Uncovering CWE-CVE-CPE Relations with Threat Knowledge Graphs

ZHENPENG SHI, Boston University, Boston, USA
NIKOLAY MATYUNIN, Honda Research Institute Europe GmbH, Offenbach am Main, Germany
KALMAN GRAFFI, Technische Hochschule Bingen, Bingen, Germany
DAVID STAROBINSKI, Boston University, Boston, USA

Security assessment relies on public information about products, vulnerabilities, and weaknesses. So far, databases in these categories have rarely been analyzed in combination. Yet, doing so could help predict unreported vulnerabilities and identify common threat patterns. In this article, we propose a methodology for producing and optimizing a knowledge graph that aggregates knowledge from common threat databases (CVE, CWE, and CPE). We apply the threat knowledge graph to predict associations between threat databases, specifically between products, vulnerabilities, and weaknesses. We evaluate the prediction performance both in closed world with associations from the knowledge graph and in open world with associations revealed afterward. Using rank-based metrics (i.e., Mean Rank, Mean Reciprocal Rank, and Hits@N scores), we demonstrate the ability of the threat knowledge graph to uncover many associations that are currently unknown but will be revealed in the future, which remains useful over different time periods. We propose approaches to optimize the knowledge graph and show that they indeed help in further uncovering associations. We have made the artifacts of our work publicly available.

CCS Concepts: • **Security and privacy → Software security engineering**; • **Computing methodologies → Knowledge representation and reasoning;**

Additional Key Words and Phrases: Vulnerability, threat modeling, knowledge graph, link prediction

ACM Reference Format:
Zhenpeng Shi, Nikolay Matyunin, Kalman Graffi, and David Starobinski. 2024. Uncovering CWE-CVE-CPE Relations with Threat Knowledge Graphs. *ACM Trans. Priv. Sec.* 27, 1, Article 13 (February 2024), 26 pages. https://doi.org/10.1145/3641819

1 INTRODUCTION

Security assessment relies on public knowledge about existing vulnerabilities. For instance, **Software Composition Analysis (SCA)** scanners, such as the OSV-Scanner [9], rely on knowledge about disclosed vulnerabilities to identify vulnerable dependencies in the codebase. Likewise,

This work was supported in part by Honda Research Institute Europe GmbH and BU Hariri Institute Research Incubation Award (2020-06-006), the Boston University Red Hat Collaboratory (award # 2022-01-RH03), and by the US National Science Foundation under grants CNS-1717858, CNS-1908087, CCF-2006628, and EECS-2128517.
Authors' addresses: Z. Shi and D. Starobinski, Department of Electrical and Computer Engineering, Boston University, 8 St. Mary's Street, Boston, MA 02215, USA; e-mails: zpshi@bu.edu, staro@bu.edu; N. Matyunin, Honda Research Institute Europe GmbH, Carl-Legien-Str. 30, 63073 Offenbach/Main, Hesse, Germany; e-mail: nikolay.matyunin@honda-ri.de; K. Graffi, Technische Hochschule Bingen, Berlinstr. 109, 55411 Bingen, Hesse, Germany; e-mail: k.graffi@th-bingen.de.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 2471-2566/2024/02-ART13
https://doi.org/10.1145/3641819

Static Application Security Testing (SAST) tools utilize categorization knowledge to provide unified reports about discovered issues, and **Dynamic Application Security Testing (DAST)** scanners dynamically probe applications for known vulnerabilities. Several threat modeling tools (e.g., OWASP pytm [25], Threagile [27], IriusRisk [12]) use publicly disclosed weaknesses as predefined threats in their libraries [28]. By checking whether the conditions of said pre-defined threats are met, these tools can identify potential threats in a system, for which corresponding mitigations can be developed.

The security assessment activities mentioned above rely on *threat databases* that provide useful common knowledge on vulnerabilities, weaknesses, and products. Prominent examples include the **Common Vulnerabilities and Exposures (CVE)** [17], **Common Weakness Enumeration (CWE)** [18], and **Common Platform Enumeration (CPE)** [20] databases. The CVE lists publicly disclosed security vulnerabilities. Presently, it aggregates over 200,000 entries, with each CVE entry containing a short description and references to the disclosure reports. The CWE lists more than 900 generic software and hardware weakness types. CWE entries classify general instances of a vulnerability. For example, CWE-502 corresponds to a software weakness that deserializes untrusted data without sufficient verification. This entry relates to many examples being discovered and disclosed as CVE entries for specific products (e.g., CVE-2021-21348, which affects the XStream library). The CWE further provides a categorization of weaknesses into *views* and *categories* [19]: A category contains a set of weaknesses that share a common characteristic, and a view provides a specific perspective of examining CWE contents, such as weaknesses in software written in C. Last, the CPE provides a structured naming scheme for common products, including information technology systems, software, and packages, and lists the names of common products following the scheme.

The **National Vulnerability Database (NVD)** [24] provides associations between entries of the aforementioned databases. Table 1 shows an example of associations between the entries of the different threat databases [23]. Specifically, vulnerability CVE-2021-21348 is an instance of weaknesses CWE-400 and CWE-502. Its exploitation affects various products, as specified by associated CPE entries. Given software configurations or packages used in a system, associations by NVD can suggest relevant potential vulnerabilities and weaknesses. These associations are highly valuable to vulnerability scanners. Furthermore, as threat modeling often involves different abstraction levels of a system, CVE-CWE associations assist in mapping concrete threats (e.g., vulnerabilities in specific products) to abstract threats (e.g., general weaknesses) that can be used in higher-level threat models.

The creation of associations by the NVD is a lengthy process, since these associations are manually analyzed and vetted by security experts from credible organizations or corporations. As such, at any given time, associations provided by the NVD may be incomplete. As an illustration, in Table 1, we split the affected products of CVE-2021-21348 into two groups: those revealed before and after August 4, 2021. By August 4, 2021, only two CPE entries were known to be affected according to the NVD. However, many more affected entries were subsequently discovered and added. Being able to predict some of these associations in advance would be highly valuable, for instance, to identify and mitigate hidden vulnerabilities and weaknesses in a timely fashion.

In this article, we propose a concrete methodology to facilitate such predictions. Our approach is based on the analysis of associations between entries of the various threat databases. To enable such analysis, we produce a knowledge graph [6, 13, 26] out of existing databases (CVE, CWE, and CPE), which we refer to as a *threat knowledge graph*. We use the threat knowledge graph to predict associations between threat database entries that are currently hidden but will be revealed in the future, as illustrated in Figure 1. The artifacts of the threat knowledge graph are publicly available [30].

Table 1. Example of Prediction Results for CVE-2021-21348

CVE ID	CVE-2021-21348
Associated CWE	CWE-400, CWE-502
Description	XStream is a Java library to serialize objects to XML and back again. In XStream before version 1.4.16, there is a vulnerability that may allow a remote attacker to occupy a thread that consumes maximum CPU time and will never return. ...
Associated CPE by August 4, 2021	(1) cpe:a:xstream_project:xstream:*:*, (2) cpe:o:debian:debian_linux:*:*
Associated CPE after August 4, 2021	(1) cpe:o:fedoraproject:fedora:*:*, (2) cpe:a:oracle:retail_xstore_point_of_service:*:*, (3) cpe:a:oracle:webcenter_portal:*:*, (4) cpe:a:oracle:banking_platform:*:*, (5) cpe:a:oracle:communications_policy_management:*:*, (6) cpe:a:oracle:communications_billing_and_revenue_management_elastic_charging_engine:*:*, (7) cpe:a:oracle:mysql_server:*:*, (8) cpe:a:oracle:business_activity_monitoring:*:*, (9) cpe:a:oracle:communications_unified_inventory_management:*:*, (10) cpe:a:oracle:banking_virtual_account_management:*:*

The products are listed in the form of `cpe:part:vendor:product:target software:target hardware`. Using a threat knowledge graph generated with data available on August 4, 2021, we predict associations between CPE entries and the aforementioned vulnerability. Eight out of 10 products are successfully predicted (marked in blue), with one false positive prediction.

Table 2. Illustration of Threat Knowledge Graph Ranking Positive (Existent) and Negative (Non-existent) Associations

CPE-ID	CVE-ID	Score	Probability	Positivity
cpe:a:oracle:banking_platform:*:*	CVE-2020-35491	8.656979	0.999826	1
cpe:a:oracle:webcenter_portal:*:*	CVE-2021-21348	5.275653	0.994911	1
cpe:a:oracle:banking_platform:*:*	CVE-2021-21348	4.921300	0.992763	1
cpe:a:oracle:banking_platform:*:*	CVE-2020-3990	1.521459	0.820753	0
cpe:a:oracle:webcenter_portal:*:*	CVE-2021-31874	0.303383	0.575270	0

The predicted probabilities of the associations are normalized from their scores. By setting a proper threshold (represented by the dashed line), the five example associations can be successfully separated.

We achieve association prediction by applying a machine learning method known as *knowledge graph embedding*. As a result of the embedding process, a trained model can assign a score to each given association that indicates how likely it is for this association to belong to the knowledge graph. By enumerating over non-existent associations and ranking the resulting assigned scores, we can predict top candidates for future associations. Table 2 illustrates this process. Five triples are ranked by the probabilities assigned by the trained model and separated by a threshold. Accordingly, the model predicts that associations above the threshold will be revealed in the future. In this article, we conduct extensive evaluations using well-established metrics for knowledge graphs and show that this model indeed performs well in predicting threat associations.

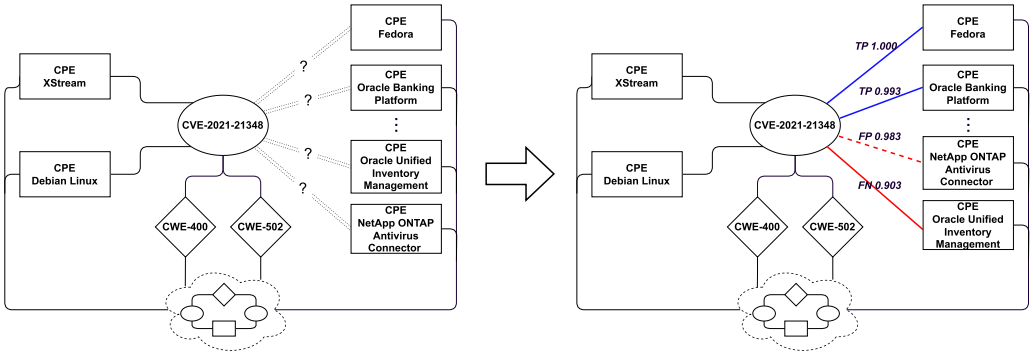


Fig. 1. Illustrations of the prediction process. For any given CVE (e.g., CVE-2021-21348) and its known associations to CPEs and CWEs at a certain date (left side), we aim to predict future associations to other CPEs and CWEs (right side). The cloud at the bottom represents the rest of CPE/CVE/CWE entries and the associations between them. An association will be predicted as positive if its estimated probability of existence (e.g., 0.993) is above a specified threshold (e.g., 0.960). The blue/red lines represent successful/unsuccessful predictions, respectively.

