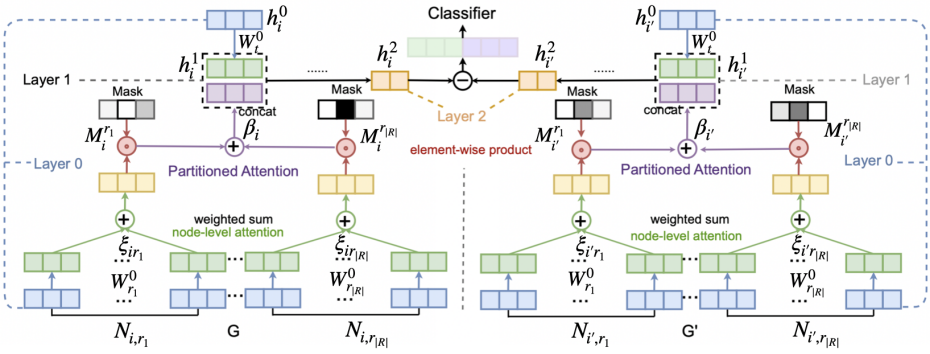

Figure 2: Architecture of CEAM.

representations and send the difference between their representations to the binary classifier to make the alignment decision. To add new entries to a KG, for those with CVE-ID, CEAM uses CVE-ID to identify vulnerabilities in other repositories for positive pairs and generates negative pairs for misaligned entries. It then calculates the loss and gradients on these pairs to update the alignment model through backpropagation. For entries without CVE-ID, CEAM automatically tests alignment with other vulnerabilities using the trained model.

Example. We showcase an intuitive example to explain the proposed alignment mechanism. Considering three vulnerabilities A from ICS-CERT Advisory, A' and B' from NVD. To align A and A', CEAM initializes GNNs with the textual representation of the associated vulnerability artifacts, and computes the representation of each relation by aggregating the artifact representation in this type. A shares the same relation representation with A' in *weakness* and *impact*. However, the product relation of A only contains the information of `Entelibus`, while the product relation of A' contains both the information of `Entelibus` and `Entelibus firmware`. This might hinder the classifier to make a positive alignment decision, while CEAM addresses this issue by a mask mechanism (§5.2). Since A' is the most similar entity with A among all entities in NVD, the parts in A's product representation which are different with A''s product representation will be masked. Also, since A is the most similar entity with A' among all entities in ICS-CERT, the parts in A''s product representation which are different A's product representation will be masked too. As a result, both the product representation of A and A' will be masked into new, similar feature vectors in between of A's product representation and A''s product representation, which boosts the classifier to make the positive alignment decision. However, since A shares the same vendor, affected product and CVSS vectors with B', while the CVSS vectors of A and A' are different, the model can mistakenly align A to B'. To circumvent this case, the partitioned attention mechanism (§5.3) enhances the importance weight of profiling artifacts (e.g., *impact*, *weakness*) and reduces the importance weight of non-profiling artifacts (e.g., *CVSS vector*) to simultaneously highlight the difference between (A, B') in impact and weakness and weaken the consistence in CVSS vector between (A, B') as well as the difference in CVSS vector between (A, A').

Application-driven Challenges and Design. In our study, we observe inconsistencies in vulnerability artifacts for identical vulnerabilities across different repositories, and similarities in artifacts for different vulnerabilities (See §3.4). Inconsistencies in positive pairs can lead to false negatives in the alignment process. However, existing entity alignment methods such as [47, 68, 70, 88, 90, 96] do not address this specific challenge. To overcome this issue, we propose relaxing the similarity constraints between positive pairs, while ensuring that no false positives are introduced. To achieve this, we introduce a masked aggregation mechanism to compute similar representations for the same vulnerabilities with inconsistent artifacts in different repositories. Specifically, it scales down a partial representation of the inconsistent relation (e.g., has_product) for the vulnerabilities, respectively, by multiplying each element in the representation by a scaling number that ranges from 0 to 1. For an entity i on the source KG, if its most similar entity i' on the target KG has inconsistent representation with i, a set of scaling numbers will be applied to scale down the inconsistent elements in the representation of i. If i is also the most similar entity to i' from the source KG, then the corresponding elements in the representation of i' will also be scaled down. Thus, the distance between the representations of i and i' will be reduced even with the inconsistency. However, if i is not the most similar entity to i' from the source KG, then different elements in the representation of i' will be scaled down and the distance between i and i' will not be reduced. This asymmetric mechanism guarantees that masked aggregation will not result in additional false positives.

Similarities in negative pairs can cause false positives because different vulnerabilities can be linked by the same artifacts. However, as we explained in §3.4, the CNA standard requires at least one profiling artifact to be different in order to identify a new vulnerability, while all non-profiling artifacts can remain the same. Therefore, reducing the influence of non-profiling artifacts in alignment can alleviate this issue, but this cannot be guaranteed with neural networks. Accordingly, we proposed *partitioned attention* to reduce the weights of non-profiling artifacts by enforcing that the sum of weights on profiling relations is fixed to a specific value.

5.2 Masked Attribute Aggregation

We use masked attribute aggregation to represent ID-like entities from neighborhood literal artifacts. Due to the heterogeneity of

knowledge graphs, we first aggregate the representation of artifacts in the same type into relation representation. Further, we use the similarity between the relations representation of different entities to calculate the mask, an attribute-level scaling operator for aggregating relation representation into entity representation. The mask relaxes the similarity constraint of a candidate pair of entities if they are mutually the most closed ones over the two KGs. Below we elaborate on the two-stage masked aggregation.

Stage 1: towards relation representation. For node i on graph $G(\mathcal{V}, \mathcal{A}, \mathcal{R})$, its corresponding relation r is represented as

$$\phi_i(r) = \sum_{j \in N_{i,r}} \alpha_{irj} W_r h_j,$$

where W_r is the transform weight for relation r, $N_{i,r} = \{j | (i, r, j) \in G\}$ represents the neighbors associated with i by relation r, α_{irj} characterizing the importance of node j to node i is the relative co-occurrence rate of two nodes across two KGs:

$$\alpha_{irj} = \frac{\exp\left(-d_{j,r}(d_{j,r} + d_{j',r})^{-1}\right)}{\sum\limits_{k \in N_{i,r}} \exp\left(-d_{k,r}(d_{k,r} + d_{k',r})^{-1}\right)},$$

where $d_{j,r}$ and $d_{j',r}$ are the number of entities connected to node j by relation r on graph G and G', respectively.

Such a design aims to emphasize the artifacts that are more representative and can boost positive alignment, while the artifacts connected to more entities on the KG are less specific and representative of a certain entity. For example, consider CVE-2013-1676 in NVD, where both Firefox and Firefox ESR (Extended Support Release) are listed as the affected product. Firefox, associated with 146 CVEs, may not be the most representative artifact to profile CVE-2013-1676, compared to Firefox ESR, which is only associated with 2 CVEs. Moreover, to ensure that positive alignments are not hindered by artifacts that are only available from one repository, it is crucial to consider emphasizing artifacts from both repositories, given that different vulnerability repositories may have varying focuses on vulnerability artifacts. For example, CVE-2016-4529 is listed with different affected products in NVD and ICS-CERT. NVD lists SoMachine HVAC, M171, and M172, while ICS-CERT only lists SoMachine HVAC. Additionally, M171 and M172 are not mentioned in any other repositories in our study. This discrepancy can make it challenging to align vulnerabilities in NVD with their corresponding entities in other repositories. To address this issue, it is essential to emphasize the artifacts that appear in both knowledge graphs, such as SoMachine HVAC, during the aggregation process.

