Cybersecurity Knowledge Graph Improvement with Graph Neural Networks

Soham Dasgupta*, Aritran Piplai†, Priyanka Ranade†, Anupam Joshi†
* Mallya Aditi International School
Email: sohamdasgupta91@gmail.com
†Dept. of Computer Science & Electrical Engineering, University of Maryland, Baltimore County,
Email: {apiplai1, priyankaranade, joshi}@umbc.edu

***Abstract*—Cybersecurity Knowledge Graphs (CKGs) help in aggregating information about cyber-events. CKGs combined with reasoning and querying systems such as SPARQL enable security researchers to look up information about past cyber-events that is helpful in understanding future cyber-events or drawing similarity with a known cyber-event recorded in a CKG. CKGs have assertions in the form of semantic triples. The triples describe a relationship between a subject and object, both of which are cybersecurity entities. The quality of information present in the CKG depends on the data source. Since data sources can have varying degrees of reliability, we need a score that should help us benchmark the veracity of the CKG assertions. Verifying the information asserted in the CKG is a challenging task. In this paper, we describe a novel method that associates a score with the semantic triples asserted in the CKG using deep learning. We use semantic triples that we know are correct, in a supervised machine learning algorithm that produces the output for each relationship. In particular, we use Graph Convolutional Neural Networks (GCN) on a dataset of CKGs that can be used to ascertain the scores for each semantic triple.**

***Index Terms*—Graph Embedding, Deep Learning, Cybersecurity, Artificial Intelligence, Representation Learning**

I. Introduction

Cyber Threat Intelligence (CTI) helps security researchers prevent, detect, and post facto analyze cyber-attacks. Recently, Cybersecurity Knowledge Graphs (CKGs) have been used to represent CTI [1]–[3]. CKGs can contain data from a variety of sources, both structured and unstructured. Structured sources are often maintained by reputed organizations that maintain and update information present in their systems [4]–[6]. However, there is another body of research that utilizes natural language processing techniques to extract data from unstructured open-source text specifically for CTI representation. In this body of work, the source of the data is open-source text that can come from security bulletins of software organizations, technical reports from cybersecurity organizations, social media posts, tweets, and sometimes even the dark web. It becomes difficult to ascertain the quality of information present in these disparate sources. When data is extracted from these sources, semantic triples are formed that are asserted to a CKG [7]. The quality of the graph is dependant on the quality of the NLP pipeline that is doing the extraction. Even state of the art systems are nowhere near perfect in extracting entities or relationships in complex domains like cybersecurity, For example, in previous work our group has done, the accuracy of relationship extraction ranged from 80%s for well behaved text sources to as low as 50% for some relationships in text collected from OSINT sources [8]. So the extracted graph will contain incorrect relationships. If the source of the data unfortunately turns out to be dubious, the CKG will also contain information that is incorrect. Security researchers, who use CKGs to query information relating to recorded cyber-incidents, will receive answers that are incorrect [3]. Identifying such incorrect assertions in a CKG is the focus of this work. Very recent research has shown that “fake but plausible” threat intelligence can be created by suitably trained large language models [1]. Addressing this is outside the scope of this paper, since the assertions in such graphs are quite possibly correct in general, but are not right for *that particular* instance of data.

Asserting extracted CTI information from open-source text comes with another risk. If the schema of the CKG does not have appropriate classes to record the time when the data was asserted, we can be left with outdated information that are no longer useful. A recent report suggests that as much as 90% security officers are presented with outdated CTI that is too old to be of any use [9]. For example, if a report from 2015 says updating Adobe Acrobat Reader in your system will mitigate a certain type of malware attack the information may not pass the test of time. In 2021, we may need some other form of patch on the same software to build adequate defence against malware attacks. If security officers are relying on old information, it would be appropriate to reduce the confidence of the conclusion they reach from that information. However, it becomes difficult to do the same if the time-related information is missing from the CKG. In our paper, we discuss a GCN model to process a CKG that has no time-specific information to reach a confidence score for the semantic triples that form the CKG.

In our paper, we discuss a Graph Neural Network-based model that will help us generate scores for the specific relationships that exist between pairs of entities in a CKG. We use a supervised approach, where we use known triples that are correct, as the ‘labels’. The relationships for all the entities that do not belong in the ‘label’ are also updated simultaneously. Each relationship in the graph has a score, that gets updated after receiving the supervision from the ‘label’. We want to use this updated score to filter out the relationships that are possibly incorrect due to being outdated, or simply because they were from a data source that had incorrect information. This will help us preserve the relationships that are still correct, and discard the relationships that are not.

