Inferring Cyber Threat Intelligence - A Knowledge Graph-based Approach*

Nidhi Rastogi,1,2 Sharmishtha Dutta,1 Alex Gittens,1 Mohammed J. Zaki,1 and Charu Aggarwal,3

1 Rensselaer Polytechnic Institute, Troy NY 12083, USA
2 Rochester Institute of Technology, Rochester, NY
3 IBM Research, Yorktown Heights, NY

nidhi.rastogi@rit.edu, {duttas,gittea}@rpi.edu, zaki@cs.rpi.edu, charu@us.ibm.com

Abstract. Security analysts prepare threat analysis upon investigating an attack, an emerging cyber threat, or a recently discovered vulnerability. Threat intelligence on malware attacks and campaigns is shared on blog posts, reports, analyses, and tweets with varying technical details. Other security analysts use this intelligence to inform them of emerging threats, indicators of compromise, attack methods, and preventative measures. Collectively known as threat intelligence, it is typically in an unstructured format and, therefore, challenging to integrate seamlessly into existing IDPS systems. In this paper, we propose a framework that aggregates and combines CTI - the openly available cyber threat intelligence information. The information is extracted and stored in a structured format using knowledge graphs such that the semantics of the threat intelligence can be preserved and shared at scale with other security analysts. We propose the first semi-supervised open-source knowledge graph (KG) framework, TINKER, to capture cyber threat information and its context. Following TINKER, we generate a Cyber Threat Intelligence Knowledge Graph (CTI-KG). We demonstrate the efficacy of CTI-KG using different use cases and its application for security analysts.

Keywords: Threat Intelligence · Knowledge Graphs · Information Retrieval

1 Introduction

Security analysts investigate cyber threats posed by attacks on organizations and determine remedial actions to prevent damage regularly. They compile reports that identify ways to pinpoint and prepare for potential threats (see figure 1). This information, called Cyber Threat Intelligence (CTI), offers a detailed summary of one or more cyber attacks and aggregates relevant knowledge and context. It covers information on zero-day attacks, newly identified vulnerabilities,

* Supported by IBM AI Research Collaboration.
indicators of compromise (IoC), threats, and attack patterns. Such information is multi-modal, predominantly unstructured and does not follow a specific format. Other cyber threat analysts utilize the shared threat intelligence (free or paid) to mitigate potential threats and defend their organization. Existing unstructured and structured sources of threat information, such as analysis reports from security organizations, blogs, common vulnerabilities and exposure databases (CVE), provide curated, sometimes very detailed accounts of the current cyber threat landscape. Security companies such as McAfee and Symantec, researchers, government agencies, and experts publish these accounts largely in unstructured text. MITRE, NIST, OASIS Open have spearheaded initiatives that make security information available in structured formats through standards like common vulnerabilities and exposures (CVE) [22] and national vulnerability database (NVD) [17], and structured threat information eXpression (STIX) [4].

![CTI report](image)

**Fig. 1.** An excerpt of a CTI report on the malware family *Backdoor.Winnti*.

**Threat Intelligence Knowledge Extractor** (TINKER) addresses these two main gaps: (a) whereas the current taxonomy and standards provide extensive coverage of threat concepts, they lack semantic and contextual associations, and (b) threat intelligence extraction models not trained on cyber threat corpus can lead to capturing fragmented and sometimes fallacious threat information.

TINKER transforms multi-modal CTI into a structured format while preserving the context of the information. TINKER uses ontologies and information extraction models to capture CTI. It seamlessly integrates heterogeneous sources of data, captures data and its context, and structures them using the Cyberthreat Intelligence Knowledge Graphs (CTI-KG), which enables further machine learning like analysis [1]. This paper focuses on extracting threat intelligence from the generic security corpus. In this paper, we make the following contributions:

1. We present TINKER, the first end-to-end framework for capturing and analyzing multi-modal threat intelligence information using a semi-supervised approach.
2. Using TINKER, we generate an open-source CTI-KG, the first open-access knowledge graphs for general cyber threat intelligence.
3. We demonstrate the use of CTI-KG through two use cases to infer previously unknown threat information such as vulnerabilities, associations between malware, and malware author.
4. We present an open-access corpus of over 52K triples comprising 30k unique named entities and 22 relationships. This is a new benchmark framework for contextual cyber security research problems.

**Knowledge Graphs (KGs) for threat intelligence:** Cyber-attacks have evolved dramatically over the past decade, and their attack patterns have become highly sophisticated and complex. An attack pattern defines a sequence of tactics, techniques, and procedures the attacker follows to conclude the attack. Machine learning has successfully analyzed cyber attacks using supervised, unsupervised, and deep learning using host and network log data. While successful, they miss out on attack vectors that use social engineering, insider threats, and similar, non-traceable vectors. Security analysts want to investigate newly discovered attack vectors (attacker, malware, vulnerability) and determine the level of threat to their organization at the earliest. Internal telemetries gathered from log data are not mapped to newly emerging threat indicators. Therefore, it is essential to aggregate, study and map external threat indicators and patterns to internal enterprise threat detection tools. KGs have been used extensively to organize and integrate data from heterogeneous sources in the biomedical domain. The proposed CTI-KG holds threat information from the CTI dataset and stores it as a triple in RDF format with two entities connected by one relationship. Entities represent classes such as malware, organizations, vulnerability, attack patterns, malware behavioral characteristics, and attack timeline. Relationships connect entities and provide context for their interpretation.

