Uncovering Product Vulnerabilities with Threat Knowledge Graphs

Zhenpeng Shi
Boston University
Boston, USA
zpshi@bu.edu

Nikolay Matyunin
Honda Research Institute Europe GmbH
Offenbach am Main, Germany
nikolay.matyunin@honda-ri.de

Kalman Graffi
Honda Research Institute Europe GmbH
Offenbach am Main, Germany
Kalman.Graffi@honda-ri.de

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

***Abstract*—Threat modeling and security assessment rely on public information on 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 paper, we propose a methodology for producing and optimizing a knowledge graph that aggregates knowledge from common threat databases (CPE, CVE, and CWE). We apply the threat knowledge graph to predict associations between threat databases, specifically between products and vulnerabilities. We evaluate the prediction performance based on historical data, using precision, recall, and F1-score metrics. We demonstrate the ability of the threat knowledge graph to uncover many associations that are currently unknown but will be revealed in the future.**

***Index Terms*—vulnerability, threat modeling, knowledge graph, link prediction**

I. Introduction

Threat databases provide useful common knowledge for enhancing system security. Prominent databases include the Common Platform Enumeration (CPE) [1], Common Vulnerabilities and Exposures (CVE) [2], and Common Weakness Enumeration (CWE) [3]. 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 CVE lists publicly disclosed cybersecurity vulnerabilities, while the CWE lists software and hardware weakness types. A weakness can lead to a specific instance of a vulnerability within a product or system. In addition to weaknesses, the CWE list also includes *views* and *categories* [4]: 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. These publicly available threat databases are extensively used for security assessment. In particular, several threat modeling tools (e.g., OWASP pytm, Threagile, IriusRisk) use entries of these databases as pre-defined threats in their libraries [5]. By checking whether the conditions of the pre-defined threats are met, potential threats in a system can be identified, and corresponding mitigations can be developed.

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 which may allow a remote attacker to occupy a thread that consumes maximum CPU time and will never return. ...
Associated CPE by Aug 4, 2021	1) cpe:a:xstream_project:xstream:*:*, 2) cpe:o:debian:debian_linux:*:*
Associated CPE after Aug 4, 2021	1) cpe:o:fedoraproject:fedora:*:* (1.000), 2) cpe:a:oracle:retail_xstore_point_of_service:*:* (0.998), 3) cpe:a:oracle:webcenter_portal:*:* (0.995), 4) cpe:a:oracle:banking_platform:*:* (0.993), 5) cpe:a:oracle:communications_policy_management:*:* (0.992), 6) cpe:a:oracle:communications_billing_and_revenue_management_elastic_charging_engine:*:* (0.973), 7) cpe:a:oracle:mysql_server:*:* (0.966), 8) cpe:a:oracle:business_activity_monitoring:*:* (0.961), 9) cpe:a:oracle:communications_unified_inventory_management:*:* (0.903), 10) cpe:a:oracle:banking_virtual_account_management:*:* (0.841)

TABLE I: Example of prediction results for CVE-2021-21348. 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. The predicted probabilities of existence are shown after the names of CPE entries. Using a prediction threshold $\alpha = 0.96$, 8 out of 10 products are successfully predicted (marked in blue), with one false positive prediction (`cpe:a:netapp:clustered data ontap antivirus connector:*:*`).

The National Vulnerability Database (NVD) [6] provides associations between entries in the different databases. Table I shows an example of how the entries in different threat databases are associated [7]. 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.

Associations provided by the NVD are incomplete, however. In Table I, we split affected products of CVE-2021-21348 into two groups: those revealed before and after August 4, 2021.

This work was supported in part by Honda Research Institute and BU Hariri Institute Research Incubation Award (2020-06-006) and by the US National Science Foundation under grants CNS-1717858, CNS-1908087, CCF-2006628, and EECS-2128517.