We organize the paper as follows. In Section II, we discuss some of the relevant papers in this area. In Section III, we will discuss our model's architecture and specifications. We will discuss the results in Section IV, and finally conclude in V.

II. Related Work

In this section we discuss some of the works published that are related to our problem. Here we provide some background information along with the relevant papers for some of the key concepts of our paper.

A. CyberSecurity Knowledge Graphs

Recently there has been a significant body of research that focuses on representing CTI in CKGs. A semantic triple is a building block of a CKG. Zareen et al. [2], in their paper, discussed an ontology UCO that represents CTI. This ontology is based on Structured Threat Intelligence Exchange (STIX) [6], as the classes and relationship mentioned in STIX have been included in UCO. Piplai et al. [7] modified this ontology and used it to represent CTI from After Action Reports. This paper also discusses a pipeline that extracts malware specific semantic triples from unstructured text, using named entity recognizers and relationship extractors. This ontology has been adapted in subsequent research for different purposes. For instance, this CKG has been modified [10] to represent malware behavior and the enriched CKG has been applied in a Reinforcement Learning algorithm [11]. There has been more research on processing open-source unstructured text for cybersecurity related information [12], [13]. To extract cybersecurity entities, NER models specific to cybersecurity have been built. Dasgupta et al. [14] concluded that a combination of BERT embeddings with Bidirectional LSTMs is the best for Cyber NERs. Recently, attention based methods are being popularized for cyber entitity extraction [15]. This forms the foundation of the semantic triples that are asserted in the CKG. Often machine learning algorithms are used to establish a relationship between pairs of entities. TransE, TransH, TransD [16] models are used for relationship extraction. In these models vector embeddings are used to infer the relationship between pairs of entities. Pingle et al. [8] developed a neural network based relationship extractor that works as a classifier to establish relationship between pairs of entities. Recently, more ontologies are being created for CTI representation [17].

B. Data Poisoning and Fake CTI

Malicious alteration of Machine Learning datasets, can lead to false conclusions drawn by the models using these datasets. Some of the works related to data poisoning [18]–[21] talk about the use changing the training data so that the machine learning models perform poorly on the actual test data. In the domain of computer vision, there has been considerable research on 'backdoor attacks'. Some specific triggers on images in the training set poison the model, such that adversaries are able to direct the model to undesired outputs [22].

CKGs, that are formed from unstructured open-source text, are not immune to data poisoning attacks. Extracting data from sources like social media posts, can prove to be risky if they contain misinformation about cybersecurity. Data poisoning can lead to machine learning models misclassify malicious code as benign data. External indicators have proved to be useful to validate the model that is susceptible data poisoning [23]. Khurana et al. [24] discuss a model that uses reddit features to build a model that identifies poisoned samples in the training set. Xiao et al. [25] discuss a feature extraction algorithm that has a defensive mechanism against poisoning attacks. Recently, Transformer-based architectures have proved to be beneficial in generating fake text that is realistic enough to be confused as real text by domain experts. Particularly, Ranade et al. [1] used GPT-2 to generate fake CTI for cybersecurity. These CTI, when processed and asserted in a CKG, will contain incorrect information.

C. Graph Neural Networks

Graph Neural Networks (or Graph Convolutional Neural Networks, GCNs) generalize convolutional neural networks typically meant for image and audio data to data represented in non-Euclidean domains, such as graphs, for robust feature learning. Spectral GCN define an approximation of convolutional neural networks for graphs using spectral graph theory. Defferrard et al. [26] define a formulation of CNNs using the Chebyshev expansion of the graph Laplacian. Bruna et al. [27] show that for low-dimensional graphs it is possible to learn convolutional layers with a number of parameters independent of the input size, resulting in efficient deep architectures. Kipf et al. [28] propose a convolutional architecture via a first-order approximation.

On the other hand, non-spectral GCNs operate on the graph directly and apply convolution or weighted average to the neighboring nodes, as shown by Duvenaud et al. [29] and Hamilton et al. [30]. Recent developments have increased the capabilities and expressive power of Graph Neural Networks. There are many practical applications in areas such as antibacterial discovery fake news detection [31], physics simulations [32] etc.