**Motivating Example:** Here, we describe a real-world malware attack analysis to motivate threat intelligence using knowledge graphs.

Curated threat intelligence information adds semantic analysis to existing evidence and the provenance of the source of information, adding evidence against the attacker (see Figure 2. For instance, CTI written in early 2021, cyber attacks on the SolarWinds software captured detailed analysis on the Sunspot malware that inserted the Sunburst malware into SolarWinds’ Orion software. In such events, with only sparse data and preliminary evidence about the threat actor, CTI-KG infers attack actors and other attributes for Sunburst with a confidence score. For instance, it infers that Sunburst and a backdoor linked to the Turla advanced persistent threat (APT) group share similarities.

**Main Challenges:** Some of the major challenges encountered in generating the knowledge-based framework for threat intelligence are:

(a) **Missing cybersecurity specific information extraction models** - CTI reports contain both generic and domain-specific threat data. Named entity recognizer (NER) models such as StanfordNER[^5] trained on generic datasets or UMLS for specific domains such as health cannot be used. For example, using existing domain-specific concepts sometimes have titles that can mislead NER models into categorizing them as generic concepts. For example, when a vulnerability description mentions “Yosemite backup 8.7,” a generic

[^4]: https://www.cyberscoop.com/turla-solarwinds-kaspersky/
[^5]: https://nlp.stanford.edu/software/CRF-NER.shtml
NER does not refer to Yosemite as a server backup software but as a national park in California, US.

(b) **Missing context in extracting key phrases** - Existing cybersecurity standards and taxonomies (like STIX, TAXII) overlook the importance of context in extracted concepts (our research addressed this via linking and semantic metadata). Text data is not a mere aggregation of words. There is a meaning in the information, and linking concepts to outside sources is crucial to its effective use. For example, a human security analyst can correctly comprehend the meaning of the term “Aurora” in a CTI report. It refers to the series of cyber attacks conducted by Advanced Persistent Threats (APTs) by China, first detected by Google in 2009. However, a computer or an AI model collecting this information might mistake it for the Northern Lights unless provided with a security context.

(c) **Lack of standard format of cyber threat corpus** - Malware CTI reports cover technical and social aspects of a malware attack in the form of unstructured text. Generating a knowledge graph requires extracting key phrases from these reports, which is not a trivial task. The varying degrees of attack and defense description and the changing landscape of the information can render prior information defunct, which is an additional challenge. Unlike a small curated benchmark dataset, we require information extraction systems that are scalable and replicable for a large amount of text corpus specific to the cyber threat domain.

(d) **Missing validation techniques** - Knowledge graphs for cybersecurity threat intelligence is a new approach for providing structured and semantically rich information to people and machines. However, since this is an area of research, there is no publicly available “ground truth” to evaluate the knowledge framework.

2 Definitions

**Cyberthreat Intelligence Knowledge Graph (CTI-KG)** is a directed multimodal graph, $\text{CTI-KG} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}$ indicate the sets of entities, relationships and triples, respectively. Each triple $(e_{\text{head}}, r, e_{\text{tail}}) \in \mathcal{T}$ indicates
that there is a relationship $r \in \mathbb{R}$ between $e_{\text{head}} \in \mathcal{E}$ and $e_{\text{tail}} \in \mathcal{E}$. Knowledge graph entity prediction is the task of predicting the best candidate for a missing entity. However, in the CTI domain, prediction is replaced with the term inference since we are inferring an attack vector based on the knowledge present in the CTI-KG. Formally, the task is to infer the value for $h$ given $(?, r, t)$ or $t$ given $(h, r, ?)$, where “?” indicates a missing entity. A summary of the notation appears in Table 1.

### Table 1. Notations and Descriptions

| Notation | Description |
|----------|-------------|
| $\mathcal{E}, \mathcal{R}, \mathcal{T}$ | set of all entities, relationships and triples respectively |
| $(e_{\text{head}}, r, e_{\text{tail}})$ | head entity, relationship and tail entity |
| $(h, r, t)$ | embedding of $h$, $r$ and $t$ |
| $d_e, d_r$ | embedding dimension of entities, relationships |
| $n_e, n_r, n_t$ | number of $h+t$, $r$, triples |
| $f_\phi$ | scoring function of each triple $(h, r, t)$, measures plausibility of a fact in $\mathcal{T}$, based on translational distance or semantic similarity [12] |

Ontological mapping is performed using classes and relationships defined in cyberthreat ontologies [18][20] to extract triples from unstructured text. An ontology maps disparate data from multiple sources into a shared structure that maintains a logic in that structure using rules and follows a common vocabulary [15]. For example, there are classes named Malware and Location in MAL-Ont. According to the relationships defined in an ontology, an entity of Malware class can have a relationship “similarTo” with another Malware class entity. However, according to the rule engine, an entity belonging to the class Malware can never be related to an entity belonging to the class Location. The same two entities can have multiple relationships between them. For example, an entity of type Malware and Application can have relationships such as uses, communicates, with have very different meaning. Therefore, context plays a significant role in analyzing entities.