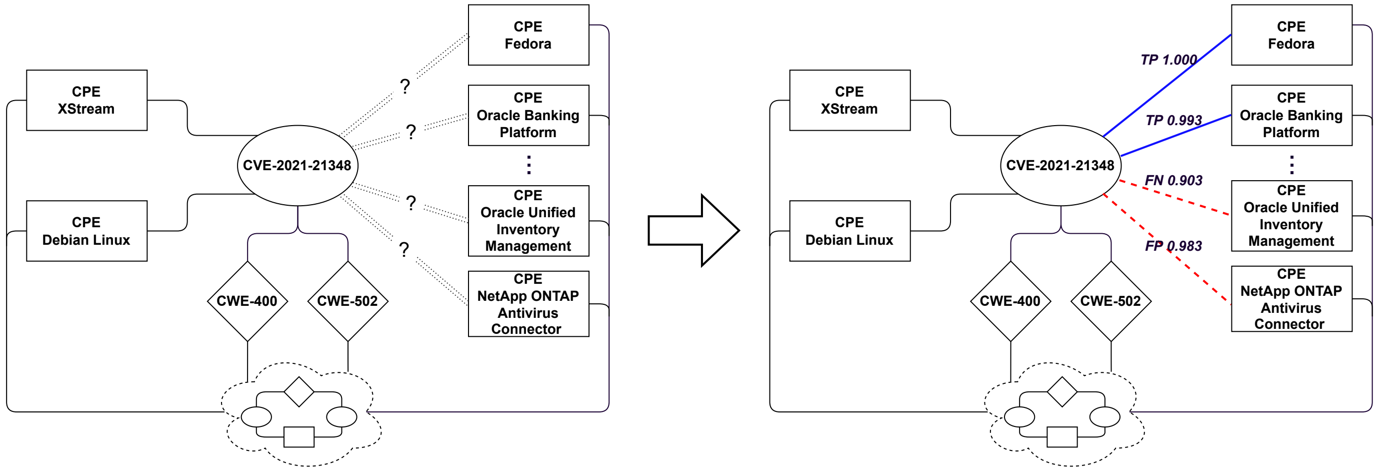
CPE XStream
CPE Debian Linux
CVE-2021-21348
CWE-400
CWE-502
CPE Fedora
CPE Oracle Banking Platform
CPE Oracle Unified Inventory Management
CPE NetApp ONTAP Antivirus Connector
?
?
?
?
TP 1.000
TP 0.993
FN 0.903
FP 0.983

Fig. 1: Illustrations of the prediction process. For any given CVE (e.g., CVE-2021-21348), and its known associations to CPEs (left side) and CWEs at a certain date, we aim to predict future associations to other CPEs (right side). The cloud at the bottom represents the rest of CPE/CVE/CWE entries and the associations between them. The blue lines represent successfully predicted associations (true positives), and the red dashed lines represent failed predictions due to false positives or false negatives.

By August 4, 2021, only two CPE entries were known to be affected. However, many more entries were subsequently discovered. Being able to predict some of these product vulnerabilities in advance would be highly valuable.

In this paper, 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 [8]–[10] out of existing databases (CPE, CVE, and CWE), which we refer to as a *threat knowledge graph*. We use the threat knowledge graph to predict future associations between threat database entries, as illustrated in Figure 1. More generally, threat knowledge graphs can suggest weaknesses based on information related to products, improve existing threat modeling tools in terms of threat identification [11], and help gain a better understanding of the characteristics and trends of common threats.

Our work 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 embed the knowledge graph onto a vector space that can be used for link prediction.
    - We optimize the knowledge graph, to improve its prediction capabilities.
2) We evaluate the ability of threat knowledge graphs to predict associations between products and vulnerabilities. Specifically, we evaluate the ability of a threat knowledge graph generated with data available on August 4, 2021 to predict associations between CPE and CVE entries that were discovered between August 4, 2021 and March 29, 2022. The evaluation shows that our approach achieves an F1-score of 0.681 with a precision of 0.775 and a recall of 0.606, which indicates useful prediction capabilities.

II. Related work

There exists a body of work leveraging threat databases to provide further insight about threats. The work in [12] applies data mining on the CVE and CWE databases to produce interesting characteristics about software vulnerabilities, such as statistics of essential vulnerabilities. The work in [13] uses historical data in threat databases for predicting the time till 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.

One problem when leveraging threat databases is that they are not always consistent and ready for use [13]. Efforts have been made to automate the association of entries in different threat databases. The work in [14] proposes an automated way of extracting related CPE entries from a CVE description, to associate CPE and CVE entries. This is achieved by applying a natural language process technique called named entity recognition (NER) on CVE descriptions. On the CVE to CWE side, since vulnerabilities can be seen as instances of weaknesses in specific products or systems, automated association between CVE and CWE can be cast as a classification problem. The work in [15] first vectorizes CVE descriptions, then classifies the vectorized descriptions into about 19 CWE classes by using the random forest algorithm based on selected features.