Researchers have also deployed GNNs in recommender system. PinSage [33] applies GNNs to the pin-board bipartite graph in Pinterest and Pixie [34] uses the Pixie Random Walk algorithm for Pinterest Object graph to generate recommendations. Monti et al. [35] and Berg et al. [36] model recommender systems as matrix completion and design GNNs for representation learning on user-item bipartite graphs. Wu et al. [37] use GNNs on user/item structure graphs to learn user/item representations. Wang et al. [38] design GNNs for heterogeneous KGs with regularization to avoid overfitting.

TABLE I: Modified UCO Classes and Relationships

Type	List
Classes	Software, Exploit-Target, Malware, Indicator, Vulnerability, Course-of-Action, Tool, Attack-Pattern, Campaign, Filename, Hash, IP Addresses
Relationships	attributedTo, indicates, hasProduct, mitigates, hasVulnerability, uses

III. Methodology

In this section, we discuss our neural model and the knowledge graphs that we used. In our previous research we discussed a pipeline to generate CKGs from malware After Action Reports. We reuse the same mechanism to generate our CKGs for this paper. We then use the generated CKGs for the GCN model that produces scores for each relationship existing between entity pairs. These scores help us identify the credibility of each relationship that can exist between every pair of entities. We now discuss each component in detail.

A. CKG

Piplai et al. [7] described a pipeline to extract entities and relationships for cybersecurity. The CKG schema was based on UCO 2.0 [2]. Most of the entity-classes and relationship-classes used in STIX [6] was used to form UCO 2.0, since STIX is an industry standard for exchanging threat-intelligence.

In Table I, we can see the classes and relationships used to represent malware-specific information from After Action Reports. This can be used to represent the same information from any technical report about malware that comprises unstructured text.

A customized Named Entity Recognizer (NER) called the Malware Entity Extractor (MEE) was used to extract cyber-entities. The MEE comprised of Conditional Random Fields (CRF) and Regular expressions. CRF was used to extract entities that require contextual information to make a classification, and Regular Expressions were used to extract entities that can be inferred from the structural composition, like IP-Addresses, Filenames, Hashes, etc.

Word embeddings for pairs of extracted entities were passed to neural network-based relationship extractor, that establishes the correct relationship (if any) existing between the pair. This results in a semantic triple that are asserted in the CKGs. 474 After Action Reports were used to create the training set for the models (NER and Relationship Extractor).

We take a subset of the CKGs that were created by the aforementioned pipeline. We take 80% of the CKGs for training.

B. Model

The problem statement of our paper is to find out scores between 0 and 1 to represent each relationship in a CKG. The higher the score the more credible or trustworthy the relationship is.

Let us consider that we have a set of entities E and a set of relationships R. Each semantic triple in the CKG is represented as $K = \{E_h, r, E_t\}$, where E_h represents the head-entity, E_t represents the tail-entity, and r represents the relationship between them, such that $r \in R$.

We represent each input CKG in the form of a matrix K, where

$K = \{0,1\}^{|E|\times|E|\times|R|}$

Here $|E|$ denotes the number of entities in K and $|R|$ denotes the number of relationships. It is a symmetric matrix, where each row denotes if the particular relationship exists or not for the entity pair. For example if $K(i,j,k) = 1$, it means R_k exists between E_i and E_j. If it is 0, then the relationship R_k does not exist between entities E_i and E_j. After successful training, $K(i,:,:)$ can be interpreted as an embedding vector representing E_i that tells us how much credible all the relationships are with respect to all the other entities in the CKG, for the entity E_i. For example if a *Malware* 'Solarwinds hack' *uses* an *Attack-Pattern* 'clicks an icon', we have to index these entities 'Solarwinds hack', 'clicks an icon', and the relationship 'uses'. Supposedly, the indexes come out to be E_3, E_7, and R_2, respectively, then $K(3,7,2)$ becomes 1.

The input to the system is a CKG that is represented by K consisting of partially correct information. We use a CKG that holds the correct entities and relationships in place that we denote by K^*. The GCN is a function approximator, f , that produces another CKG K' as output such that

$K' = [0,1]^{|E|\times|E|\times|R|}$ and

$K' = f(K)$

In this case, $K'(i,j,k)$ is a probabilistic value between 0 and 1 that tells us how important the relationship R_k is between entity-pairs E_i and E_j.