Stage 2: towards entity representation. By stacking the relation representation $\phi_i(r)$, entity i is temporarily encoded by the matrix: $\Phi_i = \left(\phi_i(r_1)^T, \phi_i(r_2)^T, ..., \phi_i(r_{|\mathcal{R}|})^T\right)$. Further, we study a mask gate that scales the attributes (i.e., features) of the relation representation during the aggregation. To do this, we consider a scaling parameter m for each attribute t in the representation of r, which reflects the confidence that entity i possesses the attribute. We construct a set of candidates $C_i = \{i' | \exists j \in (\mathcal{V} \cap \mathcal{V}') : (i, r, j) \in \mathcal{G} \wedge (i', r, j) \in \mathcal{G}'\}$ for entity i. The scaling parameter $m^t_{i,r}$ is measured by the consistency of the attribute t in the representation of r among all candidate pairs $\{(i, i')\}$ of i,

$$m^t_{i,r} = \exp(-\sum_{i' \in C_i} c_{ii'} [\phi^t_i(r) - \phi^t_{i'}(r)]^2),$$

where $\phi^t_i(r)$ is the value of attribute, $c_{ii'} = \text{softmax}_{i' \in C_i}(-||\Phi_i - \Phi_{i'}||_F)$ indicates the correspondence of i' to i. Therefore, for the complete relation representation $\phi_i(r)$, we can define the mask gate altogether as a *diagonal* matrix M^r_i,

$$M^r_i = \exp\left(-\sum_{i'} c_{ii'} \cdot \mathbf{d}_r(i, i')\mathbf{d}_r(i, i')^T\right),$$

where $\mathbf{d}_r(i, i') = \phi_i(r) - \phi_{i'}(r)$. Finally we have the aggregated representation of entity i as:

$$h_i = \frac{1}{|\mathcal{R}|} \sum_r M^r_i \phi_i(r).$$

The process of mask generation is illustrated in Figure 4 in Appendix 13.1 . The mask reduces the effect of different attributes between a candidate pair of entities only if they are *mutually* the most closed ones to each other among two KGs; otherwise, the distance between the masked entity representations will not decrease since the masks are asymmetric and will scale different attributes of the two entities. This enables CEAM to achieve high recall while preserving tolerable precision for security entity alignment.

5.3 GNN with Partitioned Attention

In GNN encoder, we learn transformation matrices W_t for each entity type and W_r for each relation. As in [95], we calculate the node-level attention s and the relation-level attention β to accumulate neighborhood information:

$$z_i = \sum_r \sum_{j \in N_{i,r}} \xi_{irj} \cdot \beta_{ir} \cdot W_r h_j,$$

and combines the transformed self information $h^{(l-1)}_i$ and accumulated neighborhood information z_i to update entity representation:

$$h^{(l)}_i = \sigma\left([W^{(l)}_t \cdot h^{(l-1)}_i || z_i]\right).$$

The node-level attention ξ_{irj} is defined as the normalized cosine similarity between the transformed hidden representations of the two connected nodes:

$$\xi_{irj} = \underset{j \in N_{i,r}}{\text{softmax}}(s_{irj})$$

$$s_{irj} = \text{cos\_sim} < W^{(l)}_t h^{(l-1)}_i, W^{(l)}_r h^{(l-1)}_j >,$$

and the relation-level attention β_{ir} is collectively decided by: 1) whether the relation is predefined as critical, and 2) the similarity between nodes. We consider the profiling relations R_P as the set of relations connecting profiling artifacts (See §2.1) with entities. Let the profiling ratio $\rho = \frac{|R_p|}{|R|}$ be the proportion of the profiling relations take among all relations. As an example, for entities with 3 profiling relations and 2 non-profiling relations, $\rho = \frac{3}{2+3} = 0.6$. Accordingly, we define a partisan term δ with a positive hyperparameter ϵ where $0 < \epsilon < 1 - \rho$:

$$\delta_{R_X} = \begin{cases} \rho + \epsilon - 0.5 & if \quad R_X = R_P \\ -\rho - \epsilon + 0.5 & if \quad R_X = R \setminus R_P \end{cases},$$

Table 5: Results of Cybersecurity Entity Alignment.

Method	CERT-NVD			SF-NVD		
	Pre@Rec=.95	F1	PRAUC	Pre@Rec=.95	F1	PRAUC
GCN	0.198 ± .002	0.330 ± .004	0.265 ± .006	0.181 ± .005	0.315 ± .003	0.257 ± .003
GAT	0.201 ± .005	0.340 ± .008	0.357 ± .003	0.195 ± .004	0.328 ± .002	0.321 ± .002
GraphSage	0.179 ± .001	0.327 ± .003	0.303 ± .004	0.163 ± .004	0.284 ± .002	0.275 ± .001
R-GCN	0.361± .003	0.475 ± .003	0.421 ± .005	0.341 ± .002	0.461 ± .003	0.408 ± .002
R-GAT	0.392 ± .004	0.610 ± .002	0.654 ± .003	0.367 ± .003	0.589 ± .005	0.601 ± .004
R-GraphSage	0.290 ± .002	0.450 ± .005	0.413 ± .009	0.305 ± .003	0.436 ± .012	0.401 ± .010
PARIS	-.-	0.712 ± .000	0.543 ± .000	-.-	0.640 ± .000	0.458 ± .000
PRASE(BootEA)	-.-	0.731 ± .000	0.576 ± .000	-.-	0.685 ± .000	0.524 ± .000
PRASE(MultiKE)	-.-	0.706 ± .000	0.512 ± .000	-.-	0.639 ± .000	0.472 ± .000
CGMuAlign	0.470 ± .026	0.761 ± .012	0.720 ± .003	0.457 ± .012	0.712 ± .015	0.683 ± .006
CEAM	0.778 ± .015	0.843 ± .015	0.875 ± .002	0.725 ± .006	0.801 ± .013	0.811 ± .003

where R_X represents the profiling or non-profiling relations. And the attention weight for $r \in R_X$ is

$$\beta_{ir} = (\frac{1}{2} + \delta_{R_X}) \operatorname*{softmax}_{r \in R_X}\left(|N_{i,r}|^{-1} \sum_{j \in N_{i,r}} s_{irj}\right),$$

Such a mechanism ensures that critical relations (e.g., affected products, etc.) suggested by vulnerability profiling standards are emphasized while reserving the relative importance within a relation group (i.e., profiling or non-profiling). Compared with a brute-force usage of profiling artifacts only, the partitioned mechanism enables full exploitation of artifact information by a soft-version emphasis on the profiling artifacts, where the degree of the emphasis is intensified when the value of ϵ increases. Experiments (see § 6.5) show that the partitioned attention mechanism outperforms traditional attention mechanisms. Also, the non-profiling artifacts, though optionally required by CNAs when CVE-ID requested for newly discovered vulnerability, still provide informative knowledge to the alignment, and the model performance decreases either when the value of ϵ is too high or too low.

6 EVALUATION AND EXPERIMENTS

This section reports our evaluation study on CEAM to understand its effectiveness and performance.

6.1 Experimental Settings

ao ao ao ao a**Implementation**. We implement CEAM and all GNN-based variants using Deep Graph Library (DGL) [84] backended on Pytorch [72] and Python 3.7 [82]. As for the other baseline models, we adopt the official implementation. To learn the textual representation of vulnerability artifacts, we train a Word2Vec [49] model using Gensim [76] with all available reports from ICS-CERT and SecurityFocus. With the learned word embeddings, we use fastText [62] to encode the textual features of artifact as 100-dimensional vectors, which are used as the input to all of the GNN-based methods.

We run CEAM on Intel Core i5-3550 3.30GHz 4-Core Processor with NVIDIA GTX 1070 Ti GPU.

Dataset. In our study, we use two alignment datasets (i.e., CERT-NVD, SF-NVD) mentioned in §3.3 for model evaluation. For the alignment targets, i.e., vulnerability entities, we replace the original CVE-IDs with randomly assigned numerical IDs so entities in a positive pair are represented by different tokens. For other artifact entities, we use the textual information as their initial representations and link them with the vulnerability entities. For example, the *weakness* entity, originally identified by CWE-ID (e.g., CWE-121) is directly referred as its weakness name (e.g., Heap-based Buffer Overflow). To normalize entity expression, for affected versions we used a dictionary consisting of 11 patterns to convert expressions like `version 3.1 and prior` into ≤ 3.1, and further into a list of discrete values $\{1.0, 2.0, 3.0, 3.1\}$ by utilizing all available versions for a product in CPE entries; and for metric values, we utilized a list of 23 keywords to map the relevant phrases (e.g., Exploitable remotely) into the corresponding CVSS metric values (e.g., AV:N).