Our work, therefore, makes the following contributions:

(1) We propose and implement the concept of *threat knowledge graph* to aggregate knowledge from common threat databases. To do so:
   — We translate entries in threat databases and their associations into triples that form the knowledge graph.
   — We optimize the knowledge graph to improve its prediction capabilities.
(2) We embed the knowledge graph onto a vector space that can be used for link prediction. More specifically:
   — We compare several embedding models (i.e., TransE, DistMult, and ComplEx). We show that the TransE model consistently performs the best for the task at hand.
   — We perform a *closed-world* evaluation of the embedded knowledge graph using standard rank-based metrics, including the Mean Rank, the Mean Reciprocal Rank, and Hits@N scores. The evaluation demonstrates the high quality of the embedding (e.g., achieving a 0.606 Hits@10 score in predicting CVE-CPE associations).
(3) We further evaluate the ability of threat knowledge graphs to predict in an *open-world* setting associations between the entries in threat databases (i.e., between products, vulnerabilities, and weaknesses):
   — We use the embedded knowledge graph to predict new associations that were discovered after August 4, 2021.
   — We evaluate the prediction results using rank-based metrics as well as precision, recall, and F1-score. The metrics suggest that the model can make useful predictions (e.g., a 0.358 Hits@10 score in predicting newly added CVE-CWE associations, and an F1-score of 0.681 for CVE-CPE predictions).
   — We demonstrate the usefulness of the prediction capability over different time periods by producing and evaluating a threat knowledge graph from data available on March 29, 2022.
(4) We investigate approaches to further improve the quality of the knowledge graph, specifically by removing obsolete entries and incorporating data from other databases (i.e., CAPEC entries and CVSS vectors). We find that the prediction capability can indeed be improved, especially by removing old entries from the knowledge graph.

The rest of the article is organized as follows: We discuss related work on threat databases and knowledge graph approaches in Section 2. Next, in Section 3, we describe the creation of the threat knowledge graph and various methods for optimizing it. In Section 4, we detail the task of knowledge graph embedding, that is, translating the knowledge graph into vectors, and compare the performance of different embedding approaches. In Section 5, we use the embedded knowledge graph for predicting hidden associations between CPE, CVE, and CWE and evaluate the prediction capability with multiple metrics, both in closed-world and open-world settings. In Section 6, we investigate methods to further optimize the knowledge graph for improving its prediction capability. We conclude the article in Section 7.

2 RELATED WORK

There exists a rich literature on leveraging threat databases to derive insights about vulnerabilities and threats. In particular, the work in Reference [14] applies data mining on the CVE and CWE databases to derive interesting characteristics about software vulnerabilities, such as statistics of essential vulnerabilities. In Reference [32], after matching software names to existing CPE entries, the CVE-CPE associations are used for identifying vulnerabilities in an information system. This enables security risk assessment based on the identified vulnerabilities. The work in [40] uses historical data in threat databases for predicting the time until the next vulnerability will affect a specific software application. Prediction models for a few specific vendors and applications are developed, and challenges to develop prediction models for more general cases are pointed out, such as inconsistency between CPE and CVE entries. Compared to Reference [40], our work proposes a *general* prediction model, based on knowledge graph embedding.

Leveraging threat databases is non-trivial, since they are not always consistent and ready for use [40]. Thus, efforts have been made to automate associations of entries in different threat databases. The work in Reference [36] proposes an automated way of extracting related CPE entries from a CVE description to associate CPE and CVE entries. This is achieved by applying on CVE descriptions a natural language processing technique called **named entity recognition (NER)**.

Another approach is to treat CVE-CPE or CVE-CWE associations as an **extreme multi-label classification (XML)** problem, where CVE entries are classified into CPE or CWE classes. The work in Reference [2] first vectorizes CVE descriptions, then classifies the vectorized descriptions into 19 CWE classes by using a random forest algorithm based on selected features. In Reference [7], the authors introduce a novel Transformer-based learning framework called V2W-BERT. This framework leverages a pre-trained BERT model and outperforms existing approaches on automated CVE-CWE association, even in few-shot or zero-shot cases with limited training data.

On the CVE-CPE side, the work in Reference [15] proposes an XML approach called CHRONOS, which leverages data enhancement and time-aware adjustment steps. CHRONOS is shown to achieve good zero-shot performance in identifying libraries from vulnerability reports. In addition to the NER-based and XML-based approaches, a two-stage approach for mapping vulnerabilities and libraries is proposed in Reference [5]. Specifically, a weighted TF-IDF matching is first used to select candidates of affected libraries, and precise mapping is then completed using a BERT-FNN model.

A common limitation of the approaches mentioned above is that they treat each CVE entry separately. Therefore, implicit relations are difficult to identify. Looking at the description of CVE-2021-21348 in Table 1, one can easily see a relation between the keyword XStream and the `cpe:a:xstream project:xstream:*:*` CPE entry. However, the description does not suggest any relation to CPE entries like `cpe:a:oracle:banking platform:*:*`. In comparison, our

knowledge graph approach is able to extract not only explicit but also *implicit* relations, by aggregating knowledge from all entries.

We note that there also exists significant work on refining the CVE database. For instance, V-SZZ strives to refine the version of products affected by known CVE entries, by investigating the vulnerability-introducing commits [3]. The work in Reference [38] classifies CVE entries into common vulnerability types with two ensemble machine learning algorithms and shows that the Gradient Boost Classifier suits the task better.

Knowledge graphs have widely been used for reasoning over interrelated data for tasks such as recommendation [34] and question answering [11]. The work in Reference [6] reviews approaches and applications for knowledge graph reasoning. As threat databases are interrelated through associations between their entries, knowledge graph approaches are suitable for understanding and investigating these associations. To complete various tasks such as link prediction, knowledge graphs are embedded onto a vector space. Many embedding models have been proposed, including TransE [4], DistMult [37], and ComplEx [31]. In Reference [33], a comprehensive survey of the embedding approaches and the applications of embedded knowledge graphs is provided. The quality of embedding is typically evaluated by rank-based metrics [4].

The work in Reference [35] produces a knowledge graph from CVE and CWE entries and provides examples of querying the knowledge graph to analyze properties of vulnerabilities. The work in Reference [39] produces a knowledge graph using multiple threat databases, including the CVE and the CWE, and uses the knowledge graph to predict hidden links, which correspond to associations within a threat database or between different databases. Yet, the work in Reference [39] does not consider the CPE, though it is valuable for identifying threats based on products in the system. Moreover, evaluation of prediction is performed by splitting the knowledge graph into training and test sets under a closed-world assumption. Hence, it is unclear how the prediction model at a certain time performs for future data. In our work, we also focus on finding relations between CVE entries to specific products in the CPE to identify potential threats in products used in real systems. Another unique contribution of our work is in the validation of our methodology using historical data to demonstrate its prediction capabilities (i.e., an open-world setting).

A shorter, preliminary version of this article appeared in Reference [29]. Compared to the work in Reference [29], this manuscript includes the following significant new contributions:

— We demonstrate the usefulness of our model for not only predicting associations between CVE and CPE entries, but also between CVE and CWE entries. The new corresponding results are reported in Sections 5 and 6.
— We conduct a comprehensive evaluation of different embedding models for the task at hand, including the TransE, DistMult, and ComplEx models. We evaluate them using rank-based metrics and accordingly select the best model for predicting the associations. The comparison results of the embedding models are detailed in Table 8 and Table 9 in Section 4. We show that metrics obtained using TransE, with appropriate parameters, perform best, and consistently better than those provided in our preliminary work in Reference [29].
— In Reference [29], the model is evaluated by predicting new triples between August 4, 2021, and March 29, 2022, using data available before that time period. Here, we leverage recent updates of threat databases to investigate the prediction capability over time. Specifically, we find that the rank-based metrics *improve* as more new associations are added to the test set, as shown in Table 11 and Figure 4 in Section 5. This suggests that our prediction performance could be viewed as lower bounds on actual performance. We also show that, as the knowledge graph gets updated, the prediction capability of the model stays strong, as shown in Table 13.

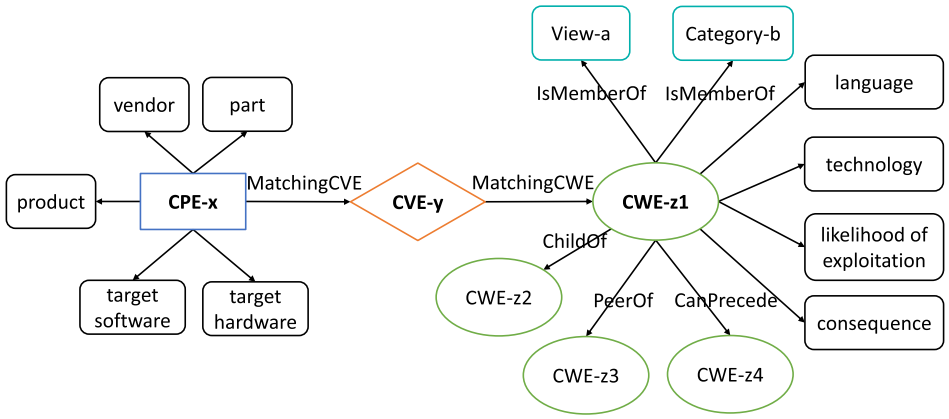


Fig. 2. Complete structure of the threat knowledge graph (a, b, x, y, z1-z4 represent the IDs of the corresponding entries).

— We evaluate the embedding and prediction results both for all triples in the graph as in Reference [29] and only certain *target triple types* (i.e., CVE→CPE and CVE→CWE triples). In Sections 4, 5, and 6, we evaluate rank-based metrics for these target triple types and find that the performance of the model for these target triples is even better than the performance averaged over all triples.
— In Section 3.1, we provide more details and examples on how to generate the knowledge graph.
— We investigate and evaluate multiple approaches to further improve the prediction capability of the knowledge graph in Section 6. The approaches include removing old CPE and CVE entries, adding entries from the CAPEC database, and adding CVSS vectors.

3 THREAT KNOWLEDGE GRAPHS

In this section, we describe our methodology for producing a knowledge graph out of the CPE, CVE, and CWE databases. We first introduce the structure of the knowledge graph. We give examples to show how the original data from the databases are used to generate triples in the knowledge graph. Alongside, we detail and justify optimizations performed on the knowledge graph.