We use K^* as the target. As discussed before, each $K^*(i,:,:)$ gives us the ideal representation of entity E_i. The neural network f produces an output K' and $K'(i,:,:)$ represents the predicted representation of the same entity E_i. We define a loss function L such that it calculates the dissimilarity between the predicted representations of all the entities and the target representations of all the entities. The cumulative loss function is defined as follows.

$Loss = \sum_{i=1}^{|E|} L(K^*(i,:,:), K'(i,:,:))$

The construction of K^* does not require an entire CKG that has been rectified by human annotators. The purpose of our research is to update the probabilistic values that signify the 'trust' scores of the relationships of our CKG. In order to make our algorithm scale well, we cannot place a requirement

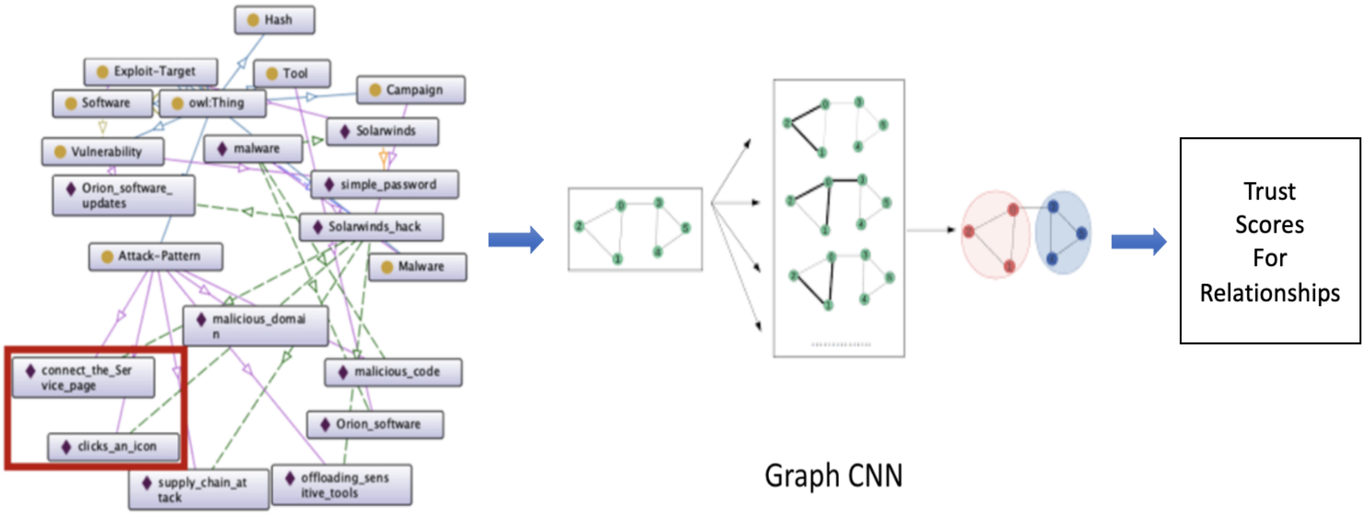


Fig. 1: High level architecture diagram for our GCN model. The input to the GCN is a CKG with incorrect information highlighted in red. The GCN produces an output that contains probabilistic scores for all the relationships in the CKG. The scores can be interpreted as how much we 'trust' each relationship.

on a completely correct CKG for training. If we do not have a complete CKG to be used as label, and we have partial triples that talk about the same malware we generate a CKG on our own in the following way.

For supervision, let us consider we have a set of triples T_p. In the original graph we have a set of triples T_s. Each triple $t \in T_p$ can be represented by P_h, P_r, P_t corresponding to the head-entity, relationship, and the tail-entity. Similarly, each triple $t \in T_s$ can be represented by S_h, S_r, S_t. We iterate through all the triples in T_p, and check if the head of any triple in T_p overlaps with any of the tail nodes of the triples in T_s. If that node is $T_p(i)_h$, we remove all triples from T_s, where $T_p(i)_h$ is a head node. After that we add all triples from T_p to T_s, that have $T_p(i)_h$ as a head node. We remove them from T_p. We repeat this for all the remaining nodes in T_p untill it is empty. We discard nodes that have no overlap with $|E|$ which is the set of entities in K. The key motivation for doing so is that T_p can come from a source that talks about some specific details of a cyber-attack. We want to update the knowledge graph for those specific details of the cyber-attack, but we also want to preserve the semantic triples that talk about the other aspects of the cyber-attack that have been collected from other CTI sources that is represented by T_s. For example, consider a malware 'Dark Caracal' that uses another malware 'xRAT'. The entire CKG that has information related to 'Dark Caracal' and 'xRAT' is T_s. Suppose, a new report comes that has updated information about 'xRAT'. We want to create a new CKG that retains the information about 'Dark Caracal', but updates the information about 'xRAT'. A possible critique of this method could be removal of all knowledge about 'xRAT' from T_s that is not covered in T_p. It should be noted here that a missing triple would not mean that after training, the scores of the relationships belonging to the removed triples will be low. The GCN model changes the scores based on subgraphs and it is a self-correcting system. With more CTI being available about 'xRAT' it will be possible to change the scores of the relationships from the discarded triples, if they happen to be correct. We can see the algorithm in Algorithm 1.