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

Knowledge graph are widely used for reasoning over interrelated data for tasks such as recommendation [16] and question answering [17]. The work in [9] 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. The work in [18] 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 [19] 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 the associations within a threat database or between different databases. The work in [19] does not consider the CPE, however, though it is valuable for identifying threats based on products in the system. More significantly, evaluation of prediction is performed by splitting the knowledge graph into training and test sets under a closed-world assumption. Hence, it is unclear if the prediction model at a certain time actually works for future data. In our work, we 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.

III. 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 next detail and justify optimizations performed on the knowledge graph. Finally, we show how to embed the knowledge graph onto a vector space, which allows one to predict associations between threat databases.

A. 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 Fig. 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 in the following. The arrows in Fig. 2 point from the head entity to the tail entity.

There exist three main types of entities: 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, which provide a higher level view of weaknesses. Each type of entries mentioned are added to the knowledge graph with a unique name (by directly using the name of entries or making slight adjustments).

In addition to the names of the CPE and CWE entries, we include key attributes in the knowledge graph, also 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 Fig. 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 (e.g., CPE-x is related to CVE-y, and CVE-y is related to CWE-z), as provided by the NVD; (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).

B. Optimizing the knowledge graph

The purpose of the threat knowledge graph is to extract relevant information from the associations between the threat databases. Therefore, instead of using all the entries from the databases, we make two key optimizations: (i) we merge CPE entries that have the same attributes, except for their version numbers; (ii) we remove CPE and CVE entries that contain no association information (e.g., all 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.

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

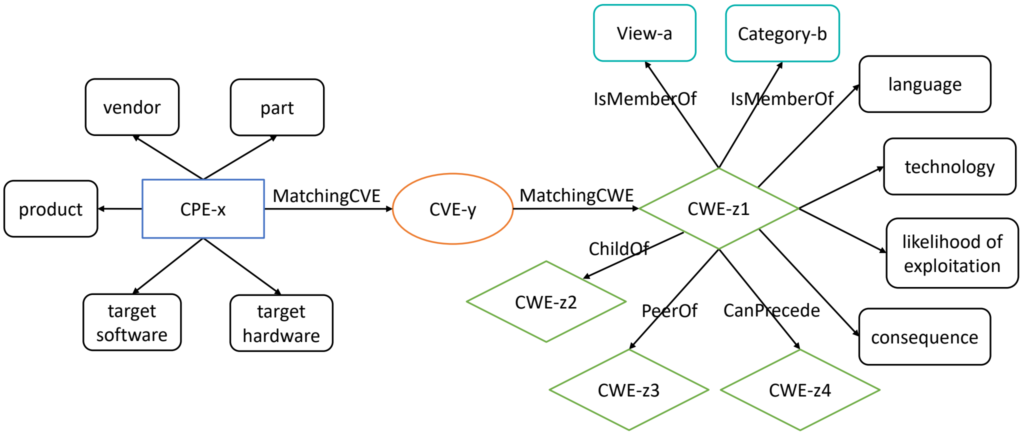

Fig. 2: Complete structure of the threat knowledge graph.

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. Yet, different versions of a product are typically similar in terms of attributes and related vulnerabilities. If we merge CPE entries that are identical except for their version numbers, 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 reduced without losing much valuable information.

We further find that a small portion of application-related CPE entries tend to have many more version numbers than other entries. Without merging, those entries would have a large weight on the knowledge graph, making it difficult to extract information from other valuable entries. Merging CPE entries results in a more balanced knowledge graph.

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. However, we find that many entries are not connected to other databases. Specifically, only 33,427 CPEs are connected to CVEs, 149,582 CVEs are connected to CPEs, 106,861 CVEs are connected CWEs, and 288 CWEs 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. 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, we opt not to include unconnected CPE and CVE entries, to increase the portion of valuable entries in the knowledge graph. Our evaluation in Section III-C validates the effectiveness of this optimization.

C. Threat knowledge graph embedding

In this subsection, 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. The embedded knowledge graph can then be used for various applications, including link prediction. In the following, we train the embedding model, and evaluate the embedded knowledge graph through standard metrics.