Evaluation metrics. We apply three metrics: Precision@Recall=0.95, F1, and PRAUC (precision-recall area under curve). For the first metric, we tune the alignment threshold during testing and report the highest precision when the recall reaches the given value. The recall of 0.95 is a desirable value in the security applications [91] to recover the identities of vulnerabilities missing identifiers, where the false positives can be excluded through handful manual verification. For F1 evaluation we use the decision threshold that achieves the highest macro F1 during validation.

Hyperparameters. We set the train-test-split rate as 0.25, and use a 5-fold cross validation on the training set to select hyperparameters, e.g., the learning rate. For each model, we apply a binary search from 0.001 to 0.1 on the validation set for learning rate selection. The best learning rate for CERT-NVD dataset and SF-NVD dataset are 0.02 and 0.025, respectively. The dimension of GNN embeddings in CEAM and baselines is 64, which achieves the best effectiveness among 32, 64, 128, and 256. To select the best value of ϵ in § 5.2, we use grid search to tune the value of $\frac{\epsilon}{(1-\rho)}$ from 0 to 1 with a step size as 0.1. Here, ρ is the portion of profiling relations among all relations, and its value for the two alignment datasets (i.e., CERT-NVD and SF-NVD) are 0.4 and 0.8, respectively. The best value of $\frac{\epsilon}{(1-\rho)}$ for the two datases are both 0.3.

6.2 Baselines

EA w./o. one-to-one constraint. We compare the proposed method with the following state-of-the art methods: *CG-MuAlign* [95] features a collective GNN framework to align entities in two domains:

music and movie. It also formalizes the alignment problem as a classification task on a given set of entity pairs. *PARIS* [79] is a conventional method that aligns entities by probabilistic reasoning. Several studies [67, 93] have noted that PARIS outperforms many "advanced" EA models. *PRASE* [75] is an extension of PARIS, which incorporates the translation-based EA models to boost the alignment decisions.

GNN variants. We evaluate the recent GNN models in security entity alignment, as shown in Table 5. GCN-Align [85] models structure and attribute embeddings by graph convolutional network. GAT [83] aggregates neighborhood information with multi-head attention weights generated by a single-layer feedforward neural network. GraphSage [56] concatenates the aggregated neighborhood representation and the self node representation for the transformation operation. R-GCN, R-GAT, and R-GraphSage are the variants of GCN, GAT, and GraphSage which train specific transformation weights for each relation. We reimplement all of the GNN variants with DGL [84]. CEAM and all the GNN variants use the same model structures: 2-layer GNNs following with two fully-connected layers.

We use the EA datasets in §3.3 to evaluate CEAM, CGMuAlign, and all GNN variants. For PARIS and PRASE, we generate datasets with the format in [75]. To ensure the comparability, for the results of PARIS and PRASE, we only consider the wrong alignments sampled in our negative pairs (See §3.3) as false positives.

6.3 Effectiveness

Table 5 shows the averaged alignment results of three runs on the two datasets. Overall, CEAM outperforms all other methods. CG-MuAlign, which is similar in problem formulation and model architecture with CEAM, shows the second best result. ()PARIS is less successful to identify the same entities in security KGs. Compared with PARIS, PRASE improves 1.9% on averaged F1 by incorporating translation-based alignment modules, such as BootEA [81] and MultiKE [59], which align entities based on their knowledge graph embeddings [65]. Note that CEAM has significantly better performance on Pre@Rec=0.95. This is mainly because the proposed mechanisms enable CEAM to achieve high recall even with a rather large decision threshold. In contrast, previous EA models attach less importance to preventing false negatives and the decision threshold needs be small to achieve high recall, which greatly reduces the precision. More specifically, CEAM employs masked aggregation to relax the similarity constraints for positive aligns, which decreases false negatives to achieve high recall. In the meantime, such relaxation is only applied when the two entities are mutually the most closed candidate to each other, which preserves evidence for negative aligns to prevent low precision. We observe all models show higher performance on CERT-NVD dataset than SF-NVD dataset. The main reason is that SF knowledge graphs provide fewer entity types and suffer more from sparsity issues.

6.4 Analysis

Computation Cost The complexity of CEAM is $O(|\mathcal{V}|\cdot N\cdot(C+\frac{S}{B}))$, where $|\mathcal{V}|$ is the size of the nodes, N is the maximum neighborhood size, C is the maximum candidate size, S is the training size and B is the batch size. The model has 340,309 parameters. Averagely, each batch of size 128 costs 295.3 ms for training.

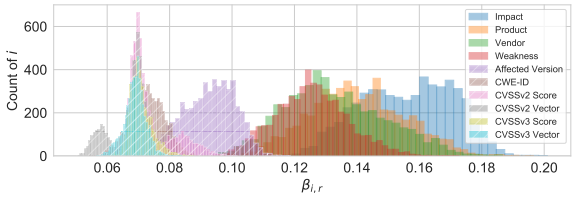

Figure 3: Attention weights of different relations.

Table 6: Analysis of the proposed method.

Method	P@R=0.95	F1	PRAUC
w.o. masked aggregation	0.534	0.721	0.718
w.o. mask (=mean aggregate)	0.550	0.772	0.784
w.o. partitioned attention	0.691	0.794	0.815
w.o. non-profiling artifacts	0.675	0.781	0.784
CEAM	0.752	0.822	0.843

Hyperparameter Sensitivity. Partitioned attention enforces that the sum of weights on profiling relations is fixed to a specific value, $\epsilon + \rho$, where $\epsilon > 0$ is a hyper-parameter empirically selected by cross-validation, and ρ is the ratio between the number of profiling relations (i.e., $|R_P|$) and the number of all relations (i.e., $|R|$). As an example, for entities with 3 profiling relations and 2 non-profiling relations, partitioned attention enforces that the sum of weights on profiling relations to be $0.6 + \epsilon$ where ϵ is the hyperparameter, and $\rho = |R_P|/|R| = 0.6$. The weights within the profiling relations are learned through the neural network. We evaluate the hyperparameter sensitivity of the proposed model by varying the value of ϵ and the results are presented in Table 7 in Appendix 13.2.

Justification of Artifacts. We assessed the importance of various artifact types by examining the attention weights of each relation computed by the trained alignment model. The computation of the attention weights depends on the hidden representations of entities and relations, which means that different vulnerabilities can yield distinct weights for each relation. To visualize this distribution, we plotted the attention weights associated with each relation for all vulnerabilities. For this analysis, we use the CERT-NVD dataset, which contains more relations than the SF-NVD dataset. Specifically, we display the distribution of $\beta_{i,r}$ for each vulnerability i, grouped by relation (i.e., type of artifact) r in Figure 3. Profiling relations are depicted by histograms filled with solid colors, while non-profiling relations are represented by those filled with slashes. As shown in Figure 3, profiling relations generally have higher weights than non-profiling relations, indicating their higher importance in aligning vulnerabilities. Among the profiling relations, `Impact` has the highest weights, while `Affected Version` has the highest weights among the non-profiling relations.

6.5 Ablation Study

We perform an ablation study to gain a better insight into how the two proposed mechanisms: asymmetric masked aggregation and partitioned attention affect the effectiveness of CEAM. Table 6 shows the improvement by each proposed mechanism, respectively.

All the scores are averaged on the two alignment datasets. The variant without both two mechanisms is equivalent to the baseline model using R-GAT to train node representations. Overall, CEAM outperforms baselines by 22.55% F1 score. Below we detail the quantative improvement by the two proposed mechanisms.