### 3 TINKER Framework

We describe TINKER, the framework for generating CTI-KG and inferring threat intelligence. (see Figure 4). The framework comprises modules that perform semi-supervised data collection using information extraction models, CTI-KG construction, generating vector embeddings from triples, and inference. **Automated data collection:** We aggregate hundreds of reports on threat intelligence uploaded on GitHub repositories. These reports are written between the years 2010-2022 by security analysts from top security companies such as FireEye, Symantec, McAfee, Microsoft Research, security bulletins, TechCrunch, and ESET. See Figure 1 for an excerpt of a CTI report on the Backdoor.Winnti malware family. Not all reports are consistent, accurate, or comprehensive and can
vary in technical depth and breadth of the attack description. Table 2 shows a set of triples generated from this snippet of an CTI.

**Information Extraction:** This module is responsible for extracting contextual and structured cyber threat information. Named entity extraction (NER) and the semantic relationship between them (relation extraction, RE) from unstructured text form triples – the fundamental unit of CTI-KG. We first create ground truth using a combination of hand-annotated CTI, supervised and semi-supervised NER, and relationship extraction. For instance, consider the following snippet from a CTI report:

“... DUSTMAN can be considered as a new variant of ZeroCleare malware ... both DUSTMAN and ZeroCleare utilized a skeleton of the modified “Turla Driver Loader (TDL)” ... The malware executable file “dustman.exe” is not the actual wiper. However, it contains all the needed resources and drops three other files [assistant.sys, elrawdisk.sys, agent.exe] upon execution...”.

**Table 2.** Triples corresponding to the CTI report.

| Head Entity       | Relationship | Tail Entity       |
|-------------------|--------------|-------------------|
| DUSTMAN           | similarTo    | ZeroCleare        |
| (DUSTMAN,         | involves     | Turla Driver Loader (TDL)) |
| (ZeroCleare,      | involves     | Turla Driver Loader (TDL)) |
| (DUSTMAN,         | involves     | dustman.exe       |
| (dustman.exe,     | drops        | assistant.sys     |
| (dustman.exe,     | drops        | elrawdisk.sys     |
| (dustman.exe,     | drops        | agent.exe         |

**Fig. 3.** Sample CKG. Nodes and edges represent entities and relations, respectively. The edge labels denote relationship type.

**CTI-KG Construction:** Together, thousands of triples construct CTI-KG where the entities and relationships model nodes and directed edges, respectively. Figure 3 shows the sample knowledge graph from the text. A knowledge graph contains 1-to-1, 1-to-n, n-to-1, and n-to-n relationships between entities. In Figure 3 the entity “dustman.exe” is an instance of a Malware File. It drops

---

6 [https://www.lastwatchdog.com/wp/wp-content/uploads/Saudi-Arabia-Dustman-report.pdf](https://www.lastwatchdog.com/wp/wp-content/uploads/Saudi-Arabia-Dustman-report.pdf)
Inferring Threat Intelligence: This task infers threat intelligence about an emerging or existing security threat along with a confidence score (0.0-1.0). We discuss various embedding models that we explored for the inference task in related work section [6]. For this paper, we adapt the TuckER [3] model, a linear model based on Tucker decomposition of the binary tensor representation of CTI-KG triples. TuckER creates vector embeddings for all triples in CTI-KG and infers missing entities for classes defined in the ontology [18].

Evaluation Criteria: Knowledge Graph-based threat intelligence is a new paradigm for inferring information about emerging threats. Unlike machine learning-based models that predict a value or label a data point based on prior learning from training data, knowledge graph-based prediction provides a list of entities inferred by a triple, using the vector embedding approach and the confidence score. Fundamentally, inferring threat intelligence entities is modeled in CTI-KG by predicting the missing entities in the graph. The training step of the CTI-KG constructs the KG. Further, in the inference step, we perform the inference operations to conclude new facts not explicitly present in CTI-KG.

For example, consider “Adobe Flash Player” as an entity belonging to the class Software in CTI-KG. Several entities have an incomplete triple with a missing tail entity in the training phase because that information was missing from the threat reports. This triple will be of the form ⟨Adobe Flash Player, hasVulnerability, ?⟩. The entity inference operation will infer a list of missing tail entities and the confidence score of the predictions, i.e., a vulnerability of the Software titled “Adobe Flash Player.” In another example, for an incomplete triple, where the tail entity is the malware family “Hydraq”, the CTI-KG returns the class of the head entity, Campaign, and the entity itself, Trojan.Hydraq for the query ⟨?, involvesMalware, Hydraq⟩. In CTI-KG, entity inference is evaluated and confidence score is computed as described below [23].
1. When is an incomplete triple, \( \langle h, r, t \rangle \) needs to be completed by an entity, a set of candidate entities are generated along with the scoring function, \( f_\phi(h,r,t) \), for each triple. The score is sorted to obtain the rank of the true test triple in the ordered set. This rank evaluates the performance of the model in predicting \( \text{tail} \) given \( \langle \text{head}, \text{relation}, ? \rangle \). A higher rank indicates a model's better efficiency in learning embeddings for the entities and relations.