3.1 Structure of the Knowledge Graph

A knowledge graph is represented with three sets (E, R, T), namely, the set of entities E, the set of relations R, and the set of triples T. Each triple $\tau \in T$ is of the form $\langle h, r, t \rangle$, where h, r, t, respectively, correspond to the head entity, the relation, and the tail entity. The structure of the knowledge graph is shown in Figure 2. Each threat database (CPE, CVE, and CWE) provides a list of *entries*, which correspond to products, vulnerabilities, or weaknesses. In addition to the name, each entry has *attributes* that provide further details, such as the vendor of a product. The entities in the knowledge graph are selected from the entries in the CPE, CVE, and CWE databases, as well as from the attributes of the entries, which are elaborated below. The arrows in Figure 2 point from the head entity to the tail entity.

There are three main types of entities, namely, product (CPE-x), vulnerability (CVE-y), and weakness (CWE-z), which are, respectively, based on the entries in the CPE, CVE, and CWE databases. The views and categories in the CWE are also added as entities in the knowledge graph. Each type

Table 3. Example of a CPE Entry

Name	Part	Vendor	Product	Target Software	Target Hardware
cpe:a:vim:vim:5.6:*:*	a	vim	vim	*	*

Table 4. Example of a CVE Entry

ID	Description	Matched CWE	Matched CPE
CVE-2021-29529	TensorFlow is an end-to-end open source platform for ...	CWE-131; CWE-193	cpe:a:google:tensorflow: 2.4.1:*:*

of entries mentioned is added to the knowledge graph with a unique name (by directly using the name of the entry or making slight adjustments, if needed).

In addition to the names of the CPE and CWE entries, key attributes are included in the knowledge graph as entities. For each CPE entry, we include the following five attributes (if present): the part (application, hardware, or operating system), the vendor (e.g., Google), the product (e.g., Chrome), the target software (e.g., Node.js), and the target hardware (e.g., x86). For each CWE entry, we include the following four attributes (if present): the language (e.g., JavaScript), the technology (e.g., web server), the likelihood of exploit (high/medium/low), and the consequence. The consequence describes which type(s) of security property are violated, for example, confidentiality and integrity.

The entities are connected by relations, as shown in Figure 2. There exist three types of relations: (i) the attribute of a node (e.g., the vendor of CPE-x); (ii) existing association between CPE, CVE, and CWE entries as provided by the NVD (e.g., CPE-x is related to CVE-y, and CVE-y is related to CWE-z); (iii) relations between weaknesses, views, and categories (e.g., CWE-z1 is a child of CWE-z2, and CWE-z1 is a member of View-a).

Next, we give a couple of examples that illustrate the process of constructing the knowledge graph. When considering a CPE entry, we add to the knowledge graph an entity with a simplified name to represent the entry, then add corresponding entities for all applicable attributes (part, vendor, product, target software, and target hardware). For example, consider the CPE entry "cpe:2.3:a:vim:vim:5.6:*:*:*:*:*:*:*," we first simplify its name as "cpe:a:vim:vim:5.6:*:*" by removing irrelevant information. The details of the entry "cpe:a:vim:vim:5.6:*:*" are shown in Table 3. We have four corresponding entities: "cpe:a:vim:vim:5.6:*:*," "a(part)," "vim(vendor)," "vim(product)." We also have three triples reflecting the relations between the entities: <cpe:a:vim:vim:5.6:*:*, HasPart, a(part)>, <cpe:a:vim:vim:5.6:*:*, HasVendor, vim(vendor)>, <cpe:a:vim:vim:5.6:*:*, HasProduct, vim(product)>. The name "cpe:a:vim:vim:5.6:*:*" also suggests that the product version is 5.6.

When considering a CVE entry, we only add an entity with a corresponding name, without additional attributes. Note that descriptions of CVE entries are unstructured text. Hence, these descriptions cannot be directly used as knowledge graph triples. As a result, we decided not to include them and leave their incorporation into the threat knowledge graph as an area for future work. Instead, we include associations from each CVE entry to CPE or CWE entries, which connect the three databases. We use relations "MatchingCVE" for the mapping between a CPE and CVE entry and "MatchingCWE" for the mapping between a CVE and CWE entry. For instance, for the entry CVE-2021-29529 in Table 4, we add the triples <cpe:a:google:tensorflow:2.4.1:*:*, MatchingCVE, CVE-2021-29529> and <CVE-2021-29529, MatchingCWE, CWE-131>.

For a CWE entry that represents a weakness, we add its name and attributes (language, technology, likelihood of exploitation, consequence) as entities in the knowledge graph, similar to the way we treat CPE entries. For example, for the CWE entry shown in Table 5, we have six enti-

Table 5. Example of a CWE Weakness Entry

ID	Name	Related Weakness	Language
CWE-1004	Sensitive cookie without "HttpOnly" flag	ChildOf: 732	Language-Independent

ID	Technology	Likelihood of Exploit	Consequence
CWE-1004	Web-based	Medium	Confidentiality; Integrity

Table 6. Example of a CWE View Entry

ID	Name	Type	Status	Has Member
View-1128	"CISQ Quality Measures (2016)"	Graph	Incomplete	CWE-1129; CWE-1130; CWE-1131; CWE-1132

Table 7. Example of a CWE Category Entry

ID	Name	Status	Has Member
Category-1217	"User Session Errors"	Draft	CWE-488; CWE-613; CWE-841

ties: "CWE-1004," "Language-Independent," "Web Based," "Medium," "Confidentiality," "Integrity." Note that each attribute may have multiple items, and each item will be a separate entity. The triples will be in the form <CWE-1004, HasTechnology, Web Based>, <CWE-1004, LikelihoodOf-Exploit, Medium>. A weakness may also be related to other weaknesses, with relations of the form "ChildOf," "CanPreceed," "PeerOf," and so on. Therefore, for the example CWE entry, we additionally include the triple <CWE-1004, ChildOf, CWE-732> in the knowledge graph.

Besides weaknesses, CWE also includes views and categories based on the format shown in Table 6 and Table 7. The views and categories provide useful characteristics about weaknesses. Hence, these relations are included in the knowledge graph. For the examples shown, this includes the triples <CWE-488, MemberOfCategory, Category-1217> and <CWE-1129, MemberOfView, View-1128>.

3.2 Optimizing the Knowledge Graph

The purpose of the threat knowledge graph is to extract information from associations between the threat databases. Therefore, instead of using all the entries from the databases, we make the following optimizations: (i) we merge CPE entries that have identical attributes, except for their version numbers; (ii) we remove CPE and CVE entries that contain no association information (e.g., CPE entries that are not associated with any CVE entry). We discuss next why these two optimizations are needed to improve the quality of the knowledge graph.

3.2.1 Merging CPE Entries. As of March 29, 2022, in total, the CPE has 858,409 entries, the CVE has 183,368 entries, while the CWE has 924 weakness entries, 326 category entries, and 46 view entries. The CWE only provides general information about threats, while the CPE and CVE are more specific. Hence, the size of the CWE is much smaller than those of the CVE and CPE. We also note that the CPE and CVE are regularly updated, while the CWE remains relatively stable.

In the CPE, the numbers of vendors and products (15,354 and 84,358, respectively) are significantly smaller than the total number of CPE entries, because many entries only differ by their version numbers. Among all the CPE entries, 742,868 are classified as "application," 77,151 are classified as "operating system," and 38,390 are classified as "hardware." We find that the vast majority

of the CPE entries are application-related, i.e., about 90%. These application-related entries tend to have many more version numbers than the other two types. If we merge CPE entries that are identical except for their version numbers, then the total number of CPE entries decreases from 858,409 to 89,895, which is only about 10% of the total number of entries. In this manner, the size of the CPE component of the knowledge graph is significantly compressed.

We further find that, after merging, a small portion of application-related CPE entries tend to have many more version numbers than other entries. We note that over 85% of the merged CPE entries have fewer than 10 version numbers, but about 3% of the entries have over 100 version numbers. Without merging, entries with many version numbers would have a too large weight on the knowledge graph, making it difficult to extract information from other valuable entries. Thus, merging CPE entries results in a more balanced knowledge graph.

3.2.2 Removal of Unconnected CPE and CVE Entries. After the merging of CPE entries, the resulting knowledge graph has 89,895 CPE entities, 183,368 CVE entities, and 924 CWE entities. Yet, many entries are not connected to entries in other databases. Specifically, only 33,427 CPEs (37.2%) are connected to CVEs, 149,582 CVEs (81.5%) are connected to CPEs, 106,861 CVEs (58.3%) are connected CWEs, and 288 CWEs (31.2%) are connected to CVEs.

We define an "unconnected" entity as a CPE/CVE/CWE entry that is not associated to entries in other threat databases, for example, a CVE entry that is not associated to either a CPE or a CWE entry. There exist many reasons why an unconnected CVE entry exists. To mention a few: (i) for a recent CVE entry, there may be a time lag due to manual analysis and association; (ii) the CVE entry is marked as "reject" though it is still stored in the NVD; (iii) the CVE entry is associated to "NVD-CWE-noinfo" or "NVD-CWE-other," which we do not consider as a meaningful CWE association; (iv) the CVE entry is associated to deprecated CPE entries and has not been remapped. The motivation behind creating a threat knowledge graph is to utilize relations between threat databases to discover new knowledge and get insights into common threats. However, unconnected entities provide no information in terms of relations between threat databases. Therefore, to increase the portion of valuable entries in the knowledge graph, we do not include unconnected CPE and CVE entries. Our evaluation in Section 4 validates the effectiveness of this optimization.

4 THREAT KNOWLEDGE GRAPH EMBEDDING

In this section, we undertake the task of knowledge graph embedding. The goal of embedding is to translate entities and relations in the knowledge graph to vectors in a continuous vector space, as illustrated in Figure 3. The embedded knowledge graph can then be used for various applications, including link prediction. We compare multiple state-of-the-art embedding models for our knowledge graph and determine the model with the best performance (TransE, in our case).

4.1 Embedding Model Training

To complete knowledge graph embedding, we need to adopt suitable embedding models. These models define how the entities and relations are translated into a vector space. Multiple models are available, and they mainly differ by their *scoring functions* [33]. A scoring function $f(\langle h, r, t\rangle)$ assigns a score to each triple $\langle h, r, t\rangle$. A good scoring function generates high scores for positive triples (i.e., triples that exist in the real world) and low scores for negative triples.