Masked aggregation. The first two rows of Table 6 show the effectiveness of CEAM without the masking mechanism. More specifically, the first row shows the model's effectiveness without any information aggregation. This is to initialize the input to GNNs by the textual embeddings of all vulnerability entities and artifact entities, where the vulnerability entities are originally represented by a sequence of randomly generated IDs and the artifact entities are originally represented by the phrases extracted from the vulnerability reports. The second row shows the result where the vulnerability entity representations are initialized as the mean of its neighboring artifacts. In this case, the importance of each artifact to the vulnerability entity is equally assigned for the aggregation of neighborhood information, while the mask mechanism is disabled. Comparing the first two rows, we observe that the information aggregation boosts the performance by 6.6% in averaged F1 and 1.6% in averaged Precision@Recall=0.95. This is because the aggregated representation of the neighborhood is more informative than the textual representation of a randomly generated ID, as the input to GNNs. Further, comparing the second and the fifth row, the mask mechanism improves 5.9% in averaged F1 and 20.2% in averaged Precision@Recall=0.95. The underlying reason is that the masking mechanism effectively alleviates the false negatives introduced by inconsistencies in artifacts of identical vulnerabilities.

Partitioned attention. In the third and fourth row of Table 6, we present the alignment result using traditional attention mechanism, where the attention weights between node i and node j under relation r is computed by $\alpha_{irj} = \underset{r \in R, j \in N_{i,r}}{\text{softmax}}\ \sigma\left(\overrightarrow{w_r}^T \left[h_i || W_r h_j\right]\right)$, and the aggregated neighborhood information is represented as $z_i = \sum_{r \in R} \sum_{j \in N_{i,r}} \alpha_{irj} W_r h_j$. Compared with traditional attention operation on all entity types, the partition mechanism enhances 2.8% averaged F1, since the profiling artifacts are enforced to be emphasized to alleviate the noises introduced by non-profiling artifacts. We also compare the result on dataset that only employs profiling artifacts while dropping the non-profiling ones. The result shows that the non-profiling artifacts still provide useful information to improve 1.3% F1. Upon this comparison, the best solution to deal with the non-profiling artifacts is to limit their importance with the proposed partition mechanism, rather than directly dumping them.

7 FINDINGS

Among the 3,546 annotated vulnerabilities in NVD, SF and ICS-CERT (§3), 52 are released in reports without the identifier. Note that those vulnerability entities do not appear in the EA datasets for model training and evaluation since their corresponding NVD vulnerabilities are unkown. To find the aligned vulnerabilities for those without CVE-ID in other vulnerability repositories, we construct the candidate pairs for the 52 vulnerabilities as described in §5.2 and run CEAM to launch the vulnerability alignment task. Given the results of CEAM, we manually verify the *positive* alignments according to the reference checklist and the artifacts provided from both sides.

CEAM makes 39 positive alignments for 32 vulnerabilities, while the other 20 vulnerabilities are not aligned with any vulnerability in our knowledge base. We manually checked the 20 vulnerabilities without positive alignment results and cannot find matching records in our KB, either. For example, ICS Advisory (ICSA-10-313-01) reports a buffer overflow vulnerability in RealFlex RealWin versions prior to 2.1.8 (Build 6.1.8), which may result in remote code executions. We search in our vulnerability knowledge base and cannot find a vulnerability with the same (or similar) vendor, affected product, vulnerable functions, weakness, and impact.

Among all 39 positive alignments, 28 are correct and 11 are false positives. Below we perform case studies of those positive alignments to discuss the security implications delivered by CEAM, and also elaborate on the challenges in vulnerability information alignment across different repositories.

Case Studies. We present a sample of the positive alignment results made by CEAM and clarify the manual verification process as well as the evidence we use to justify the correctness of the alignment decision.

ICS Advisory (ICSA-10-362-01) [8] reports a *directory traversal* vulnerability that affects all *IntegraXor* versions *prior to Version 3.6 (Build 4000.0) without* CVE-ID. The report describes this vulnerability as follows:

> *"IntegraXor is vulnerable to a directory traversal exploit. An attacker may add an arbitrary path and file and read any arbitrary file. This vulnerability is exploitable from a remote machine. A low level of skill is needed to exploit this vulnerability."*

This indicates that the vulnerability can be exploited remotely by appending an arbitrary path via an HTTP URL parameter to read any arbitrary file, which could lead to data leakage. Besides, to launch this attack, threat actors can just utilize existing scanners (e.g., RIPS Scanner [9]) to scan through a directory tree. Hence, a `low` level of skill is needed to exploit this vulnerability (i.e., `AC:L`). The exploiting characteristics correspond to the CVSS exploitability metrics metric values of `AV:N`, which means that this vulnerability can be exploited remotely.

CEAM aligns this vulnerability to *CVE-2010-4598*. To check the correctness of this alignment, we manually check the vulnerability artifacts associated with CVE-2010-4598 in other repositoreis, i.e. NVD and SecurityFocus. In NVD, this vulnerability is presented with the description as:

> *"Directory traversal vulnerability in Ecava IntegraXor 3.6.4000.0 and earlier allows remote attackers to read arbitrary files via a .. (dot dot) in the **file_name parameter** in an open request."*

SecurityFocus records CVE-2010-4598 as Ecava IntegraXor 'file_name' Parameter Directory Traversal Vulnerability with Bugtraq ID 45535, affected products as Ecava IntegraXor 3.6.4000.0, Ecava IntegraXor 3.5.3900.5, Ecava IntegraXor 3.5.3900.10, and Ecava IntegraXor 3.5.

From the above information, we conclude that the reported vulnerability in ICS Advisory (ICSA-10-362-01) shares the same weakness (i.e., directory traversal), affected product (i.e., IntegraXor), versions (i.e., versions prior to 3.6.4000.0), and parameter *file_name* with CVE-2010-4598, which help us to confirm the correctness of the alignment result. Also, we observe additional evidence in

CVSS metric values and reference checklists. More specifically, in NVD, the vulnerability is assigned with a CVSSv2 base score of 5.0 along with a CVSSv2 vector of (AV:N/AC:L/Au:N/C:P/I:N/A:N). SecurityFocus also assigns the metric value (AV:N) to the vulnerability with the description "Remote:Yes Local:No". These are consistent with the metric values assigned by the ICS advisory report. In addition, by checking the PoC [1] of *CVE-2010-4598*, we observed that the PoC is exactly matched to the vulnerability description in the ICSA-10-362-01, which appending an arbitrary path `..\boot.ini` as a parameter to a HTTP URL http://SERVER:7131/PROJECT_NAME/open?file_name=.

Falsely aligned vulnerabilities. The falsely aligned vulnerabilities mainly come from the low quality and information incompleteness of the unstructured reports. In addition to the absence of CVE-IDs, these vulnerabilities are reported without certain artifacts, such as weakness or affected version. As an instance, ICS Advisory (ICSA-10-214-01) [2] reports a vulnerability affecting the Wind River Systems' VxWorks platform, which is a debug service enabled by a weak hashing algorithm used in authentication and may allow an attacker to brute force the password. CEAM aligns this vulnerability to CVE-2010-2968, which identifies the vulnerability that the FTP daemon in Wind River VxWorks does not close the TCP connection after a number of failed login attempts, which makes it easier for remote attackers to obtain access via a brute-force attack. We carefully check the detailed information of the two vulnerabilities and find out this is a false positive. CEAM mistakenly aligns the two vulnerabilities due to the lack of information about *weakness* and *affected version*, as well as the high similarity in vendor, affected product, and impact.