2. \( \text{Hits}_@n \) is calculated from the ranks of all true test triples, where \( \text{Hits}_@n \) denotes the ratio of true test triples in the top \( n \) positions of the ranking. When comparing embedding models for prediction, smaller values of \( \text{Hits}_@n \) indicate a better model.

### 4 Dataset Preparation

The first dataset comprises hand-annotated triples, and the second is from a larger CTI corpus from various cyber threats such as APT, malware, and phishing. CTI report is originally in .pdf or .html format and then converted into text format. Pre-processing includes several techniques to minimize noise and remove unnecessary information from the text. We tokenize and parse the plain text sentences into words with start and end positions using a python library called spaCy\(^7\) which allows POS tagging, dependency parsing, and word vectors. Each token is assigned an id and stored for each CTI report.

**Ground Truth:** 85 reports were hand-annotated by mapping entities and relationships to classes defined by ontology. Two graduate and five undergraduate student researchers with cybersecurity education performed this task in 3 to 4 months. A senior, more experienced full-time researcher oversaw the task and broke ties when needed. Hand annotated triples were later verified by the same expert. The CTI reports were written between 2010-2022. In Figure 6, “PowerPoint file” and “installs malicious code” are labeled as classes **Software** and **Vulnerability** respectively. The arrow from **Software** to **Vulnerability** denotes the semantic relationship **hasVulnerability** between them. The annotation followed rules defined in cyber threat ontologies \[18, 20\]. 11 object properties (\( n_r \)) are defined in cyber threat intelligence ontology \[18\]. The comparison between the structure of the knowledge graphs generated from the hand-annotated ground truth and CS40K dataset, their properties, and corpus details are described in Table 3. The columns in the table compare the number of entities, relations, and triples, the average degree of entities, and graph density of our datasets (CS3K, CS40K) and benchmark datasets. The average degree across all relationships and graph density are defined as \( n_t/n_e \) and \( n_t/n_e^2 \), respectively \[10\]. A lower score of these two metrics indicates a sparser graph.

**Larger Dataset:** A total of 1,100 threat advisory reports were downloaded from GitHub repositories. They were written by analysts from major security organizations’ websites such as Symantec, Kaspersky, Microsoft, IBM, FireEye, TrendMicro, and McAfee. The annotated key phrases were classified into entities derived from semantic categories defined in cyber threat ontologies \[18, 20\].

\(^{7}\) [https://spacy.io/api/tokenizer](https://spacy.io/api/tokenizer)
Table 3. Description of Datasets

| Dataset | $n_c$  | $n_r$ | $n_t$ | docs  | avgDeg | density   |
|---------|--------|-------|-------|-------|--------|-----------|
| CS3K    | 5,741  | 22    | 3,027 | 81    | 0.5273 | 0.00009   |
| CS40K   | 27,354 | 34    | 40,000| 1,100 | 1.46   | $5.34 \times 10^{-5}$ |

The inference rule engine from the ontologies was applied when forming relationships between entities to produce triples used to build the knowledge graphs. We use semi-supervised learning to train NER and relationship extraction models by providing a few hand-annotated samples. The CTI-KG comprises 40,000 triples generated from 27,354 unique entities and 34 relations.

Information Extraction and Ontological Mapping: Concepts that a generic named entity recognition model can easily classify require training using a representative dataset. We train several NER models with suggestive keywords and the context within which the keyword occurs. Both approaches are critical in providing entity mapping for classes representing facts and contextual data. We train an odd number of NER models for each class, and the entity extraction (EE) step combines the results of the ensemble of various extraction techniques. A shortcoming of using multiple EE models is that some entities can get classified multiple times, which might even be inaccurate in some cases. We observed empirically that this phenomenon was not limited to any specific class. This phenomenon explains the variation and grammatical inconsistency in different CTI reports and their writing style. For example, the entity “Linux” was classified as Software and Organization for the same position in a document. Since the entity extraction models are running independently, it was likely for close class types to have overlapped. We resolve this issue by following these rules

1. We compared the confidence score of entities occurring at the same position in the same document and between classes.
2. For the same confidence scores, we marked those classes as ambiguous. We retained these instances until the relationship extraction phase. At that time, we relied on the rule engine from the ontology to determine the correct class of the head and tail entity for a given relationship (described in the next section).