TransE, DistMult, ComplEx, and HolE are among the state-of-the-art knowledge graph embedding models. TransE [4] is based on translational distances in the vector space. With TransE, the entities h, t and the relation r are all translated into vectors $\mathbf{h}, \mathbf{t}, \mathbf{r}$ in real space $\mathbb{R}^k$, such that $\mathbf{h} + \mathbf{r}$ can approximate $\mathbf{t}$ for the positive triples. DistMult [37] represents a model that is based on semantic matching. It uses the trilinear dot product $\mathbf{h}^T \text{diag}(\mathbf{r})\mathbf{t}$ to capture similarities between

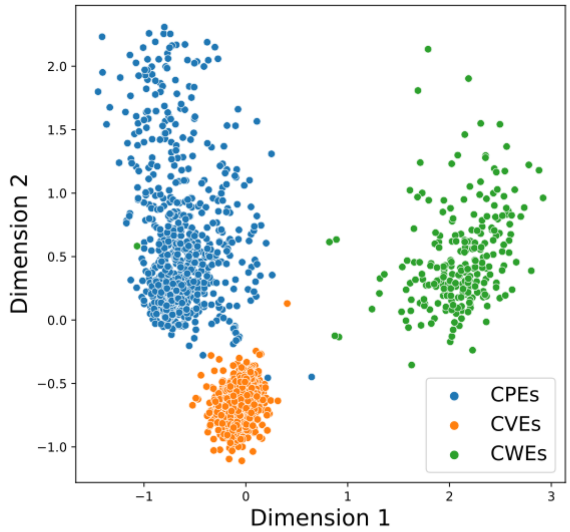


Fig. 3. Illustration of knowledge graph embedding. The CPE, CVE, and CWE entries shown in Figure 2 are embedded as vectors in a 200-dimensional vector space and then projected onto a 2-dimensional space by principal component analysis for illustration. Note that the relations and attributes in Figure 2 are also embedded as vectors but not shown here.

the triples. ComplEx [31] extends DistMult in complex space $\mathbb{C}^k$ with the Hermitian dot product $\text{Re}(\mathbf{h}^T \text{diag}(\mathbf{r})\bar{\mathbf{t}})$ to better capture relations while maintaining a low computational complexity. HolE [21] is another semantic-matching-based model but uses circular correlation. The work in Reference [10] shows that ComplEx and HolE are equivalent for link prediction. A detailed comparison of the embedding models above can be found in Reference [33]. Hence, we focus on comparing the TransE, DistMult, and ComplEx models in our evaluation.

In the training process, all the triples in the knowledge graph are considered as *positive triples*, while *negative triples* are the triples that do not exist in the knowledge graph. Note that our knowledge graph only has positive triples, but negative samples are also needed for training to help the embedding model differentiate between positive and negative triples. We follow the typical approach under the closed-world assumption, where negative triples are generated from positive triples $\langle h, r, t \rangle$, by replacing h or t with another random entity in the knowledge graph [33]. A generated negative triple is discarded if it belongs to the knowledge graph (a step called *filtering*).

The three embedding models used in our evaluations share a similar set of embedding parameters. For example, k for the dimension of the embedding vector space, *eta* for the number of negative triples generated against each positive one. The model can also utilize the same loss functions, including pairwise loss, max-margin loss, and negative log-likelihood loss. To make the results comparable, we test a large variety of combinations of the embedding parameters on each of the models and compare the models with their most suitable parameters identified using grid search.

4.2 Embedding Model Evaluation

We next evaluate the knowledge graph embedded by different models through standard metrics. To evaluate the embedded knowledge graph, we select 10,000 triples from the entire knowledge graph (566,279 triples) as the test set (triples in this set are not used for training). For each positive triple $\langle h, r, t \rangle$ in the test set, we perform the following: (i) generate all possible negative triples by replacing h or t with other entities; (ii) compute the scores of the positive triples and of the

generated negative triples; (iii) *rank* (order) the positive triples and the negative triples according to their scores (higher scores have lower rank). Positive triples in the test set should be ranked favorably compared to negative triples.

For quantitative evaluation of the embedding, the following metrics are commonly used (the mathematical definitions follow below) [4, 31, 37]: (1) the **mean rank (MR)**, which computes the average of ranks of the positive triples; (2) the **mean reciprocal rank (MRR)**, which computes the reciprocal mean of ranks of the positive triples; and (3) **hits-at-N-score (Hits@N)**, which computes how many positive triples are ranked within the top N positions. Mathematically,

$$MR = \frac{1}{|T|}\sum_{i=1}^{T} rank_{\langle h,r,t\rangle_i},$$

$$MRR = \frac{1}{|T|}\sum_{i=1}^{T} \frac{1}{rank_{\langle h,r,t\rangle_i}},$$

$$Hits@N = \frac{1}{|T|}\sum_{i=1}^{T} \mathbf{1}\left[\, rank_{\langle h,r,t\rangle_i} \leq N\right],$$

where T is a set of knowledge graph triples. For a good embedding, we expect low MR, high MRR, and high Hits@N scores.

By computing the MR, MRR, and Hits@N scores for all knowledge graph triples in the test set, we get an overall evaluation of the embedded knowledge graph. The knowledge graph includes multiple types of relations (as illustrated by the arrows in Figure 2). We also focus on computing metrics for associations between CVE and CPE entries, as well as those between CVE and CWE entries. In the test set, there are 5,612 CVE-CPE triples and 2,304 CVE-CWE triples, and we evaluate how well the embedding models learn these specific relations. For CVE-CPE and CVE-CWE triples, we generate the negative triples by substituting the tail entity of each triple with entities of the same type (*filtering* out existing positive triples). For example, given a CVE-CWE triple, we generate negative triples by substituting the CWE entity with all the other CWE entities. We denote the resulting set of positive and negative CVE-CWE triples as "CVE→CWE triples." The metrics for such triples can reflect how well we can predict the associated CWE entities for each CVE entity. Similarly, we have "CVE→CPE triples" to be evaluated. Moreover, we evaluate the average metrics for all types of triples, where the negative triples are generated by substituting both sides of all triples in the test set with all the other candidate entities. In this case, the resulting set of positive and negative triples is denoted as "all triples."

Table 8 summarizes the evaluation results. For each embedding model, we have three groups of data, which correspond to the average metrics for all triples, the metrics for CVE→CPE triples, and the metrics for CVE→CWE triples in the test set. The MR of 1,643 implies that, on average, the TransE model ranks a positive CVE→CPE triple at the 1,643rd position against the negative triples generated from it (with about 30,000 negative triples). The Hits@3 score of 0.314 implies that the TransE model ranks positive triples within the top-three positions 31.4% of the time.

In general, we find that translational-distance-based models (TransE) perform better than semantic-matching models (DistMult, ComplEx) for our knowledge graph. TransE performs the best among the three models in terms of all metrics. The likely reason is that while the names of CPE entries in our knowledge graph consist of meaningful words (e.g., cpe:a:google:tensorflow:*:*) that provide additional information by themselves, the names of CVE and CWE entries are only ID numbers and do not imply properties about the entries. If more semantic information about entries were added to the knowledge graph, then ComplEx might have better metrics. We also note that ComplEx is superior to DistMult for metrics that consider all triples and CVE-CPE triples only,

Table 8. Evaluation Results of Embedding Using Different Embedding Models for the Threat Knowledge Graph

Triple type	Model	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
All	TransE	**0.290**	**14,206**	**0.430**	**0.391**	**0.314**	**0.238**
	DistMult	0.243	20,091	0.376	0.332	0.259	0.196
	ComplEx	0.255	21,124	0.395	0.350	0.274	0.205
CVE→CPE	TransE	**0.424**	**1,643**	**0.626**	**0.573**	**0.468**	**0.345**
	DistMult	0.368	1,924	0.539	0.490	0.397	0.302
	ComplEx	0.392	1,944	0.581	0.531	0.424	0.321
CVE→CWE	TransE	**0.445**	**13**	**0.840**	**0.731**	**0.504**	**0.309**
	DistMult	0.315	228	0.518	0.444	0.338	0.244
	ComplEx	0.291	218	0.533	0.450	0.324	0.207

The knowledge graph uses data available on **August 4, 2021**. The metrics are computed for different types of triples, and TransE yields the best metrics in general. "All" indicates computing the average metrics for all triples in the test set. "CVE→CPE" ("CVE→CWE") indicates that only CVE-CPE (CVE-CWE) triples are evaluated, and the negative triples are generated by replacing the CPE (CWE) entity.

but slightly worse for CVE-CWE triples only. The reason is the CPE side of the knowledge graph has rich semantics in the CPE names and attributes. As an improved version of semantic-matching models, ComplEx better utilizes the semantics, including learning relations between CPE and CVE entities. In contrast, the relations between CVE and CWE entities are less suitable for semantic-matching models, and ComplEx learns no better than DistMult.

We also note that all three models yield better metrics for the target CVE→CPE and CVE→CWE triples, compared to results averaged over all triples. Intuitively, this suggests that CVE→CPE and CVE→CWE triples are easier to learn than other types of triples in the knowledge graph. Note that these two types of triples are exactly those we are interested in. Hence, our knowledge graph is well-suited for predicting the associations between CPE, CVE, and CWE entities. We also observed that the knowledge graph has good performance in predicting CVE→CPE and CVE→CWE relations, but generally performs worse in the opposite direction. The likely reason is that the number of CVE entries is significantly larger than the number of CPE and CWE entries. It is easier to make predictions starting from the side with more entries.

So far, we have computed the metrics based on a knowledge graph constructed with data available on August 4, 2021. To make sure that the comparison between the models truly reflects their performance difference, we perform a similar evaluation based on a knowledge graph constructed with data available on November 1, 2022. The results are shown in Table 9. We find that almost all metrics improve with the newer knowledge graph. Still, the relative performance between the three models is similar to that observed in Table 8. This suggests that our comparison results are consistent even though the knowledge graph grows over time.

Through comparison between the embedding models, we determine that TransE performs well for the task at hand. The metrics for the TransE model are sufficient for good prediction, as validated by our experiments in Section 5. We therefore selected TransE as the embedding model for the prediction of unknown associations. Note that our knowledge graph has many entities with relatively sparse connections, making it hard to get metrics as good as traditional tasks in knowledge graph embedding benchmarks [1]. Fine-tuning training parameters of the embedding model may help further improve these metrics.

Last, we investigate the influence of optimizing the knowledge graph under the TransE model, specifically removing unconnected CPE and CVE entities. The results are shown in Table 10. After optimizing the knowledge graph, the MRR increases from 0.232 to 0.290. The Hist@N scores also

Table 9. Evaluation Results of Embedding Using Different Embedding Models for the Threat Knowledge Graph

Triple type	Model	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
All	TransE	**0.303**	**15,785**	**0.445**	**0.403**	**0.328**	**0.249**
	DistMult	0.253	21,442	0.394	0.350	0.271	0.204
	ComplEx	0.272	20,633	0.415	0.372	0.294	0.220
CVE→CPE	TransE	**0.465**	**2,044**	**0.654**	**0.606**	**0.511**	**0.388**
	DistMult	0.367	2,158	0.556	0.508	0.400	0. 295
	ComplEx	0.414	2,149	0.604	0.559	0.453	0.338
CVE→CWE	TransE	**0.413**	**18**	**0.807**	**0.698**	**0.471**	**0.276**
	DistMult	0.281	228	0.501	0.411	0.304	0.209
	ComplEx	0.282	163	0.545	0.453	0.308	0.195

The knowledge graph uses data available on **November 1, 2022**. The TransE model still yields the best metrics in general, similar to the results in Table 8.