Algorithm 1 Algorithm to create K^* from partial sub-graphs

```
$N_s \leftarrow |T_s|$
$N_p \leftarrow |T_p|$
while $N_p \neq 0$ do
    $P_h, P_r, P_t \leftarrow T_p(N_p)$
    while $N_s \neq 0$ do
        $S_h, S_r, S_t \leftarrow T_s(N_s)$
        $C \leftarrow \emptyset$
        $A \leftarrow \emptyset$
        if $P_h$ equals $S_t$ then
            $C \leftarrow T_s(i) \quad \forall i \in N_s \;\; s.t. \;\; T_s(i)_h = P_h$
            $A \leftarrow T_p(i) \quad \forall i \in N_p \;\; s.t. \;\; T_p(i)_h = P_h$
        end if
        $T_s \leftarrow T_s - C + A$
        $T_p \leftarrow T_p - A$
        $N_p \leftarrow |T_p| - 1$
    end while
    $N_s \leftarrow |T_s| - 1$
end while
return $T_s$
```

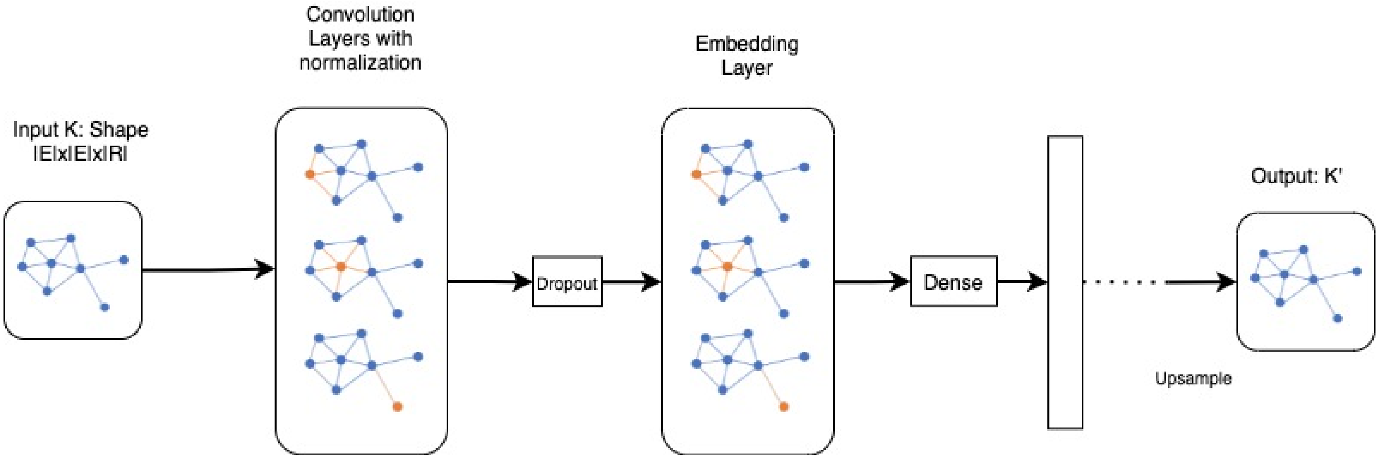

Convolution Layers with normalization
Embedding Layer
Input K: Shape |E|x|E|x|R|
Dropout
Dense
Output: K'
Upsample

Fig. 2: Neural Network architecture for GCN

The GCN used in our paper is based on the principle multi-label classification. Further, our network incorporates a fixed threshold for the output scores - a probabilistic measure between 0 and 1.0. Our multi-label graph convolutional neural network gives as an output, an array of independent probabilities of a relationship existing between the given two entities, with one or more relations possibly being true. More details of the GCN architecture can be seen in Figure 2.