The entity classification (EC) step addresses the disambiguation of an entity classified into more than one semantic category and assigns only one class based on an empirically learned algorithm. Together, EE and EC maximize the automation of an otherwise exhaustive entity recognition process. Semantic text patterns from NER and RE map to classes (called entity in KG) and object properties (called relationships in KG) defined by an ontology. For example, in [18], three classes largely describe malware behavior – Malware, Vulnerability, and Indicators of Compromise. See Figure 1 for a snapshot on an instance of ontology and Figure 5 for a snippet of the top-level classes of the ontology. We segregate the extraction of tokenized words into two categories: domain-specific key-phrase extraction and generic key-phrase extraction (class: Location). The cybersecurity
domain-specific EE task was further subdivided into factual key-phrase extraction (class: Malware) and contextual key-phrase extraction (class: Vulnerability). We use precision and recall measures as evaluation criteria to narrow down the EE models for different classes. For example, the Flair library \[2\] gave high precision scores (0.88-0.98) for the classification of generic and factual domain-specific key-phrases such as - Person, Date, Geo-Political Entity, Product (Software, Hardware). The diversity in writing style for the CTI reports and the noise introduced during reports’ conversion to text led to different precision scores for classes but within the range of 0.9-0.99. For contextual entity extraction, we expanded the SetExpan \[19\] model to generate all entities in the text corpus. The feedback loop that inputs seed values of entities based on confidence score allowed for improving the training dataset. Some of the semantic classes used in this approach are Malware, Malware Campaign, Attacker Organization, Vulnerability. We perform relationship extraction for CTI-KG using a distantly supervised artificial neural network model \[26\] pre-trained on the Wiki data and Wikipedia, initially with three features. The CS3K triples and text corpus form the training dataset and the entities generated in the previous steps, and the processed text corpus the testing dataset. We built on this learning and improved automatic relation extraction (RE) by using semi-supervised relation extraction methods\[8\] aimed to leverage unlabeled data in addition to learning from limited samples.

![Main Classes (left), Object Properties (right) from a cyber threat ontology. Instances of classes, called entities and instances of object properties called relationships are mapped to CTI text using NER and relation extraction models](image1)

**Fig. 5.** Main Classes (left), Object Properties (right) from a cyber threat ontology. Instances of classes, called entities and instances of object properties called relationships are mapped to CTI text using NER and relation extraction models.

![Annotation using Brat annotation tool.](image2)

**Fig. 6.** Annotation using Brat annotation tool.

\[8\] https://github.com/THU-BPM/MetaSRE
5 Experiment and Evaluation

We infer threat intelligence on the CTI-KG through a tensor factorization-based approach \[3\] using PyTorch. To train all the datasets, we choose all hyperparameters through random search based on validation set performance. Since all the datasets contain a significantly smaller number of relationships relative to the number of entities, we set entity and relationship embedding dimensionality to \(d_e = 200\) and \(d_r = 30\), learning rate = 0.0005, batch size of 128 and 500 iterations. We use batch normalization and dropout to decrease training time. For evaluation, for each test triple, we generate \(n_e\) candidate triples by combining the test entity-relation pair with all possible entities \(E\), followed by ranking the computed scores. We use evaluation metrics standard across the link prediction literature, and they are described in Section \(??\).

CS3K validation set is used for tuning the hyper-parameters for CS40K. For the prediction experiment on CS3K, we achieved the best performance with \(d_e = 200\) and \(d_r = 30\), learning rate = 0.0005, batch size of 128 and 500 iterations. We split each dataset into 70% train, 15% validation, and 15% test data. Due to the limited hand-annotated dataset, we tested the CS40K dataset using CS3K. The challenge with a smaller ground truth can lead to overfitting of results. To increase the size of the ground truth, we increase the size of triples through a semi-supervised method, i.e., by the process described earlier. A human-in-the-loop (a security expert) is employed to fix the false positives manually.

5.1 Inferring Threat Intelligence from CTI-KG

Table \[7\] displays the results of entity prediction existing embedding models on the benchmark datasets. An upward arrow (↑) next to an evaluation metric indicates that a larger value is preferred for that metric and vice versa. For ease of interpretation, we present \(\text{Hits@n}\) measures as a percentage value instead of a ratio between 0 and 1. TransE has the largest \(\text{Hits@10}\), whereas TransR performs better on the rest of the three datasets. However, TransH is more expressive (capable of modeling more relations) than TransE despite having the simplicity of TransE (time complexity \(\mathcal{O}(d_e)\)). The space complexity of TransR (\(\mathcal{O}(n_e d_e + n_r d_r)\)) is more expensive than TransH (\(\mathcal{O}(n_e d_e + n_r d_e)\))\[23\]. Additionally, on average, it takes six times more time to train TransR\[^9\] compared to TransH\[^3\]. Considering these model capabilities and complexity trade-offs, we choose TransH as a representational translation-based model. DistMult performs better than TuckER by a small margin of 0.37\% in only one case (\(\text{Hits@10}\) on WN18). TuckER and ComplEx both achieve very close results on multiple cases.

Therefore, we look closely at the results of the recently refined datasets (FB15K-237 and WN18RR). The CS3K dataset shares similar properties with these two datasets (CS3K has hierarchical relationships as in WN18RR and no redundant triples as in FB15K-237). Since the space complexity is very close

\[^9\] On benchmark datasets
Table 4. Experimental evaluation of CS3K for entity prediction.

| Model   | Hits@1↑ | Hits@3↑ | Hits@10↑ | MR↓  | MRR↑ |
|---------|---------|---------|----------|------|------|
| TransH  | 26.8    | 50.5    | 65.2     | 32.34| 0.414|
| TuckER  | 64.3    | 70.5    | 81.21    | 7.6  | 0.697|

Table 5. Experimental evaluation of CS40K for entity prediction.

| Model   | Hits@1↑ | Hits@3↑ | Hits@10↑ | MR↓  | MRR↑ |
|---------|---------|---------|----------|------|------|
| TuckER  | 73.9    | 75.9    | 80.4     | 202  | 0.75 |

(ComplEx: O(n_e d_e + n_r d_r) and TuckER: O(n_e d_e + n_r d_r)) we choose TuckER for evaluation on CS3K, as it subsumes ComplEx [2].