Table 10. Evaluation Results of Embedding with TransE Model before and after Removing Unconnected CPE and CVE Entities

	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
Unoptimized	0.232	19,642	0.363	0.326	0.261	0.178
Optimized	**0.290**	**14,206**	**0.430**	**0.391**	**0.314**	**0.238**

increase, indicating that positive triples are more likely to be found in top-N results. We conclude that removing unconnected CPE and CVE entries indeed improves the quality of our knowledge graph.

5 PREDICTION USING THREAT KNOWLEDGE GRAPH

In this section, we use the threat knowledge graph to uncover hidden associations between threat databases, namely, associations between CVE and CPE, and those between CVE and CWE. We evaluate the prediction results using rank-based metrics, as well as the precision, recall, and F1-score, which we define in the next subsection. Our results demonstrate that one can use historical data to correctly predict many associations of threats in the future.

5.1 Prediction Results and Metrics

We use entries from the CPE, CVE, and CWE databases available on August 4, 2021, to predict CVE-CPE and CVE-CWE associations that appeared from August 4, 2021, until March 29, 2022.

Toward this end, we produce a knowledge graph from the threat data available by August 4, 2021. Note that, using this approach, we can only predict associations between entries that already existed on August 4, 2021, and were not removed during the knowledge graph optimization. Associations between the entries added after August 4, 2021, cannot be predicted, since the knowledge graph does not include their information.

From August 4, 2021, to March 29, 2022, there were 41,583 newly added CVE-CPE associations. We find that out of the newly added associations, 4,134 of them are between entities that already existed in the August 2021 version of the knowledge graph. In practice, we would like to retrieve those triples from a large set of candidates. To evaluate prediction in this scenario, we create a test set that includes both positive and negative triples, and the goal is to correctly predict them as positive or negative. We use the 4,134 new triples as positive triples in the test set. However, there were 422 newly added CVE-CWE associations that are between entities that already existed,

which serve as positive triples in the test set. Following the approach detailed in Section 4.2, we generate the negative triples by substituting the CPE/CWE side of the positive triples with entities of the same type.

The prediction is based on the embedding model of the knowledge graph. Each triple is assigned a score by the scoring function of the embedding model. Scores are then normalized into estimated probabilities as the prediction output. A triple is predicted to be positive if its computed probability exceeds a *prediction threshold* α, where $0 \leq \alpha \leq 1$. A positive predicted triple means that the threat association (CVE-CPE or CVE-CWE) is currently hidden but is predicted to be revealed in the future.

To evaluate the prediction capability, we use two groups of metrics: (i) rank-based metrics, including MRR, MR, and Hits@N scores; (ii) precision, recall, and F1-score. The rank-based metrics can indicate the prediction capability of the knowledge graph. If the positive triples in the test set are mostly ranked higher than the negative triples, then setting a proper threshold α can effectively separate positive triples from negative triples, and the top-ranked triples can be seen as the prediction results.

Furthermore, as this is essentially a prediction task, we use the *precision*, *recall*, and *F1-score* to evaluate the prediction results. A high precision implies that a high fraction of the predicted positive triples is correct. A high recall implies that a high fraction of the positive triples is successfully identified. It is well-known that there exists a tradeoff between the precision and recall metrics, resulting in a precision-recall curve. To combine and balance these two metrics, the F1-score, which is defined as the harmonic mean of the precision and the recall, is often used. A high F1-score indicates one is able to simultaneously achieve relatively high precision and high recall.

5.1.1 Evaluation with Rank-based Metrics. In Table 11, we show the rank-based metrics when using the new triples to generate the test set. The results in Table 11 can be compared with the closed-world evaluation results in Table 8. As expected, when predicting new triples in an open-world setting, the results are worse than when predicting triples in a closed-world setting (i.e., where the tested triples are from the knowledge graph itself). Nonetheless, the rank-based metrics suggest that the knowledge graph is able to provide useful predictions, which can significantly save manual work in finding potential triples. Moreover, the metrics for new triples tend to improve over time, since some false positive triples in the evaluation turn out to be true positive in the future. For example, if the test set only covers triples until March 2022, then the true triples that were newly added between March 2022 and November 2022 will be determined as false positives in our evaluation results. To validate this statement, we conduct evaluations with test sets covering different time periods (namely, new triples appearing until March 29, 2022, July 14, 2022, or November 1, 2022). We find that the metrics are improved in general when the test set covers a longer period, as shown in Table 11. For instance, when evaluating CVE→CWE prediction, the MRR increases from 0.143 to 0.168 if the test set covers new triples until November 2022 rather than March 2022. The Hits@N scores also tend to increase, as illustrated in Figure 4. Hence, this confirms that the actual prediction capability improves over time.

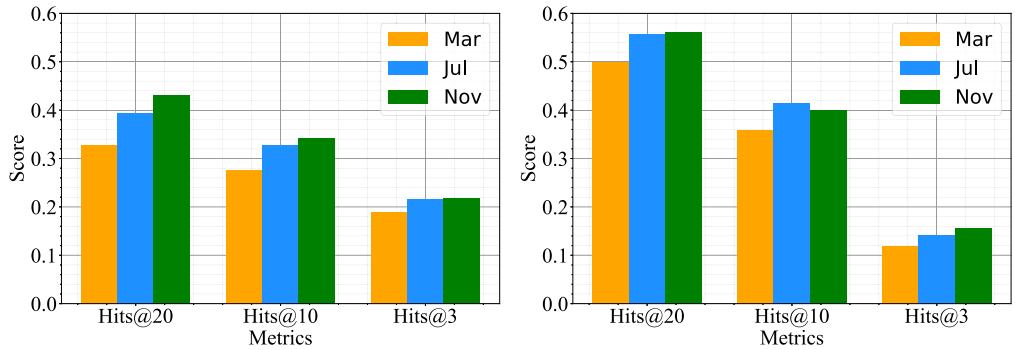
5.1.2 Evaluation with Precision, Recall, and F1-score. In this evaluation, for each unique CVE entry in new CVE-CPE triples, we generate 50 negative triples by substituting the CPE side. As a result, we get 66,984 CVE-CPE triples in total for testing prediction, where 4,134 triples are positive and 62,850 triples are negative. Similarly, we generate 20 negative triples for each unique CVE entry in new CVE-CWE triples by substituting the CWE side. We get 8,862 CVE-CWE triples in total for testing prediction, where 422 triples are positive and 8,440 triples are negative.

We remind that our prediction model uses a prediction threshold α. The evaluation metrics change with α, since only triples with estimated probability above α are predicted to be posi-

Table 11. Evaluation Results of Predicting Associations Based on Historical Data, Using the TransE Model

Triple type	Time range	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
CVE→CPE	Aug. 2021–Mar. 2022	0.159	2,068	0.327	0.276	0.189	0.091
	Aug. 2021–July 2022	0.186	1,814	0.394	0.328	0.217	0.110
	Aug. 2021–Nov. 2022	0.187	1,277	0.432	0.342	0.218	0.104
CVE→CWE	Aug. 2021–Mar. 2022	0.143	38	0.500	0.358	0.118	0.059
	Aug. 2021–July 2022	0.162	35	0.557	0.414	0.141	0.069
	Aug. 2021–Nov. 2022	0.168	37	0.561	0.401	0.157	0.074

The knowledge graph is trained with data collected from August 4, 2021, to March 29, 2022. The time range in the table specifies the time period during which the test set is selected. The metrics improve as the time range of the test set becomes wider, because the test set includes more newly added triples.



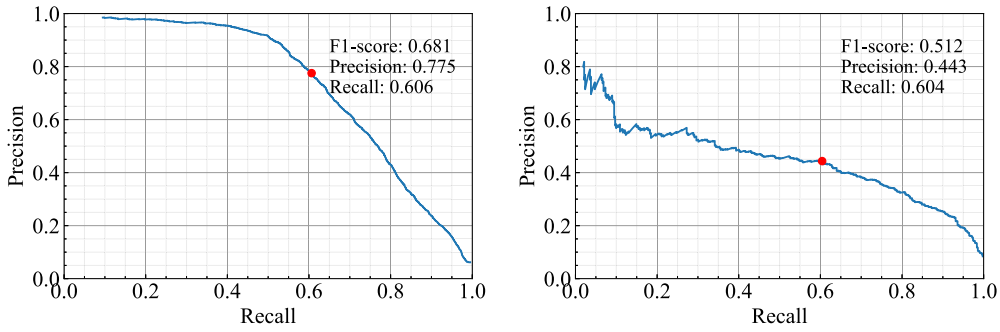
(a) CVE→CPE (b) CVE→CWE

Fig. 4. The Hits@N scores for predicting new associations, with the test set selected from different time periods. Note that only the end time of the periods is different (marked by different colors and all in 2022), while the start time is August 2021 for all three groups. In general, the Hits@N scores increase as the test set includes more newly added triples.

tive. For example, a high threshold α results in a higher precision but lower recall. We depict the precision-recall curve in Figure 5, which reflects the tradeoff between precision and recall as the threshold changes. When $\alpha = 0.981$, the F1-score for CVE-CPE prediction is maximized with a value of 0.681. The corresponding precision and recall are 0.775 and 0.606, respectively. This result means that about 78% of the predicted associations are correct, while 61% of unknown associations are retrieved. Similarly, the F1-score for CVE-CWE prediction is maximized at 0.512, with the corresponding precision as 0.443 and recall as 0.604. Note that the value of α that maximizes the F1-score only provides a reference. In practice, setting the proper value of α depends on the specific needs for precision and recall. For instance, a high value of α leading to a high precision might be preferred by a practitioner who wishes to have a reliable list of new associations that can easily be added to a threat database. However, a lower α leading to a high recall might be more suitable if a practitioner prefers to have an exhaustive candidate list of new associations that will be further verified manually. Note that changing the number of negative triples may impact the precision (and thereby the F1-score) but not the recall.

5.2 Examples of Prediction

We next illustrate the prediction results through a few examples. The first example is CVE-2021-0144 [22], detailed in Table 12. CVE-2021-0144 is a vulnerability related to Intel processors. We



(a) CVE→CPE (b) CVE→CWE

Fig. 5. Precision-recall curve of predicting CVE-CPE associations using threat knowledge graph. The curve reflects the tradeoff between precision and recall as the threshold α for positive prediction changes. When α decreases, the precision decreases and the recall increases. A reference point based on the maximized F1-score is marked in red.

Table 12. Example of Prediction Results for CVE-2021-0144

ID	CVE-2021-0144
Associated CWE	CWE-1188
Description	Insecure default variable initialization for the Intel BSSA DFT feature may allow a privileged user to potentially enable an escalation of privilege via local access.
Associated CPE by August 4, 2021	(1) cpe:h:intel:core_i5-7640x:*:*, (2) cpe:h:intel:core_i9-10900x:*:*, (3) cpe:h:intel:xeon_e5-1660_v4:*:*, (4) cpe:h:intel:xeon_platinum_8274:*:*, ...
Associated CPE after August 4, 2021	(1) cpe:h:intel:atom_c3000:*:*, (2) cpe:h:intel:core_i3-6006u:*:*, (3) cpe:h:intel:core_i9-11950h:*:*, (4) cpe:h:intel:xeon_e-2124:*:*, ...