We first explain the training process. Multiple models are available for embedding. These models mainly differ by their *scoring functions* [20]. The 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 real world) and low scores for negative triples. ComplEx is one of the state-of-the-art embedding models based on semantic matching, which uses a complex space for embedding and handles well asymmetric relations [21]. Through comparison with other scoring functions, we found that ComplEx performs well for the task at hand, and therefore selected ComplEx as the embedding model.

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. Typically, 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. A generated negative triple is discarded if it belongs to the knowledge graph (a step called filtering).

We next discuss the evaluation of the knowledge graph. 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 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 triple and the negative triples according to their scores (higher scores have lower rank). Positive triples in the test set to should be ranked favorably compared to the negative triples.

For quantitative evaluation of the embedding, the following metrics are commonly used (the mathematical definitions

	MRR	MR	Hits@10	Hits@3	Hits@1
Before optimization	0.221	27503	0.303	0.240	0.176
After optimization	0.248	21113	0.345	0.267	0.198

TABLE II: Evaluation results of embedding, before and after removing unconnected CPE and CVE entities.

follow below) [21]–[23]: (1) the mean rank (MR), which computes the average of ranks of the positive triples in the test set; (2) the mean reciprocal rank (MRR), which computes the reciprocal mean of ranks of the positive triples in the test set; 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 the set of all triples in the knowledge graph. For a good embedding, we expect low MR, high MRR, and high Hits@N scores.

Table II summarizes the evaluation results. Removing unconnected CPE and CVE entities indeed improves all the metrics. After optimizing the knowledge graph, the MRR is 0.248. The Hits@3 score of 0.267 implies that the model ranks positive triples within the top-3 positions 26.7% of the time. These metrics are sufficient for good prediction, as also validated by our experiments in the next section. Our knowledge graph has many entities with relatively sparse connections, making it hard to get metrics as good as those from benchmarking [24]. Fine-tuning training parameters of the embedding model may help further improve these metrics.

IV. Prediction using threat knowledge graph

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

A. Prediction results and metrics

We use entries from the CPE, CVE, and CWE databases available on August 4, 2021 to predict product vulnerabilities that appeared from August 4, 2021 until March 29, 2022. We note that our results should be viewed as lower bounds on performance, since some of the hidden vulnerabilities could still be revealed after this period.

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.

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 a product has a specific vulnerability that is currently hidden, but that we predict will be revealed in the future.

We next discuss the prediction of associations between CPE and CVE entries. From August 4, 2021 to March 29, 2022, there were 41,583 newly added CPE-CVE associations. We find that out of the newly added associations, 4,157 of them are between entities that already existed in the August 2021 version of the knowledge graph. Specifically, the 4,157 triples involve 920 unique CPE entries and 1257 unique CVE entries in 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 4,157 new triples as positive triples in the test set. We then generate negative triples by replacing the CPE side of positive triples with other random CPE entries that appear in the knowledge graph. For each unique CVE entry in the positive triples, we generate 50 negative triples. As a result, we get 67,007 triples in total for testing prediction, where 4,157 triples are positive, and 62,850 triples are negative.

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 are correct. A high recall implies that a high fraction of the positive triples are successfully identified. It is well-known that there exists a trade-off between the precision and recall metrics, resulting in a precision-recall curve. In order 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.

The evaluation metrics change with α since only triples with estimated probability above α are predicted to be positive. For example, a high threshold α results in a higher precision but lower recall. We depict the precision-recall curve in Fig. 3, which reflects the trade-off between precision and recall as the threshold changes. When $\alpha = 0.981$, the F1-score 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 77% of the predicted associations are correct, while 60% of unknown associations are retrieved. 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

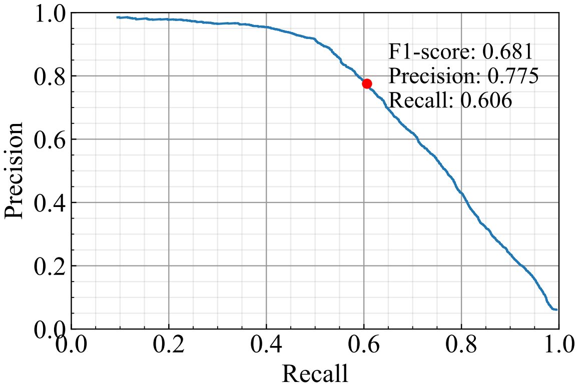

Fig. 3: Precision-recall curve of predicting CPE-CVE associations using threat knowledge graph. The curve reflects the trade-off between precision and recall as the threshold α for positive prediction changes. When α decreases, the precision decreases and the recall increases. A reference point based on the maximized F1-score is marked in red.