We evaluate the performance of TransH [24] and TuckER [3] on our datasets. Tables 4 and 5 display the performance of TransH and TuckER on CS3K and CS40K. TransH’s MRR score on CS3K improved when compared to the benchmarks. Tucker’s performance improved in all measures when compared to the other benchmark results. TuckER outperforms TransH by a large margin on the CS3K dataset.

Since TuckER performed significantly better than TransH on the ground truth dataset; we employ only TuckER on the CS40K dataset. It is noteworthy that all the measures except mean rank (MR) improved on CS40K compared to CS3K. It confirms the general intuition that the model learns the embeddings more effectively as we provide more data. MR is not a stable measure and fluctuates even when a single entity is ranked poorly. We see that the MRR score improved even though MR worsened, which confirms the assumption that only a few poor predictions drastically affected the MR on CS40K. Hits@10 being 80.4 indicates that when the entity prediction model predicted an ordered set of entities, the true entity appeared in its top 10 for 80.4% times. Hits@1 measure denotes that 73.9% times, the model ranked the true entity at position 1, i.e., for 80.4% cases, the model’s best prediction was the corresponding incomplete triple’s missing entity. We can interpret Hits@3 similarly. We translate the overall entity prediction result on CS40K to be sufficient such that the trained model can be improved and further used to infer unseen facts.

5.2 Use-case Description

We query CTI-KG by forming an incomplete test triple as this is the formal approach to inferring information from a knowledge graph. The results are a set of predictions where each prediction has a confidence score associated with it. The confidence score of a predicted entity $e_i$ denotes the probability of $e_i$ being the accurate candidate for the incomplete triple. The predictions are sorted in descending order by this score, and we observe the top 10 predictions as potential predictions (Hits@10).
1. **Case Study 1, Predicting a malware family:** A security analyst wants to identify which malware family is associated with the indicator `intel-update[.]com` in a CTI-KG. The following query is posed to the CTI-KG in the form of an incomplete triple:

$$\langle \text{intel-update[.]com, indicates, ?} \rangle$$

Here, the accurate malware family associated with the domain `intel-update[.]com` is named Stealer. We refer the reader to the FireEye report titled ‘Operation Saffron Rose’\(^{10}\) for validating this fact. This CTI was not included in the training corpus. However, the training set had some triples involving Stealer, such as:

$$\langle \text{office.windowsessentials[.]tk, indicates, Stealer} \rangle$$
$$\langle \text{Saffron_Rose, involvesMalware, Stealer} \rangle$$

The model learns latent features of Stealer, its indicators of compromise, and other related entities (e.g., campaigns that involved Stealer) present in the training set. Table 6 shows the ordered set of predictions returned by the model, and the true answer is ranked #1. This is a significant result as it can simplify the job of an analyst to a great extent by narrowing down the scope for further analysis. Starting with this set of 10 results with different confidence scores, the analyst finds the accurate information at the second item. By definition, the \textit{Hits@3} and \textit{Hits@10} metrics increase as the true entity exist in the top 3 and certainly in the top 10. As seen in table 5, we can say 75.9% times the true entity will appear in the top 3 predictions of the model.

2. **Case study 2, Inferring Attack Target:** Note that CTI-KG is built by extracting semantically rich information on a variety of attacks. We, therefore, constructed CTI-KG exclusively for android malware. We expected better inferences as the corpus was specific to one category of attacks. A security analyst queries CTI-KG to learn about the region impacted by a type of attack. The following query is posed to the CTI-KG in the form of an incomplete triple:

$$\langle \text{“coronavirus-themed attacks”, targets, ?} \rangle$$

The training set had some triples on “coronavirus-themed attacks”. Note that the training set contains facts from different CTI reports, and the terms “coronavirus-themed attacks” and ‘targets’ refer to the same attack category. The model learns latent features of these entities and relationships during training and later uses these features to infer a set of regions highly probable to involve the attack.

Table 6 demonstrates the inferences made by this query. We can see that the

\(\text{https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-operation-saffron-rose.pdf}\)
Table 6. Detailed result of Inferring Entities

| Rank | Inferred Entity | Confidence score | Case study 2: Inferring Attack Target |
|------|-----------------|------------------|--------------------------------------|
|      |                 |                  |                                      |
|      | Stealer         | 0.5769           | India                                |
| 2    | A malware hash12 | 0.5694           | recorded future                      |
| 3    | A malware hash13 | 0.5679           | issuemakerslab                       |
| 4    | Gholee          | 0.5679           | google                               |
| 5    | 200.63.46.33    | 0.5668           | China                                |
| 6    | 26978_ns2.aeroconf2014.org | 0.5662 | maas                                |
| 7    | A malware hash14 | 0.5620           | United Arab Emirates                 |
| 8    | Capstone Turbine | 0.5602           | Kaspersky                            |
| 9    | 2012/06/06      | 0.5592           | italy                                |
| 10   | Google Hack Attack | 0.5566          | portugal                             |