Using a threat knowledge graph generated with data available on August 4, 2021, we predict associations between CPE entries and the aforementioned vulnerability. Using a prediction threshold $\alpha = 0.96$, all the 465 affected products are successfully predicted with 11 false positive predictions.

explicitly separate its associated CPE entries into two groups: before August 4, 2021, and from August 4, 2021, through March 29, 2022. We predict CPE entries associated to CVE-2021-0144 in the second group based on those in the first group. We only list a few known affected products (i.e., related CPE entries) due to the space limit. With $\alpha = 0.96$, all the 465 affected products subsequently added to the NVD database are successfully predicted, with only 11 false positive predictions.

A second, more complex example is CVE-2021-21348, detailed in Table 1, shown in the Introduction. CVE-2021-21348 is a vulnerability related to a Java library called XStream. Apart from XStream itself, products that use XStream are also affected by this vulnerability, such as Debian Linux and Oracle SQL Server. In this example, the names of CPE entries associated to CVE-2021-21348 before August 4, 2021, are not directly related to the names of the CPE entries added af-

terward. Nonetheless, by aggregating relations between all CVE and CPE entries, the knowledge graph can discover the implicit relation between XStream and Oracle SQL Server. As a result, the association between `cpe:a:oracle:mysql server:*:*` and CVE-2021-21348 can be uncovered accordingly.

Comparison to Other Approaches. By connecting various entries, the threat knowledge graph methodology is able to uncover many associations based on implicit relations, in contrast to prediction approaches that consider specific entries or relations separately. Intuitively, given a CVE-CPE association, we aim to predict associations to other CPE entries similar to the already associated CPE entry. We strive to design a general approach to uncover such associations, which involve hundreds of thousands of products, vulnerabilities, and weaknesses. The knowledge graph approach is a sensible solution to this problem, as the embedding process captures the attribute information, the semantics in the CPE names, as well as the huge amount of complex relations between the products, vulnerabilities, and weaknesses.

To illustrate this, we present in the following two simple approaches for CVE→CPE prediction and show that they do not perform as well as the knowledge graph approach. The first approach is based on semantic matching of CPE names to learn about their similarities, while the second one makes predictions by a simple logical look-up of existing CVE-CPE relations. It is worth noting that our adoption of knowledge graphs does not conflict with these simple approaches. Instead, we consider knowledge graphs as a more efficient and effective approach to leverage the same kinds of information that are also used by the simple approaches.

The first simple approach performs semantic matching of the CPE entries and predicts affected CPE entries that are semantically similar to the known vulnerable products. This would work on the example of CVE-2021-0144, shown in Table 12. Specifically, since CVE-2021-0144 is associated to Intel Core i9-10900X, one can predict a new association to Intel Core i9-11950H, since the semantic embeddings of the two products are similar. However, in many cases, the semantic information in the CPE names is insufficient to learn about product similarities. Consider the example of CVE-2021-21348 shown in Table 1. There is no way that one can find the similarities between XStream and Oracle SQL Server through simple semantic matching. As a result, one would miss many possible affected products by relying only on semantic matching.

Another approach is to use simple logical look-ups between existing CVE-CPE associations. Considering again CVE-2021-21348, we find that XStream and Oracle SQL Server are connected by a prior CVE entry (CVE-2019-10173). This implies that the similarities of CPE entries can be learned by looking up commonly related CVE entries that have already been revealed. More generally, this approach can be described as follows: Given CVE1 and known relations (e.g., CVE1-CPE1, CPE1-CVE2, and CVE2-CPE2), one infers that CPE2 is similar to CPE1 (since they are both affected by CVE2) and hence predicts an association between CVE1 and CPE2. In other words, the approach looks up an indirect path from CVE1 to CPE2. However, this approach may lead to a large number of false positives. Consider the example of CVE-2021-21348. From Table 1, we know that it affects Debian Linux. However, Debian Linux is affected by 5,099 other CVEs. Looking up existing CVE-CPE associations would yield over 18,000 predictions.

Although the look-up approach does not work in its naive implementation, it could be optimized by adding constraints on where to start the look-up. For instance, one can start the look-up from CPE entries affected by a limited number of CVEs to avoid paths involving very general CPE entries. If we set the limit to, say, 20, then the look-up from Debian Linux will be excluded, leaving us only the path from XStream (which is affected by 18 other CVEs). As a result, we get 13 candidate CPE entries that are potentially affected by CVE-2021-21348. This process is visualized in Figure 6, where CVE-2021-21348 is represented by the leftmost orange node, Debian Linux is represented

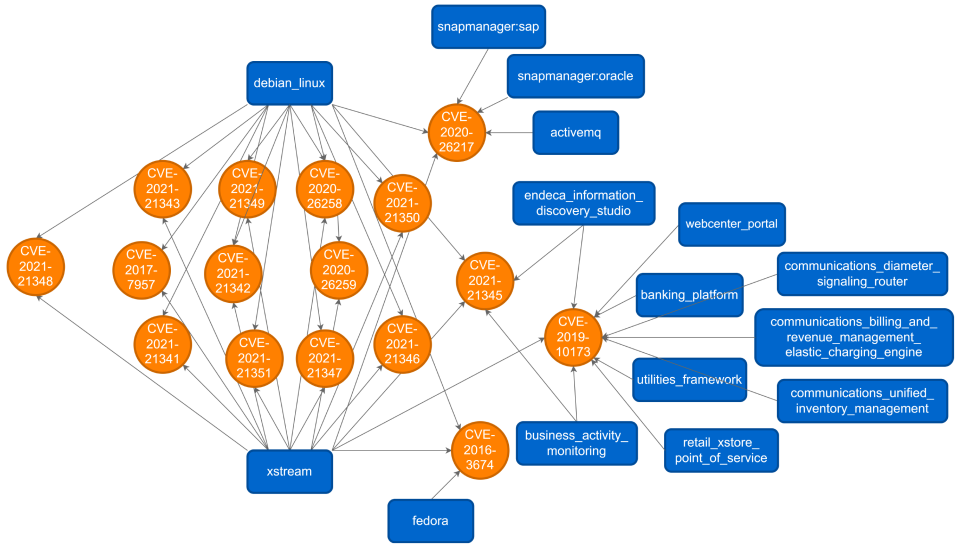


Fig. 6. An example of a mini knowledge graph starting from CVE-2021-21348 (leftmost orange node). The CVE entries are marked in orange, and the CPE entries are marked in blue. By August 4, 2021, CVE-2021-21348 had been associated to Debian Linux (top-left blue node) and XStream (bottom-left blue node). Expanding the graph from XStream, we can get a candidate list of CPE entries that share common CVEs with XStream and might also be affected by CVE-2021-21348.

by the top-left blue node, XStream is represented by the bottom-left blue node, and the predictions are made based on the rest of the blue nodes.

The predictions from the constrained look-up have a reasonable size, with 7 out of the 10 true positives successfully discovered. The rightmost orange node represents CVE-2019-10173, from which we find relations to a bunch of Oracle products that were associated to CVE-2021-21348 after August 4, 2023. However, we still experience the problem of false positives, as not all of these Oracle products are affected, and the three blue CPE nodes on the top in Figure 6 have no relation to CVE-2021-21348 at all. False negatives exist as well, as some affected CPE entries have more complex and implicit relations to CVE-2021-21348. An alternative approach could be to constrain look-ups to CVEs that affect both Debian Linux and XStream, but this results in worse predictions in this example, with only two true positives being discovered. Finding a proper set of general constraints that work for all the predictions appears challenging.

In comparison to the two simple approaches described above, the knowledge graph approach offers an automated way to learn many types of information, including the names and attributes of CVE, CWE, and CPE entries, as well as their complex relations. This is achieved by the embedding process and yields satisfying prediction capability, as shown in Section 5.1. Moreover, the knowledge graph can provide the likelihood of existence for predicted associations, while the two simple approaches above have no metric on the confidence of predictions. Note that the predictions are based purely on generally available information, that is, the CVE, CWE, CPE entries, and their associations provided by NVD. If there exist other sources of valuable information, then one can use them to augment the threat knowledge graph, as shown in Section 6, where we augment the threat knowledge graph with CVSS and CAPEC information. Given specific software applications, one could also generate a **Software Bill of Materials (SBOM)** that tracks the components for building the applications, as well as the relations between the components. SBOM provides more

Table 13. Comparison of the Prediction Metrics of Two Knowledge Graphs

Triple type	Knowledge graph date	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
CVE→CPE	August 4, 2021	0.159	2,068	0.327	0.276	0.189	0.091
	Mar 29, 2022	0.185	748	0.484	0.369	0.208	0.094
CVE→CWE	August 4, 2021	0.143	38	0.500	0.358	0.118	0.059
	Mar 29, 2022	0.175	38	0.552	0.383	0.185	0.073

For the knowledge graph using data by August 4, 2021, the test set is generated from the new triples from August 4, 2021, to March 29, 2022. For the knowledge graph using data by March 30, 2022, the test set is generated from the new triples from March 30, 2021, to November 1, 2022. The metrics suggest that our approach can consistently make useful predictions.

explicit CPE relations and should enhance prediction capability once integrated into the knowledge graph. The challenge is that, unlike the CPE database, there is currently no common database of SBOMs applicable to all products.

5.3 Usefulness of Prediction Capability over Different Periods

The prediction results above are based on the knowledge graph using data available on August 4, 2021. As the knowledge graph changes over time, in this subsection, we investigate whether the prediction capability remains useful over different time periods using rank-based metrics. To do so, we generate another threat knowledge graph, using data on March 29, 2022, and evaluate how well we can predict associations with it.

We generate two knowledge graphs, the first one using data available on August 4, 2021, and the second using data available on March 29, 2022. After the optimization introduced in Section 3.2, the knowledge graph using data on August 4, 2021, has 28,033 CPE entities, 237 CWE entities, 108,128 CVE entities connected to CPEs, and 84,307 CVE entities connected to CWEs. In comparison, the knowledge graph using data on March 29, 2022, has 33,427 CPE entities, 288 CWE entities, 149,582 CVE entities connected to CPEs, and 106,861 CVE entities connected to CWEs.

For each knowledge graph, we generate a test set from the new triples added in a few following months. We also keep the time span of the two test sets roughly the same (due to limitations of the data we have, we cannot make them exactly the same). Specifically, for the knowledge graph using data by August 4, 2021, the test set is generated from the new triples from August 4, 2021, to March 29, 2022; for the knowledge graph using data by March 29, 2022, the test set is generated from the new triples from March 29, 2021, to November 1, 2022. During March 29, 2021, to November 1, 2022, 7,597 new CVE-CPE associations and 1,041 new CVE-CWE associations are added that are between entities that already existed in the knowledge graph. We use the MRR, MR, and Hits@N scores to evaluate how well the triples in the test sets can be predicted.