The remaining falsely aligned cases mainly result from the high similarity between candidate entities. For example, CVE-2011-0406 identifies a heap-based buffer overflow vulnerability in HistorySvr.exe in Wellintech Kingview 6.53 which allows remote attackers to execute arbitrary code. CVE-2011-4536 identifies a heap-based buffer overflow in nettransdll.dll in HistorySvr.exe in WellinTech KingView 6.53 and 65.30.2010.18018 which allows remote attackers to execute arbitrary code. ICS Advisory (ICSA-11-017-01) [3] reports a heap overflow vulnerability in KingView V6.53 that would allow a remote attacker to cause the service to crash and arbitrary code execution. CEAM aligns this vulnerability to both CVE-2011-0406 and CVE-2011-4536. By checking the information of affected version, vulnerability function and NVD reference checklists, we find out CVE-2011-0406 is the right alignment to the vulnerability in ICS Advisory (ICSA-11-017-01) while CVE-2011-4536 is a wrong alignment. The main reason is that CVE-2011-4536 and CVE-2011-0406 have high similarities, especially in profiling artifacts. In our study, we revealed that the quality and the information completeness of the reports are crucial for the security entity alignment. Also, artifacts not in the mandatory requirement for requesting CVE-ID, such as affected versions and vulnerable functions, are also important to identify the vulnerability entity.

8 APPLICATIONS

In this section, we discuss a few potential applications, which can be built on CEAM, for the integration and inspection of vulnerability intelligence. Particularly, using CEAM, we can seek to a more comprehensive and credible vulnerability database. Below we elaborate on the design to (1) supplement vulnerability artifacts across repositories and (2) debunk erroneous vulnerability artifacts.

Given a new vulnerability repository $\mathcal{R}$ with structured or semi-structured reports, we first programmatically extract the vulnerability artifacts from the reports and construct the knowledge graph for $\mathcal{R}$, where each vulnerability artifact is represented by its textual representation. Then we apply CEAM to update the representation of the vulnerability entities and their artifacts, and align the new vulnerability entities in $\mathcal{R}$ to the ones on our constructed vulnerability knowledge base (§ 3). To constrain the size of the candidates, we only construct alignment pairs between vulnerabilities with the same artifacts.

Among all types of vulnerability artifacts, some of them has a multi-to-one relation to the vulnerabilities (e.g., affected devices and versions), while others has a one-to-one relation to the vulnerabilities (e.g., vulnerable function). For example, the Linux kernel evolves quickly and has a large number of versions and derivatives such as Android, Ubuntu, Red Hat, and IoT systems. Therefore, the records for Linux kernel-related vulnerabilities are supposed to cover the information of these various versions and derivatives [25]. In addition, given a vulnerability and its associated affected device and version, vendors and CVE maintainers adopt a *single* CVSS score for each CVSS vector [34]. Hence, the records for CVSS-related vulnerability artifacts are supposed to be consistent among different repositories. We call the artifacts with one-to-one relation as unique artifacts and the ones with multi-to-one relation as non-unique artifacts.

By aligning the vulnerabilities from different repositories, we propose two applications: 1) for non-unique artifacts, we check whether a repository suffers from information incompleteness and supplement the missing information by artifacts associated with the same vulnerability from other repositories; 2) for unique artifacts, we check whether the artifact recorded in different repositories are consistent and reveal the inconsistent circumstances. Below we elaborate on the two applications on the reports associated with Industrial Control System (ICS) vulnerabilities. In addition to the reports in our vulnerability knowledge base, we involve SecurityTracker (SF) [14] and CVERecords [50] as new repositories. SecurityTracker (ST) [13] is a vulnerability report gathering platform, which monitors the latest public vulnerability reports on the Internet and collects around 40K vulnerability reports. In this application, we fetch 671 ICS device vulnerability reports from this platform based on the vendor names and affected products we identified from ICS-CERT Advisories. CVERecords is an enumeration list of security vulnerabilities with textual descriptions. By applying CEAM, we align the artifacts associated with 2,691 ICS vulnerabilities from 1,368 ICS-CERT Advisories reports, 1,190 SF reports, 2,641 NVD reports, 671 ST reports, and 2,656 CVE reports. Note that here we added vulnerability repositories with a relatively small size (e.g., ST) or well-structure (e.g., CVE), which are easy for manually result validation, to demonstrate our applications.

Vulnerability artifact supplementary. We observe critical information loss (e.g., missing CVSS score, affected version) in the vulnerability repositories, even for the high-profile ones including ICS-CERT Advisories and NVD. This has a negative impact on the

security professionals to assess vulnerability timely and effectively, given the wide adoption of the vulnerability repositories by security vendors [74]. The alignment between vulnerabilities enables information integration among different repositories, making it possible to supplement missing vulnerability artifact for each independent repository. In general, ICS-CERT Advisories deliver the most comprehensive information for ICS vulnerabilities. However, we still observe the absence of critical artifacts. Through the information integration on 1,368 ICS-CERT Advisories reports, 1,190 SF reports, 2,641 NVD reports, 671 ST reports, and 2,656 CVE reports, we supplement *28* `CVE-IDs` for ICS-CERT. In addition, we observe that among the five repositories, SecurityTracker is the only one that provides CVSS temporal metrics, i.e., Remediation Level (RL) and Report Confidence (RC). For example, vulnerabilty CVE-2011-3389 is confirmed by the vendor while the solution to fix the vulnerability is not available. Such description corresponds to CVSS metric values of Remediation Level: Unavailable (RL:U) and Report Confidence: Confirmed (RC:C). With these information, we could further supplement the CVSS temporal score by taking the complementary metric values and leaving the value of the other metric, i.e., Exploit Code Maturity (E) as Not Defined (X). By aligning the vulnerabilities in SecurityTracker with the ones in other repositories, we supplement `CVSS temporal metric values and scores` for *163 vulnerabilities* in ICS-CERT Advisories, *159 vulnerabilities* in SecurityFocus, and *186 vulnerabilities* in NVD and CVERecords with the corresponding textual descriptions.

Credibility analysis of vulnerability artifacts. Existing vulnerability artifacts (e.g., vulnerable function, severity level) in the vulnerability report are usually determined manually by vulnerability assessors. It lacks details for the public to know how such vulnerability artifacts were generated. Even worse, a recent study [43] shows that the procedure of vulnerability assessment is extremely subjective: 50% of security professionals in the survey mis-scored vulnerabilities with the same vulnerability reports. In our study, we observe a vulnerability can be reported and disclosed over multiple channels, and inconsistent information is common. Current studies [44, 51, 53, 69] on data quality issues of the public vulnerability repositories either focus on limited vulnerability artifacts (e.g., vulnerable device version [51, 69]) or limited vulnerability repositories (e.g., only focus on the NVD [44]). Using CEAM, we can systematically detect the inconsistency of various kinds of vulnerability artifacts on a large scale.

More specifically, we measure the inconsistency of those unique artifacts with one-to-one relations to debunk erroneous artifacts. Below we demonstrate this application using the artifacts CVSS vector, CVSS score, and CVSS metric values as an example. Note that in this application, we aim at revealing the inconsistency for manual validation, not for automatic vulnerability assessment.

• *CVSS score mismatches the assigned CVSS vector.* Given a CVSS vector, a unique CVSS score can be calculated using the score formula in CVSS specifications [35]. We find that 18 (17 in CVSSv3 and 1 in CVSSv2) vulnerabilities in ICS-CERT have erroneous CVSS scores, which are inconsistent with those in other repositories. When manually examining these cases, we found that those CVSS scores mismatch the assigned CVSS vectors. However, vulnerabilities in other repositories do not have this issue.

• *CVSS vector varies among different sources.* The inconsistency of CVSS vector is measured in the same version of the metrics (either CVSS v3/3.1 or CVSS v2) and is identified when a vulnerability is assigned different CVSS vectors. Vulnerabilities in ICS-CERT and NVD show inconsistency in 182 CVSSv2 vectors and 616 CVSSv3/3.1 vectors, given they are associated with the same affected devices and versions.

• *Inconsistency in CVSS metric values.* CVSS metric values encounter credibility issues when metric values vary between different repositories. In our study, we recognizes 20 vulnerabilities described as "This Vulnerability Could Be Exploited Remotely" in ICS-CERT (i.e., AV:N) while as "Remote:No, Local:Yes" (i.e., AV:L) in SF. The opposite case occurs in 26 vulnerabilities.