Bold text denotes the true entity for the corresponding test triple

Table 7. Inferring Entity for different Data triple

| Model  | FB15K Hits@10† | MRR† | WN18 Hits@10† | MRR† | FB15K-237 Hits@10† | MRR† | WN18-RR Hits@10† | MRR† |
|--------|----------------|------|---------------|------|-------------------|------|----------------|------|
| TransE | 44.3           | 0.227| 75.4          | 0.395| 32.2              | 0.169| 47.2           | 0.176|
| TransH | 45.5           | 0.177| 76.2          | 0.434| 30.9              | 0.157| 46.9           | 0.178|
| TransR | 48.8           | 0.236| 77.8          | 0.441| 31.4              | 0.164| 48.1           | 0.184|
| DistMult| 45.1           | 0.206| 80.9          | 0.531| 30.3              | 0.151| 46.2           | 0.264|
| ComplEx| 43.8           | 0.205| 81.5          | 0.597| 29.9              | 0.158| 46.7           | 0.276|
| TuckER | 51.3           | 0.260| 80.6          | 0.576| 35.4              | 0.197| 46.8           | 0.272|

The top three rows are models from the translation-based approach, rest are from tensor factorization-based approach

The true answer (India) is ranked #1. In order to validate this finding, we refer the reader to the CTI report

This result contributes to increasing the Hits@10 metric (since the rank is within the top 10), as well as to Hits@1 and Hits@3. We obtain an intuition about what the Hits@n metric on the overall test dataset represents. Hits@10 score reported in table 5 can be interpreted that 80.4% time the true entity would appear in the top 10 inferences. We elaborate on this analysis in the following section.

6 Related Work

Knowledge graphs store large amount of domain specific information in the form of triples using entities, and relationships between them. General purpose knowledge graphs such as Freebase, Google’s Knowledge Graph and

11 shorturl.at/mnBFU
15 https://developers.google.com/knowledge-graph
WordNet have emerged in the past decade. They enhance “big-data” results with semantically structured information that is interpretable by computers, a property deemed indispensable to build more intelligent machines. No open-source malware knowledge graph exists, and therefore, in this research, we propose the first of its kind automatically generated knowledge graph for cyber threat intelligence. Creating and completing a knowledge graph from raw text comprises multiple tasks, e.g., entity discovery, relationship extraction, and designing embedding models. In this section, we review notable research works to accomplish these tasks. Entity discovery refers to categorizing entities in a text according to a predefined set of semantic categories.

Named Entity Recognition (NER) is a machine learning task for entity discovery. The machine learning model learns features of entities from an annotated training corpus and aims to identify and classify all named entities in a given text. Deep Neural Network-based NER approaches identify complex latent features of text corpora without any human intervention for feature design. Long Short Term Memory (LSTM), a recurrent neural network architecture, in NER can learn beyond the limited context window of a component and can learn long dependencies among words. Convolutional Neural Network (CNN) is a class of neural networks that learns character-level features like prefixes, and suffixes when applied to the task of NER. Chiu et al. combine LSTM and CNN’s strengths to learn word-level and character-level features. Lample et al. use a layer of conditional random field over LSTM to recognize entities with multiple words. MGNER uses a self-attention mechanism to determine how much context information of a component should be used, and is capable of capturing nested entities within a text.

Vector embeddings address the task of knowledge representation by enabling computing on a knowledge graph. CTI-KG has highly structured and contextual data and can be leveraged statistically for classification, clustering, and ranking. However, the implementations of embedding vectors for the various models are very diverse. They can be broadly categorized into three categories: translation-based, neural-network-based, or trilinear-product-based. We discuss trilinear-product based due to space restrictions.

Tensor based approaches: Tensor factorization-based approaches consider a knowledge graph as a tensor of order three, where the head and tail entities form the first two modes and relationships form the third mode. For each observed triple \( (e_i, r_k, e_j) \), the tensor entry \( X_{ijk} \) is set to 1; and otherwise 0. The tensor is factorized by parameterizing the entities and relationships as low dimensional vectors. For each entry \( X_{ijk} = 1 \), the corresponding triple is reconstructed so that the value returned by the scoring function on the reconstructed triple is very close to \( X_{ijk} \), i.e., \( X_{ijk} \approx f_{\phi}( (e_i, r_k, e_j) ) \). The model learns the entity and relationship representations.
relationship embeddings with the objective to minimize the triple reconstruction error:

\[ L = \sum_{i,j,k} (X_{ijk} - f_\phi(e_i, r_k, e_j))^2 \]

where \( f_\phi \) is a triple scoring function.

7 Conclusion

This paper proposes a framework to generate the first cyber threat intelligence knowledge graph, CTI-KG, to infer threat intelligence from unstructured CTI reports. The framework outputs a CTI-KG, which comprises 27,354 unique entities, 34 relationships and approximately 40,000 triples in the form of entity\text{head}-relation-entity\text{tail}. Since this is a relatively new line of research, we demonstrate how a knowledge graph can accurately infer threat information from the text corpus through the entity inference application. We motivate and empirically explore different models for encoding triples and concluded that the tensor factorization-based approach resulted in the highest overall performance among our datasets.