The evaluation results of the two knowledge graphs are shown in Table 13. We find that the knowledge graph using data on March 2022 predicts new triples in the following months better than the knowledge graph using data on August 2021. Since the two test sets are generated from different time periods, we can expect the metrics to vary accordingly. Nonetheless, the MRR, Hits@10, Hits@3, and Hits@1 scores are roughly on the same level. Thus, we can conclude that a knowledge graph generated by our approach can consistently predict threat associations uncovered in the future. Moreover, we can expect the prediction metrics similar to those in Table 13, if not better.

5.4 Effectiveness of Merging CPE Entries

In this subsection, we justify the optimization made by merging CPE entries, which was introduced in Section 3.2.1. We show that merging CPE entries does indeed improve the prediction capability of the knowledge graph by comparing performance before and after the merging.

Table 14. The Prediction Metrics of the Knowledge Graph with Unmerged CPE Entries vs. the One with Merged CPE Entries (as in Table 11).

Triple type	Knowledge graph	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
CVE→CPE	Unmerged CPEs	0.036	23,485	0.105	0.073	0.032	0.019
	Merged CPEs	**0.159**	**2,068**	**0.327**	**0.276**	**0.189**	**0.091**
CVE→CWE	Unmerged CPEs	0.097	47	0.377	0.224	0.086	0.026
	Merged CPEs	**0.143**	**38**	**0.500**	**0.358**	**0.118**	**0.059**

Both knowledge graphs use data by August 4, 2021, with test sets generated from the new triples from August 4, 2021, to March 29, 2022. The metrics suggest that optimization by merging CPE entries significantly improves the prediction capability of hidden relations.

So far, we have been using the optimized knowledge graph from the threat data available by August 4, 2021. Details about the optimized knowledge graph are discussed in Section 5.1. For comparison, the same data is used to generate another knowledge graph, where we do not merge the CPE entries that are identical except for their version numbers. In the following, based on whether the CPE entries are merged, we refer to this new knowledge graph as the "unmerged knowledge graph" and the threat knowledge graph before as the "merged knowledge graph." The unmerged knowledge graph follows a similar structure to the one in Figure 2, but the CPE entries are more specific, as the version information is included. Similar to the merged knowledge graph, we remove the unconnected CPE and CVE entries in the unmerged knowledge graph, so we can focus on the effectiveness of merging the CPE entries.

Generated from the same data, the unmerged knowledge graph has about three million triples and is roughly eight times larger than the merged knowledge graph. Although the unmerged knowledge graph provides more detailed CVE-CPE associations, our results show that it actually performs worse in predicting hidden relations. Due to the size of the unmerged knowledge graph, another benefit of merging the CPE entries is the reduced time in the training and evaluation of the knowledge graph. To compare the prediction capability of the two knowledge graphs, we use the same open-world setting. Specifically, the new CVE-CPE and CVE-CWE associations between August 4, 2021, and March 29, 2022, are used for evaluation. The test set for the merged knowledge graph was presented in Section 5.1. For the unmerged knowledge graph, we have a test set consisting of 8,066 new CVE-CPE associations and 422 new CVE-CWE associations. As before, we use rank-based metrics to evaluate how well the knowledge graphs can distinguish the new positive triples from other negative triples.

The evaluation results on predicting new associations are summarized in Table 14. From the table, we find that the merged knowledge graph significantly outperforms the unmerged one in all metrics. Specifically, the unmerged knowledge graph only has an MRR of 0.036 on CVE→CPE prediction, which indicates nearly no prediction capability. The reason is that there are too many similar CPE entries that only differ in version numbers, resulting in a lot of redundant information in the knowledge graph. The unmerged knowledge graph is also worse at CVE→CWE prediction, with an MRR of 0.097. This is because the CVE-CWE side of the knowledge graph represents a smaller fraction of the unmerged knowledge graph compared to the merged case, and CVE-CWE associations cannot properly be learned when training on the entire knowledge graph.

In addition to the evaluation of the prediction capability using the new associations, we evaluate the embedding of the unmerged knowledge graph in a closed-world setting. Similar to the process described in Section 4.2, we randomly split the unmerged knowledge graph into training and test sets, where the test set has 30,000 triples and is evaluated using rank-based metrics. We find that, although the rank-based metrics of the unmerged knowledge graph can be pretty good when evaluating the embedded model, they do not translate to good prediction of new associations. For

example, the unmerged knowledge graph can achieve up to a good MRR of 0.739 on CVE→CPE triples, but falls short when predicting the new triples added afterward as shown in Table 14. The reason is that triples in the test set have already been partially learned in the training process. For example, consider some CVE→CPE triple in the test set. It is likely that similar triples were already used for training—the only difference between them being the minor version of the CPE entry.

In summary, by comparing the unmerged and merged knowledge graphs on predicting newly added associations, we find that merging CPE entries (as detailed in Section 3.2.1) is necessary to achieve useful prediction capability. While this comes at the expense of losing detailed version information, we believe that striking such a tradeoff is worthwhile. Indeed, our knowledge graph methodology allows learning implicit, non-obvious associations between different products. Nonetheless, there might exist other approaches for merging CPE entries that strike a better tradeoff between prediction capability and detailed product version information, which we leave as an interesting area for future work.

6 IMPROVING THE THREAT KNOWLEDGE GRAPH

In this section, we further optimize the threat knowledge graph for predicting CVE-CPE and CVE-CWE associations. The approaches include removing old entries and adding more useful threat information (such as the CAPEC database and CVSS vectors). We first provide details on each of these three approaches and then evaluate their effectiveness using rank-based metrics.

6.1 Removal of Old Entries

By observing the dates of creating CVE entries and associating them with CPE entries, we find that most associations are added within two years after a CVE entry is created. For example, over 79% of the associations added between August 4, 2021, and July 14, 2022, only involve CVE entries created since 2019, and over 94% of them only involve CVE entries created since 2016. As a result, old entries tend to become obsolete if we only care about predicting future associations. Thus, we propose to improve the prediction capability of the knowledge graph by removing old entries before a certain date.

Specifically, we first remove all the CVE entries that were created before 2016. The removal of old CVE entries results in new unconnected CPE entries, and we remove them as well. Since the CWE entries are relatively stable and interconnected with each other, we do not modify the CWE side of the knowledge graph. As a result, the new threat knowledge graph has much fewer CPE and CVE entries. The total number of triples decreases by 34.8%, from 380,138 to 247,705. We expect that the remaining entities and relations are more closely related to the associations added in the future.

6.2 Adding the CAPEC Database

The **Common Attack Pattern Enumeration and Classification (CAPEC)** database [16] enumerates known attack patterns employed to exploit weaknesses from the adversary's perspective. The CAPEC entries are associated with the CWE entries, such that we can add them to provide more useful information. We add the CAPEC entries as well as their attributes to the knowledge graph as entities. These new entities introduce new relations, including relations between CAPEC entries and their attributes, and associations between CAPEC entries and CWE entries. As a result, we have richer information on the CWE side of the knowledge graph, which could be useful for predicting the associations between CPE, CVE, and CWE entries. The CAPEC part of the knowledge graph is illustrated within the yellow border on the right in Figure 7.

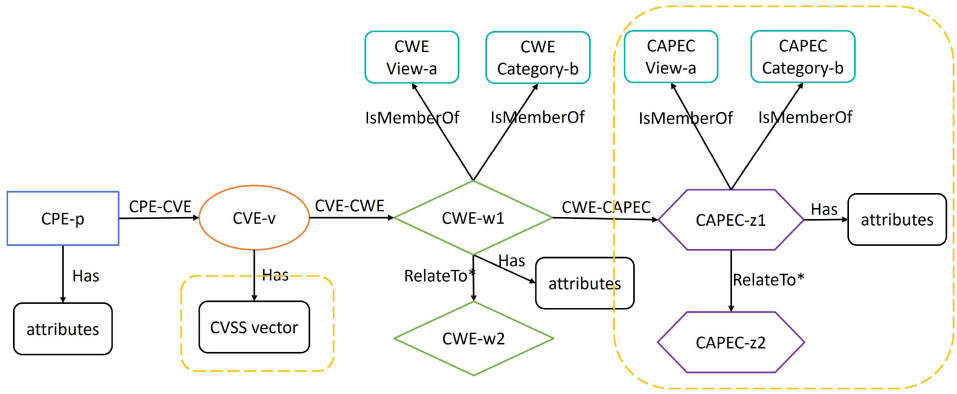


Fig. 7. Structure of the threat knowledge graph with CAPEC and CVSS vectors. The parts within the yellow borders represent the new data added to the knowledge graph.

6.3 Adding CVSS Vectors

In the threat knowledge graph, no attributes of CVE entries are included. We can enrich the information about the CVE entries by using the CVSS vectors provided by NVD. Each CVE entry is associated with a CVSS vector, which evaluates the vulnerability in three metric groups. Each CVSS metric takes a predefined value. For example, the **attack vector (AV)** metric is of the form **network (N), adjacent (A), local (L)**, or **physical (P)**, and the **attack complexity (AC)** metric is of the form **low (L)** or **high (H)**. Thus, by including CVSS vectors, we add more information on CVE entries to the knowledge graph. This information should help the embedding models better learn the similarities between different CVE entries. The CVSS part of the knowledge graph is illustrated within the yellow border on the left in Figure 7.

6.4 Evaluation of the Approaches

Next, we evaluate the effectiveness of the three approaches introduced above. For each approach, we generate a new threat knowledge graph, and we compare them with the "base" knowledge graph detailed in Section 3. We conduct both closed-world evaluations, where the test set is split from the knowledge graph itself, and open-world evaluations, where the test set is from the new triples added afterward.

The evaluation results are shown in Table 15 and Table 16. For closed-world evaluations, we find that the knowledge graph using data after 2016 and the knowledge graph with CAPEC yield similar metrics to those of the base knowledge graph. However, the knowledge graph with CVSS vectors has better embedding metrics on average for all types of triples but still does not outperform the base knowledge graph when it comes to CVE→CPE and CVE→CWE triples. Overall, the closed-world evaluation results suggest that the proposed approaches will not let the knowledge graph learn existing CVE-CPE and CVE-CWE associations better.

Although the closed-world embedding metrics are not significantly improved by the proposed approaches, the open-world evaluations show that some of the approaches indeed help predict the associations that will be revealed and added in the future. For CVE→CPE triples, removing old entries before 2016 brings the most significant improvement. For instance, the MRR increases by 35%, and the Hits@10 score increases by 34%. For CVE→CWE triples, both removing the old entries and adding CVSS vectors significantly improve the embedding metrics. Adding the CAPEC database, on the contrary, does not improve the prediction performance, possibly due to the fact that CWE-CAPEC associations are not strongly related to CVE-CPE and CVE-CWE associations.