9 DISCUSSION

Lesson learned. According to our research, additional artifacts are found critical to the identification of security vulnerabilities. First, we observe that critical parameters of vulnerable functions, vulnerable functions or vulnerable files are essential to profile security vulnerabilities, which are not required by CNAs when requested CVE-IDs. The critical parameters are variables that can cause potential erroneous execution states without security checks [66]. The vulnerable function refers to the target function of a vulnerability exploit, and the vulnerable file is a program that loads a vulnerable function. For example, in the Linux kernel, the size variable in memcpy() in memcpy.c is a critical parameter that could cause buffer overflow without a sanity check on the input length. Here, memcpy() is a vulnerable function, and memcpy.c is a vulnerable file. Without such information, a newly discovered vulnerability may be considered a duplication of someone in the existing vulnerability database. For example, CVE-2018-20031 and CVE-2018-20032 describe a denial-of-service vulnerability on FlexNet Publisher version 11.16.1.0 and earlier, which allows a remote attacker to send a combination of messages to lmgrd or the vendor daemon, causing the heartbeat between lmgrd and the vendor daemon to stop, and the vendor daemon to shut down. The vulnerability type, vendor, affected product and versions, and impact of these two different vulnerabilities are all identical. However, these two vulnerabilities can be distinguished by the vulnerable function: CVE-2018-20031 is related to `message decoding` in lmgrd and vendor daemon components, while CVE-2018-20032 is related to `preemptive item deletion` in lmgrd and vendor daemon components. Without such information, these two vulnerabilities will be identified as the same one. Therefore, we encourage CNA to involve vulnerability exploitation related artifacts (i.e., parameters, vulnerable function, vulnerable file, etc.) in the required information for applying CVE-IDs.

In addition, we observe multiple inconsistencies in *weakness* of identical vulnerabilities across vulnerability repositories, some of which are not in conflict indeed. For example, for vulnerability CVE-2019-9009, ICS Advisory (ICSA-19-255-05) [12] reports its weakness as Detection of Error Condition without Action (CWE-390), while NVD reports its weakness as Improper Handling of Exceptional Conditions (CWE-755) [11]. The two weaknesses are compatible, where CWE-390 emphasizes that no actions are taken against a faulty condition, while CWE-755 focuses more on the incorrect handles of this exceptional condition. In addition, the official CWE

lists are presented in tree architecture [37–39], indicating the inclusion relation between different weaknesses from three perspectives: hardware, software, and research. For these reasons, it is likely that a vulnerability can be categorized into multiple vulnerability types (i.e., weakness). Hence, for the sake of information consistency, we suggest CNAs request the vulnerability discoverers to enumerate all possible weaknesses when reporting a vulnerability.

Limitations. Our research demonstrates that the proposed domain-driven mechanisms benefit security entity alignment. Most false positives are produced in vulnerabilities with similar impacts and the same other artifacts. The representations of such underlying different vulnerabilities are very closed, and it's hard to distinguish them for all similarity-based alignment methods. Meanwhile, the false negatives are mainly due to data quality issues such as information inconsistency, which is commonly observed in real-world vulnerability repositories [44, 51, 61]. For example, the affected product, vendor, and weakness of CVE-2018-7084 provided by ICS-CERT and NVD are all different. Although the mask mechanism relaxes the similarity constraint, CEAM cannot align the two records in this case with too many profiling artifacts inconsistent. We leave the entity alignment with information quality assessment as future work.

User impact of missing or inconsistent artifacts. As centralized and rich information sources, vulnerability repositories play an essential role in the risk management process. Security analysts rely heavily on these repositories for manual vulnerability assessments and prioritization [23]. Commercial vulnerability management tools [21, 22, 24, 26, 27] also widely utilize these repositories to automate the scanning and analysis of common vulnerabilities. For example, `PAS Cyber Integrity` [26] and `Nozomi networks` [24] refer to both NVD and ICS-CERT to automate the monitoring and evaluation of ICS vulnerabilities. Thus, missing or inconsistent artifacts in vulnerability repositories may hinder the vulnerability assessment process. Experts widely agree that discrepancies in NVD and advisories from vendors make patch prioritization more difficult [28, 29, 31]. Aligning the vulnerabilities from different repositories mitigates the missing artifacts by cross-referencing. For example, as we presented in §8, by aligning SecurityTracker with other repositories, CEAM supplements CVSS temporal metric values and scores for 163 vulnerabilities in ICS-CERT, 159 vulnerabilities in SecurityFocus, and 186 vulnerabilities in NVD and CVERecords. The supplemented artifacts can support the assessment and prioritization of these critical vulnerabilities in the customized devices, which may change over the lifetime of the vulnerabilities. We recognize the need for an in-depth user study to investigate the impact of missing or inconsistent artifacts on the usability of vulnerability repositories for end-users. As a result, we leave it as future work to investigate the impact of the issues we have identified in this study via a human-subject study.

Mitigation of Inconsistencies. In our study, CEAM reported 2,962 inconsistent artifacts among 2,691 vulnerabilities across vulnerability repositories ICS-CERT, NVD, SecurityFocus, and SecurityTracker. The artifact category with the highest number of inconsistent artifacts is CVSSv3 vector (41.59%), followed by Affected Version (23.30%), Weakness (19.72%). Notably, a significant portion of inconsistent artifacts is related to CVSS vector/value (e.g., *AV:N*, *AV:L*), which is assigned based on manual assessment of the vulnerability severity. This assessment process can be influenced by a variety of factors such as the assessor's expertise, configuration environment, and other contextual factors. To address this issue, it is essential to conduct vulnerability assessment automatically, considering the comprehensive factors on the vulnerability and accounting for its unique environment, context, code, and security posture. Automatically assessing a vulnerability to obtain credible vulnerability artifacts is still an open research question, which usually requests in-depth system analysis. Existing solutions include DIFFCVSS [86], an automated tool for analyzing CVSS artifacts of a vulnerability on Linux derivatives and versions. Specifically, given a Linux vulnerability, DIFFCVSS employs both static program analysis and natural language processing (NLP) to precisely identify and map Linux functions to CVSS metrics, and match code paths related to the CVE in both the mainstream version and other derivatives.

Other application domains. The current design of CEAM is for vulnerability alignment across different vulnerability repositories. We believe that it will also work well for the vulnerability deduplication, which determines whether a newly-coming vulnerability report has been disclosed and is under process or not, though further study and evaluation are certainly needed here. CEAM automatically aligns the vulnerability in the report with those existing in the enterprise's vulnerability databases and eliminates any duplication. What is less clear is the technique's effectiveness on more diverse and less-structured vulnerability artifacts in vulnerability reports, like vulnerability-associated functions system calls and critical variables across different systems. We will leave it as our future work. In addition, it will be exciting to see future work that could consider using CEAM for the correlation of two similar vulnerabilities to accelerate the vulnerability severity assessment and remediation. More specifically, with less strict ϵ, which looses the degree of emphasizing the profiling artifacts (see §5.3), CEAM can output similar vulnerabilities with similar impact. The CVSS assessment information of one vulnerability can potentially guide the evaluation of the other.

10 RELATED WORKS

Graph Entity Alignment. Recent graph entity alignment methods assume that same entities on different KGs have similar attribute distributions and neighborhood structures. Translation-based methods [48, 57, 59, 80, 81, 94] use knowledge graph embedding models to generate entity embeddings from triple structures. On the other hand, GNN-based methods [45, 47, 68, 70, 87, 88, 96] apply graph convolution operations to utilize neighborhood information. Additionally, [73] proposes a reinforced training strategy to achieve noise-aware entity alignment, while [90] proposes topology-invariant gates to dynamically align evolving knowledge graphs. By applying graph isomorphism, [67] learns a permutation matrix that transforms one KG to another by reordering the entity node indices.

Domain-specific entity linking. To disambiguate candidate entities for a given mentions, [60] utilizes domain information to filter candidates; [77] designs a probabilistic linking model that combines the distribution of entities and domains. On the other hand, [64] proposes entity linking for low-source domains by allowing human annotators to make corrections on ranked candidate entities for

each mention. However, all existing works have overlooked domain-specific properties while using domain information to constrain the search space of candidates.