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 Aug 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 Aug 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:*:*, ...

TABLE III: Example of prediction results for CVE-2021-0144. 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.

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. On the other hand, 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 which will be further verified manually.

B. Examples of prediction

We next illustrate the prediction results through a few examples. The first example is CVE-2021-0144 [25] detailed in Table III. CVE-2021-0144 is a vulnerability related to Intel processors. We 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. In this example, all the names of CPE entries associated to CVE-2021-0144 are similar. As a result, predictions are mainly based on the naming patterns and therefore relatively simple.

A second, more complex example is CVE-2021-21348, detailed in Table I 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 afterwards. 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. In general, 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 each entry separately.

V. Conclusion

In this work, we introduced a methodology for predicting future associations between products and vulnerabilities, using threat knowledge graphs. We explained how to produce and optimize such threat knowledge graphs, and how to embed them into a vector space. We validated our approach with historical data and demonstrated good precision, recall, and F1-score performance on a set that includes 50 random negative triples for each positive triple. Note that changing the number of negative triples may impact the precision (and thereby the F1-score), but not the recall. We provided examples illustrating how threat knowledge graphs extract implicit relations from threat databases to facilitate prediction.

Predicted associations between entries of different threat databases can be utilized to enhance security in various ways. For example, if a specific product is used in a system, a threat modeler can list both existing and predicted vulnerabilities associated with the product. Our prediction methodology can also help prioritizing 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 include additional threat databases in the threat knowledge graph, such as the Common Attack Pattern Enumeration and Classification (CAPEC) database [26]. CAPEC enumerates known attack patterns employed to exploit weaknesses from the adversary's perspective. The extent to which this addition could improve prediction performance represents an interesting open problem.

References

[1] "Official common platform enumeration (CPE) dictionary," 2022. [Online]. Available: https://nvd.nist.gov/products/cpe

[2] "Common vulnerabilities and exposure (CVE)," 2022. [Online]. Available: https://cve.mitre.org/

[3] "Common weakness enumeration (CWE)," 2022. [Online]. Available: https://cwe.mitre.org/

[4] "CWE glossary," 2022. [Online]. Available: https://cwe.mitre.org/documents/glossary/index.html

[5] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, "Threat modeling tools: A taxonomy," *IEEE Security & Privacy*, vol. 20, no. 04, pp. 29–39, 2022.

[6] "National vulnerability database," 2022. [Online]. Available: https://nvd.nist.gov/general

[7] "CVE-2021-21348 detail," 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-21348

[8] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[9] X. Chen, S. Jia, and Y. Xiang, "A review: Knowledge reasoning over knowledge graph," *Expert Systems with Applications*, vol. 141, p. 112948, 2020.

[10] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[11] L. Sion, D. Van Landuyt, K. Yskout, S. Verreydt, and W. Joosen, "Automated threat analysis and management in a continuous integration pipeline," in *2021 IEEE Secure Development Conference (SecDev)*. IEEE, 2021, pp. 30–37.

[12] X. Li, J. Chen, Z. Lin, L. Zhang, Z. Wang, M. Zhou, and W. Xie, "A mining approach to obtain the software vulnerability characteristics," in *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*. IEEE, 2017, pp. 296–301.

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

[14] E. Wåreus and M. Hell, "Automated CPE labeling of CVE summaries with machine learning," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2020, pp. 3–22.

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

[16] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 950–958.

[17] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 105–113.

[18] Y. Wang, Y. Zhou, X. Zou, Q. Miao, and W. Wang, "The analysis method of security vulnerability based on the knowledge graph," in *2020 the 10th International Conference on Communication and Network Security*, 2020, pp. 135–145.

[19] L. Yuan, Y. Bai, Z. Xing, S. Chen, X. Li, and Z. Deng, "Predicting entity relations across different security databases by using graph attention network," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2021, pp. 834–843.

[20] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[21] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International conference on machine learning*. PMLR, 2016, pp. 2071–2080.

[22] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[23] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[24] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li, "Realistic re-evaluation of knowledge graph completion methods: An experimental study," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1995–2010.

[25] "CVE-2021-0144 detail," 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-0144

[26] "Common attack pattern enumerations and classifications (CAPEC)," 2022. [Online]. Available: https://capec.mitre.org/