Table 15. Closed-world Evaluation Results of Embedding Using TransE Model

Triple type	Knowledge graph	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
All	Base	0.290	14,206	0.430	0.391	0.314	0.238
	After 2016	0.297	7,340	0.446	0.400	0.320	0.242
	With CAPEC	0.277	13,507	0.421	0.378	0.300	0.224
	With CVSS	**0.391**	**4,334**	**0.493**	**0.469**	**0.430**	**0.339**
CVE→CPE	Base	0.424	1,643	**0.626**	**0.573**	**0.468**	0.345
	After 2016	**0.426**	**1,278**	0.620	0.566	0.460	**0.354**
	With CAPEC	0.420	1,810	0.610	0.563	0.460	0.346
	With CVSS	0.380	1,565	0.602	0.549	0.438	0.290
CVE→CWE	Base	**0.445**	13	0.840	**0.731**	**0.504**	**0.309**
	After 2016	0.415	17	0.804	0.686	0.464	0.285
	With CAPEC	0.411	15	0.813	0.701	0.451	0.280
	With CVSS	0.441	**11**	**0.860**	0.719	0.485	0.307

Four knowledge graphs are compared: the base knowledge graph introduced in Section 3, the graph using data after 2016, the graph with CAPEC, and the graph with CVSS vectors. The best metrics for each triple type are marked by bold text.

Table 16. Open-world Evaluation Results of Predicting Associations Based on Historical Data Using TransE Model

Triple type	Knowledge graph	MRR	MR	Hits@20	Hits@10	Hits@3	Hits@1
CVE→CPE	Base	0.159	2,068	0.327	0.276	0.189	0.091
	After 2016	**0.215**	**1,488**	**0.429**	**0.341**	**0.241**	**0.146**
	With CAPEC	0.168	2,214	0.343	0.300	0.205	0.095
	With CVSS	0.170	1,712	0.324	0.272	0.202	0.106
CVE→CWE	Base	0.143	38	0.500	0.358	0.118	0.059
	After 2016	**0.203**	37	0.597	0.432	**0.219**	**0.098**
	With CAPEC	0.147	39	0.517	0.351	0.140	0.052
	With CVSS	0.192	**36**	**0.628**	**0.443**	0.194	0.083

Four knowledge graphs are compared: the base knowledge graph introduced in Section 3, the graph using data after 2016, the graph with CAPEC, and the graph with CVSS vectors. The best metrics for each triple type are marked by bold text.

Nonetheless, adding CAPEC provides us with the bonus of predicting new CWE-CAPEC associations. The likely reason that adding CVSS vectors works better for CVE→CWE triples is that associations between CVE and CWE entries are much denser than those between CPE and CVE entries (e.g., a CWE entry can be associated with hundreds of CVE entries).

7 CONCLUSION

In this work, we introduced a methodology for predicting future associations between products, vulnerabilities, and weaknesses using threat knowledge graphs. We explained how to generate and optimize such threat knowledge graphs and how to embed them into a vector space. Through comparison between multiple embedding models, we found that the TransE model performs the best for our task. We validated our approach with both closed-world and open-world evaluations. In open-world evaluation, we demonstrated good rank-based metrics on the associations revealed between August 4, 2021, and November 1, 2022, based on historical data by August 4, 2021. This suggested that we can make good predictions on future associations with the help of the knowledge graph. Moreover, we showed that the prediction capability of our knowledge graph improves over

time and is further improved with judicious optimizations, such as removing outdated entries. One can utilize the predicted associations between entries of different threat databases in various ways. For example, if a specific product is used in a system, then a threat modeler can list both existing and predicted vulnerabilities associated with the product. Our prediction methodology can also help prioritize manual verification of potential vulnerabilities of products.

Directions for future work include developing an embedding model for threat knowledge graphs that is specifically tailored to threat databases and could further improve prediction performance. Another direction is to investigate more approaches to optimize the knowledge graph, for example, incorporating the descriptions of CVE entries into the knowledge graph. The CVE descriptions indeed provide rich information on the vulnerabilities, but the challenge is that they are unstructured and require different techniques (e.g., NLP) to convert them into knowledge graph triples. It is worth noting that we aim to develop a general approach to predict the relations between vulnerabilities, weaknesses, and products, such that it can work not only on the CVE, CWE, and CPE, but on other threat databases as well (e.g., the OSV database [8]). In Section 6, we show the potential of augmenting the threat knowledge graph with other sources of information, such as the CAPEC database and CVSS vectors. It would also be an interesting future direction to integrate other vulnerability and product/package databases into the knowledge graph.

REFERENCES

[1] Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. 2020. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *ACM SIGMOD International Conference on Management of Data*. 1995–2010.

[2] Masaki Aota, Hideaki Kanehara, Masaki Kubo, Noboru Murata, Bo Sun, and Takeshi Takahashi. 2020. Automation of vulnerability classification from its description using machine learning. In *IEEE Symposium on Computers and Communications (ISCC'20)*. IEEE, 1–7.

[3] Lingfeng Bao, Xin Xia, Ahmed E. Hassan, and Xiaohu Yang. 2022. V-SZZ: Automatic identification of version ranges affected by CVE vulnerabilities. In *44th International Conference on Software Engineering*. 2352–2364.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.* 26 (2013).

[5] Tianyu Chen, Lin Li, Bingjie Shan, Guangtai Liang, Ding Li, Qianxiang Wang, and Tao Xie. 2023. Identifying vulnerable third-party libraries from textual descriptions of vulnerabilities and libraries. *arXiv preprint arXiv:2307.08206* (2023).

[6] Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Applic.* 141 (2020), 112948.

[7] Siddhartha Shankar Das, Edoardo Serra, Mahantesh Halappanavar, Alex Pothen, and Ehab Al-Shaer. 2021. V2W-BERT: A framework for effective hierarchical multiclass classification of software vulnerabilities. In *IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA'21)*. IEEE, 1–12.

[8] Google. 2023. OSV—A distributed vulnerability database for Open Source. Retrieved from https://osv.dev/

[9] Google. 2023. OSV-Scanner. Retrieved from https://github.com/google/osv-scanner

[10] Katsuhiko Hayashi and Masashi Shimbo. 2017. On the equivalence of holographic and complex embeddings for link prediction. *arXiv preprint arXiv:1702.05563* (2017).

[11] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *12th ACM International Conference on Web Search and Data Mining*. 105–113.

[12] IriusRisk. 2023. Irius Risk | Automated Threat Modeling Tool. Retrieved from https://www.iriusrisk.com/

[13] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 2 (2022), 494–514.

[14] Xiang Li, Jinfu Chen, Zhechao Lin, Lin Zhang, Zibin Wang, Minmin Zhou, and Wanggen Xie. 2017. A mining approach to obtain the software vulnerability characteristics. In *5th International Conference on Advanced Cloud and Big Data (CBD'17)*. IEEE, 296–301.

[15] Yunbo Lyu, Thanh Le-Cong, Hong Jin Kang, Ratnadira Widyasari, Zhipeng Zhao, Xuan-Bach D Le, Ming Li, and David Lo. 2023. Chronos: Time-aware zero-shot identification of libraries from vulnerability reports. *arXiv preprint arXiv:2301.03944* (2023).

[16] MITRE. 2023. Common Attack Pattern Enumerations and Classifications (CAPEC). Retrieved from https://capec.mitre.org

[17] MITRE. 2023. Common Vulnerabilities and Exposure (CVE). Retrieved from https://cve.mitre.org
[18] MITRE. 2023. Common Weakness Enumeration (CWE). Retrieved from https://cwe.mitre.org
[19] MITRE. 2023. CWE Glossary. Retrieved from https://cwe.mitre.org/documents/glossary/index.html
[20] MITRE. 2023. Official Common Platform Enumeration (CPE) Dictionary. Retrieved from https://nvd.nist.gov/products/cpe
[21] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI Conference on Artificial Intelligence*.
[22] NVD. 2023. CVE-2021-0144 Detail. Retrieved from https://nvd.nist.gov/vuln/detail/CVE-2021-0144
[23] NVD. 2023. CVE-2021-21348 Detail. Retrieved from https://nvd.nist.gov/vuln/detail/CVE-2021-21348
[24] NVD. 2023. National Vulnerability Database (NVD). Retrieved from https://nvd.nist.gov/general
[25] OWASP. 2023. pytm: A Pythonic framework for threat modeling. Retrieved from https://github.com/izar/pytm
[26] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semant. Web* 8, 3 (2017), 489–508.
[27] Christian Schneider. 2023. Threagile. Retrieved from https://threagile.io/
[28] Zhenpeng Shi, Kalman Graffi, David Starobinski, and Nikolay Matyunin. 2022. Threat modeling tools: A taxonomy. *IEEE Secur. Privac.* 20, 04 (2022), 29–39. DOI: https://doi.org/10.1109/MSEC.2021.3125229
[29] Zhenpeng Shi, Nikolay Matyunin, Kalman Graffi, and David Starobinski. 2022. Uncovering product vulnerabilities with threat knowledge graphs. In *IEEE Secure Development Conference (SecDev'22)*. IEEE, 84–90.
[30] Zhenpeng Shi, Nikolay Matyunin, Kalman Graffi, and David Starobinski. 2023. Threat Knowledge Graph. Retrieved from https://github.com/nislab/threat-knowledge-graph
[31] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *33rd International Conference on Machine Learning*. PMLR, 2071–2080.
[32] Roman Ushakov, Elena Doynikova, Evgenia Novikova, and Igor Kotenko. 2021. CPE and CVE based technique for software security risk assessment. In *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'21)*. IEEE, 353–356.
[33] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.* 29, 12 (2017), 2724–2743.
[34] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.
[35] Yongfu Wang, Ying Zhou, Xiaohai Zou, Quanqiang Miao, and Wei Wang. 2020. The analysis method of security vulnerability based on the knowledge graph. In *10th International Conference on Communication and Network Security*. 135–145.
[36] Emil Wåreus and Martin Hell. 2020. Automated CPE labeling of CVE summaries with machine learning. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 3–22.
[37] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).
[38] Veneta Yosifova. 2021. Vulnerability type prediction in common vulnerabilities and exposures database with ensemble machine learning. In *International Conference Automatics and Informatics (ICAI'21)*. IEEE, 146–149.
[39] Liu Yuan, Yude Bai, Zhenchang Xing, Sen Chen, Xiaohong Li, and Zhidong Deng. 2021. Predicting entity relations across different security databases by using graph attention network. In *IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC'21)*. IEEE, 834–843.
[40] Su Zhang, Xinming Ou, and Doina Caragea. 2015. Predicting cyber risks through national vulnerability database. *Inf. Secur. J.: Global Perspect.* 24, 4-6 (2015), 194–206.

Received 20 April 2023; revised 14 October 2023; accepted 13 December 2023