Security-domain entity identification. Cross-platform entity identification has been studied in the cybersecurity domain to recognize the same entity in different platforms. For example, [58] and [89] detect the same code function across multiple platforms such as x86, ARM, MIPS or different applications, using a graph embedding generation network. [52] identify the same account/member in different anonymous underground marketplaces by building a weighted multi-view network based on user relatedness and applying GCN for node representations. However, unlike these works, our approach is the first to incorporate security-domain KG properties (§3.4) into the entity alignment process.

11 CONCLUSION

In this paper, we released the first annotated datasets for cybersecurity-domain entity alignment, and unveil their characteristics that challenge the assumption made in traditional entity alignment tasks. Based on this feature, we propose an entity alignment model, CEAM, which equips GNN-based entity alignment model with two application-driven designs, asymmetric masked aggregation and partitioned attention. Experimental results demonstrate the effectiveness of our method, which outperforms the state-of-the-art entity alignment models on the vulnerability alignment task. In addition, two potential applications of CEAM are discussed, including supplementing vulnerability artifacts across repositories and debunking erroneous vulnerability artifacts.

12 ACKNOWLEDGMENT

We would like to express our gratitude to Dr. Raheem Beyah and Dr. Chengkai Li for their insightful discussions and feedback. We also thank the anonymous reviewers for their valuable and constructive feedback. This work is supported in part by the NSF-1850725, 1937143, the Indiana University Institute for Advanced Study (IAS) and the Grant Thornton Institute.

REFERENCES

[1] 2010. CVEexploit. https://www.exploit-db.com/exploits/15802.
[2] 2010. ICSA-10-214-01. https://www.cisa.gov/uscert/ics/advisories/ICSA-10-214-01.
[3] 2011. ICSA-11-017-01. https://www.cisa.gov/uscert/ics/advisories/ICSA-11-017-01.
[4] 2015. Csoonline. https://www.csoonline.com/article/3122460/over-6000-vulnerabilities-went-unassigned-by-mitres-cve-project-in-2015.html.
[5] 2015. VulnDB. https://vulndb.cyberriskanalytics.com.
[6] 2016. InfoSecInstitute. https://resources.infosecinstitute.com/topic/vulnerability-scanners-2/.
[7] 2016. Synopsys. https://www.synopsys.com/blogs/software-security/cve-ids-missing/.
[8] 2018. ICSA-10-362-01. https://www.cisa.gov/uscert/ics/advisories/ICSA-10-362-01.
[9] 2018. RIPS - A static source code analyser for vulnerabilities in PHP scripts. http://rips-scanner.sourceforge.net/.
[10] 2018. Tenable. https://tenable.my.site.com/s/question/0D5f200005QpXEfCAN/cve-id-does-not-get-reflected-for-all-the-vulnerabilities-detected.
[11] 2019. CVE-2019-9009. https://nvd.nist.gov/vuln/detail/CVE-2019-9009.
[12] 2019. ICSA-19-255-05. https://www.cisa.gov/uscert/ics/advisories/icsa-19-255-05.
[13] 2019. SecurityTracker. https://securitytracker.com/.
[14] 2020. SecurityFocus: an online computer security news portal and purveyor of information security services. https://www.securityfocus.com/.
[15] 2021. BugTraq mail list. https://bugtraq.securityfocus.com/archive.
[16] 2021. CISA ICS-CERT Advisories. https://www.us-cert.gov/ics/advisories.
[17] 2021. SecList BugTraq mail list. https://seclists.org/bugtraq.
[18] 2021. SecList Fulldisclosure mail list. https://seclists.org/fulldisclosure.
[19] 2021. SecList OSS-SEC mail list. https://seclists.org/oss-sec.
[20] 2021. Tenable2. https://community.tenable.com/s/question/0D53a00007jcmOxCAI/cve-number-is-missing.
[21] 2022. Castlegarde Security. https://www.castlegarde.com/Virtual-Vulnerability-Assessment-VVA.
[22] 2022. Connectwise. https://info.connectwise.com/cybersecurity/cybersecurity/demo/sem/vulnerability-management?.
[23] 2022. IBM security. https://www.ibm.com/support/pages/how-identify-security-vulnerabilities-within-application-impacts-and-remediation.
[24] 2022. Nozominetworks. https://www.nozominetworks.com/webinars/nozomi-networks-labs-1st-half-2022-ot-iot-security-review/.
[25] 2022. OS-Aware Vulnerability Prioritization via Differential Severity Analysis. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity22/presentation/wu-qiushi
[26] 2022. PAS Cyber Integrity. https://pas.com/products-and-services/ot-ics-cyber-security/pas-cyber-integrity.
[27] 2022. Red Balloon Security. https://redballoonsecurity.com/.
[28] 2022. SecurityWeek. https://www.securityweek.com/many-ics-vulnerability-advisories-contain-errors-report/.
[29] 2022. The Good, Bad and Ugly of CVSS Scores. https://blog.denexus.io/beyond-cvss-scoring.
[30] 2023. Android Bulletin. https://source.android.com/security/bulletin.
[31] 2023. Conflicting vulnerability scores can affect patch prioritization, researchers warn. https://www.cybersecurity-help.cz/blog/3111.html.
[32] 2023. CPE Dictionary - Common Platform Enumeration: IT products and Platforms. https://csrc.nist.gov/Projects/Security-Content-Automation-Protocol/Specifications/cpe/dictionary.
[33] 2023. CVE Numbering Authorities. https://cve.mitre.org/cve/cna.html.
[34] 2023. CVSS (Common Vulnerability Scoring System). https://www.first.org/cvss/.
[35] 2023. CVSS Specification Document (Version 3.1 Release). https://www.first.org/cvss/specification-document.
[36] 2023. CWE (Common Weakness Enumeration: vulnerability categorizations). https://cwe.mitre.org/.
[37] 2023. CWE Architecture by Hardware Design. https://cwe.mitre.org/data/definitions/1194.html.
[38] 2023. CWE Architecture by Research Concept. https://cwe.mitre.org/data/definitions/1000.html.
[39] 2023. CWE Architecture by Software Development. https://cwe.mitre.org/data/definitions/699.html.
[40] 2023. Implementation of CEAM: code, datasets, appendices, and full paper. https://sites.google.com/view/vulnerablity-ailignment/home.
[41] 2023. Palo Alto Networks Security Advisories. https://security.paloaltonetworks.com/.
[42] 2023. Vulnerability Database. https://vuldb.com/.
[43] Luca Allodi, Sebastian Banescu, Henning Femmer, and Kristian Beckers. 2018. Identifying relevant information cues for vulnerability assessment using CVSS. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 119–126.
[44] Afsah Anwar, Ahmed Abusnaina, Songqing Chen, Frank Li, and David Mohaisen. 2021. Cleaning the NVD: Comprehensive quality assessment, improvements, and analyses. *IEEE Transactions on Dependable and Secure Computing* (2021).
[45] Max Berrendorf, Evgeniy Faerman, Valentyn Melnychuk, Volker Tresp, and Thomas Seidl. 2020. Knowledge graph entity alignment with graph convolutional networks: Lessons learned. *Advances in Information Retrieval* 12036 (2020), 3.
[46] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).
[47] Yixin Cao, Zhiyuan Liu, Chengjiang Li, Juanzi Li, and Tat-Seng Chua. 2019. Multi-channel graph neural network for entity alignment. *arXiv preprint arXiv:1908.09898* (2019).
[48] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2016. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. *arXiv preprint arXiv:1611.03954* (2016).
[49] Kenneth Ward Church. 2017. Word2Vec. *Natural Language Engineering* 23, 1 (2017), 155–162.
[50] CVE. 2023. CVE. https://cve.mitre.org/.
[51] Ying Dong, Wenbo Guo, Yueqi Chen, Xinyu Xing, Yuqing Zhang, and Gang Wang. 2019. Towards the detection of inconsistencies in public security vulnerability reports. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 869–885.
[52] Yujie Fan, Yiming Zhang, Shifu Hou, Lingwei Chen, Yanfang Ye, Chuan Shi, Liang Zhao, and Shouhuai Xu. 2019. idev: Enhancing social coding security by cross-platform user identification between github and stack overflow. In *28th International Joint Conference on Artificial Intelligence (IJCAI), 2019*.