Using CS40K as the baseline dataset, we will explore robust entity and relation extraction models for future work that can generalize to the cybersecurity domain for disparate datasets, including blogs, tweets, and multimedia. Another application uses vector embeddings to identify possible cybersecurity attacks from the background knowledge encoded in CTI-KG. The current prediction model can generate a list of top options for missing information. We plan to extend this capability to infer threat intelligence about newly emerging attack vectors and use that information to defend against attacks that have yet to propagate widely. Inferring correctly depends on the choice of embedding models; we ensured that the embedding model meets the following requirements:

1. An embedding model has to be expressive enough to handle different kinds of relationships (as opposed to general graphs where all edges are equivalent). It should create separate embeddings for entities that can be involved in the same relation. To elaborate, for two triples- \( \langle e_1, \text{hasVulnerability}, e_3 \rangle \) and \( \langle e_2, \text{hasVulnerability}, e_3 \rangle \), the embedding of an entity \( e_1 \in \text{class:ExploitTarget} \) is different than that of an entity \( e_2 \in \text{class:Software} \).
2. An embedding model should also be able to create different embeddings for an entity when it is involved in different kinds of relationships [24]. This requirement ensures creating embedding of triples with 1-to-n, n-to-1, and n-to-n relations. For example, a straightforward and expressive model TransE [6] has flaws in dealing with 1-to-n, n-to-1, and n-to-n relationships [24, 13].
3. Entities that belong to the same class should be projected in the same neighborhood in the embedding space [10]. This requirement allows using an embedding’s neighborhood information when predicting the class label of an unlabeled entity (this task is called entity classification).
References

1. Knowledge graphs, https://www.turing.ac.uk/research/interest-groups/knowledge-graphs
2. Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., Vollgraf, R.: Flair: An easy-to-use framework for state-of-the-art nlp. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations). pp. 54–59 (2019)
3. Balazević, I., Allen, C., Hospedales, T.M.: Tucker: Tensor factorization for knowledge graph completion. arXiv preprint arXiv:1901.09590 (2019)
4. Barnum, S.: Standardizing cyber threat intelligence information with the structured threat information expression (stix). MITRE Corporation 11, 1–22 (2012)
5. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250 (2008)
6. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
7. Chiu, J.P., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics 4, 357–370 (2016)
8. Christian, R., Dutta, S., Park, Y., Rastogi, N.: Ontology-driven knowledge graph for android malware. arXiv preprint arXiv:2109.01544 (2021)
9. Fellbaum, C.: Wordnet. In: Theory and applications of ontology: computer applications, pp. 231–243. Springer (2010)
10. Guo, S., Wang, Q., Wang, B., Wang, L., Guo, L.: Semantically smooth knowledge graph embedding. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 84–94 (2015)
11. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)
12. Li, J., Sun, A., Han, J., Li, C.: A survey on deep learning for named entity recognition. IEEE Transactions on Knowledge and Data Engineering (2020)
13. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 29 (2015)
14. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE 104(1), 11–33 (2015)
15. Oltramari, A., Cranor, L.F., Walls, R.J., McDaniel, P.D.: Building an ontology of cyber security. In: Semantic Technology for Intelligence, Defense and Security. pp. 54–61. Citeseer (2014)
16. Padia, A., Kalpakis, K., Ferraro, F., Finin, T.: Knowledge graph fact prediction via knowledge-enriched tensor factorization. Journal of Web Semantics 59, 100497 (2019)
17. Radack, S., Kuhn, R.: Managing security: The security content automation protocol. IT professional 13(1), 9–11 (2011)
18. Rastogi, N., Dutta, S., Zaki, M.J., Gittens, A., Aggarwal, C.: Malont: An ontology for malware threat intelligence. In: International Workshop on Deployable Machine Learning for Security Defense. pp. 28–44. Springer (2020)
19. Shen, J., Wu, Z., Lei, D., Shang, J., Ren, X., Han, J.: Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 288–304. Springer (2017)
20. Swimmer, M.: Towards an ontology of malware classes. Online 27 (2008)
21. Tran, H.N., Takasu, A.: Analyzing knowledge graph embedding methods from a multi-embedding interaction perspective. arXiv preprint arXiv:1903.11406 (2019)
22. Vulnerabilities, C.: Common vulnerabilities and exposures (2005)
23. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering 29(12), 2724–2743 (2017)
24. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. vol. 14, pp. 1112–1119. Citeseer (2014)
25. Xia, C., Zhang, C., Yang, T., Li, Y., Du, N., Wu, X., Fan, W., Ma, F., Philip, S.Y.: Multi-grained named entity recognition. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 1430–1440 (2019)
26. Yao, Y., Ye, D., Li, P., Han, X., Lin, Y., Liu, Z., Liu, Z., Huang, L., Zhou, J., Sun, M.: Docred: A large-scale document-level relation extraction dataset. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 764–777 (2019)