[53] Sadegh Farhang, Mehmet Bahadir Kirdan, Aron Laszka, and Jens Grossklags. 2020. An empirical study of Android security bulletins in different vendors. In *Proceedings of The Web Conference 2020*. 3063–3069.

[54] Xuan Feng, Xiaojing Liao, X Wang, Haining Wang, Qiang Li, Kai Yang, Hongsong Zhu, and Limin Sun. 2019. Understanding and securing device vulnerabilities through automated bug report analysis. In *SEC'19: Proceedings of the 28th USENIX Conference on Security Symposium*.

[55] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.

[56] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[57] Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint embedding method for entity alignment of knowledge bases. In *China Conference on Knowledge Graph and Semantic Computing*. Springer, 3–14.

[58] Ningyu He, Lei Wu, Haoyu Wang, Yao Guo, and Xuxian Jiang. 2020. Characterizing code clones in the ethereum smart contract ecosystem. In *International Conference on Financial Cryptography and Data Security*. Springer, 654–675.

[59] Wei Hu, Qingheng Zhang, Zequn Sun, and Jiacheng Huang. 2019. MultiKE: a Multi-view Knowledge Graph Embedding Framework for Entity Alignment.. In *OM@ ISWC*. 189–190.

[60] Emrah Inan and Oguz Dikenelli. 2018. A Sequence Learning Method for Domain-Specific Entity Linking. In *Proceedings of the Seventh Named Entities Workshop, NEWS@ACL 2018, Melbourne, Australia, July 20, 2018*, Nancy F. Chen, Rafael E. Banchs, Xiangyu Duan, Min Zhang, and Haizhou Li (Eds.). Association for Computational Linguistics, 14–21. https://doi.org/10.18653/v1/w18-2403

[61] Yuning Jiang, Manfred Jeusfeld, and Jianguo Ding. 2021. Evaluating the Data Inconsistency of Open-Source Vulnerability Repositories. In *The 16th International Conference on Availability, Reliability and Security*. 1–10.

[62] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651* (2016).

[63] Elmar Kiesling, Andreas Ekelhart, Kabul Kurniawan, and Fajar Ekaputra. 2019. The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity. In *The Semantic Web – ISWC 2019*, Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon (Eds.). Springer International Publishing, Cham, 198–214.

[64] Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2020. From Zero to Hero: Human-In-The-Loop Entity Linking in Low Resource Domains. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 6982–6993. https://doi.org/10.18653/v1/2020.acl-main.624

[65] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[66] Kangjie Lu, Aditya Pakki, and Qiushi Wu. 2019. Detecting {Missing-Check} Bugs via Semantic-and {Context-Aware} Criticalness and Constraints Inferences. In *28th USENIX Security Symposium (USENIX Security 19)*. 1769–1786.

[67] Xin Mao, Wenting Wang, Yuanbin Wu, and Man Lan. 2021. From Alignment to Assignment: Frustratingly Simple Unsupervised Entity Alignment. *arXiv preprint arXiv:2109.02363* (2021).

[68] Xin Mao, Wenting Wang, Huimin Xu, Yuanbin Wu, and Man Lan. 2020. Relational reflection entity alignment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1095–1104.

[69] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. 2015. The attack of the clones: A study of the impact of shared code on vulnerability patching. In *2015 IEEE symposium on security and privacy*. IEEE, 692–708.

[70] Tam Thanh Nguyen, Thanh Trung Huynh, Hongzhi Yin, Vinh Van Tong, Darnbi Sakong, Bolong Zheng, and Quoc Viet Hung Nguyen. 2020. Entity alignment for knowledge graphs with multi-order convolutional networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[71] NVD. 2023. National Vulnerability Database. https://nvd.nist.gov/.

[72] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[73] Shichao Pei, Lu Yu, Guoxian Yu, and Xiangliang Zhang. 2020. REA: Robust cross-lingual entity alignment between knowledge graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2175–2184.

[74] Henrik Plate, Serena Elisa Ponta, and Antonino Sabetta. 2015. Impact assessment for vulnerabilities in open-source software libraries. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 411–420.

[75] Zhiyuan Qi, Ziheng Zhang, Jiaoyan Chen, Xi Chen, Yuejia Xiang, Ningyu Zhang, and Yefeng Zheng. 2021. Unsupervised Knowledge Graph Alignment by Probabilistic Reasoning and Semantic Embedding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Zhi-Hua Zhou (Ed.). ijcai.org, 2019–2025. https://doi.org/10.24963/ijcai.2021/278

[76] Radim Rehrek, Petr Sojka, et al. 2011. Gensim—statistical semantics in python. *Retrieved from genism. org* (2011).

[77] Wei Shen, Jiawei Han, Jianyong Wang, Xiaojie Yuan, and Zhenglu Yang. 2018. SHINE+: A General Framework for Domain-Specific Entity Linking with Heterogeneous Information Networks. *IEEE Trans. Knowl. Data Eng.* 30, 2 (2018), 353–366. https://doi.org/10.1109/TKDE.2017.2730862

[78] Anselm Strauss and Juliet Corbin. 1990. *Basics of qualitative research*. Sage publications.

[79] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. 2011. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *Proc. VLDB Endow.* 5, 3 (2011), 157–168. https://doi.org/10.14778/2078331.2078332

[80] Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*. Springer, 628–644.

[81] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping Entity Alignment with Knowledge Graph Embedding.. In *IJCAI*, Vol. 18. 4396–4402.

[82] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[83] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[84] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[85] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 349–357.

[86] Qiushi Wu, Yue Xiao, Xiaojing Liao, and Kangjie Lu. 2022. {OS-Aware} Vulnerability Prioritization via Differential Severity Analysis. In *31st USENIX Security Symposium (USENIX Security 22)*. 395–412.

[87] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2019. Jointly learning entity and relation representations for entity alignment. *arXiv preprint arXiv:1909.09317* (2019).

[88] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2020. Neighborhood matching network for entity alignment. *arXiv preprint arXiv:2005.05607* (2020).

[89] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. 2017. Neural network-based graph embedding for cross-platform binary code similarity detection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 363–376.

[90] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic Knowledge Graph Alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4564–4572.

[91] Guanqun Yang, Shay Dineen, Zhipeng Lin, and Xueqing Liu. 2021. Few-Sample Named Entity Recognition for Security Vulnerability Reports by Fine-Tuning Pre-trained Language Models. In *Deployable Machine Learning for Security Defense. MLHat 2021. Communications in Computer and Information Science*, Vol. 1482. Springer, Cham, 55–78. https://doi.org/10.1007/978-3-030-87839-9_3

[92] Kaisheng Zeng, Chengjiang Li, Lei Hou, Juanzi Li, and Ling Feng. 2021. A comprehensive survey of entity alignment for knowledge graphs. *AI Open* 2 (2021), 1–13.

[93] Xiang Zhao, Weixin Zeng, Jiuyang Tang, Wei Wang, and Fabian Suchanek. 2020. An experimental study of state-of-the-art entity alignment approaches. *IEEE Transactions on Knowledge & Data Engineering* 01 (2020), 1–1.

[94] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via knowledge embeddings. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[95] Qi Zhu, Hao Wei, Bunyamin Sisman, Da Zheng, Christos Faloutsos, Xin Luna Dong, and Jiawei Han. 2020. Collective Multi-Type Entity Alignment Between Knowledge Graphs. In *Proceedings of The Web Conference 2020*. https://doi.org/10.1145/3366423.3380289

[96] Renbo Zhu, Meng Ma, and Ping Wang. 2021. RAGA: Relation-aware Graph Attention Networks for Global Entity Alignment. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 501–513.