Let us consider a knowledge graph comprising triplets of the format (h,l,t). When the 2 entities 'h' and 't' of a knowledge graph triplet (h,l,t) is passed into our GCN, an array of independent probabilities is given as an output for the value of the relationship 'l'.

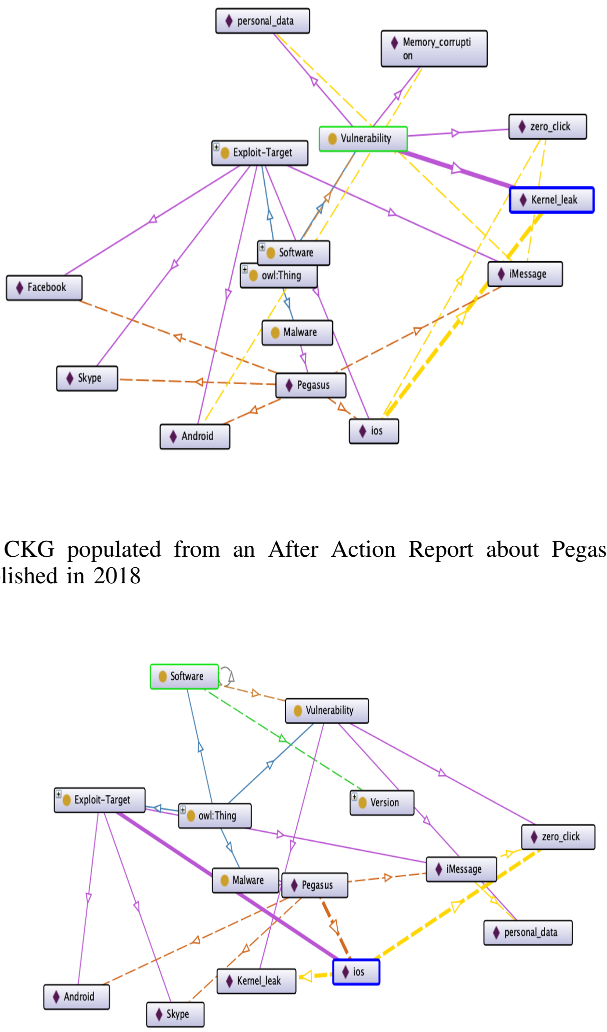
The GCN model utilises 2 essential parameters - the relation vocabulary and entity vocabulary, which are the number of relationships and entities in our knowledge graph (KG) respectively. We define an embedding as a transformation of the knowledge graph entities to their vector representations. Further, we define the length of the transformed vector as the embedding dimension. There are two optional parameters that define our model, the first one states is the embedding dimensions of our entities and the second is a hyperparameter used to determine the dimensions of the second and third dense layers of our network architecture.

Our network architecture comprises of 8 hidden layers - the embedding layer, the concatenate layer, 3 dense layers and 3 dropout layers. The first two dense layers use a Rectified Linear Unit (ReLU) activation function while Dense Layer 3 incorporates a sigmoid activation function, since we want to use a multi-label approach to have our output probabilities independent of each other. The embedding layer is used to transform the two entities of each KG triple into their respective embeddings (vector representations) and the concatenate layer takes as an input, the algebraic sum of the 2 entity vectors 'h' and 't'. It should be noted that the embedding layers for both entities share the same biases and weights along with L2 normalisation (Euclidean distance). Finally, the three dropout layers are used to suppress overfitting.

Figure 1 describes the main architecture of our system with an example. A CKG is passed to the GCN with some incorrect information. The incorrect information is highlighted with a red box. The entities 'connect to the Service Page' and 'clicks an icon' are incorrect. Both these entities are related to the malware entity 'Solarwinds Hack'. The GCN is trained with labels that correspond to semantic triples that are known to be correct. The desired output of our system is not the removal of these entities, but placing probabilistic scores to the relationships held by these incorrect entitites that deem them to be less useful that the other relationships in the CKG. As described above GCN uses convolution layers that uses spatial information to extract repeated features. These process of repeated feature extraction that is performed through 8 hidden layers work on the sub-matrices. Each of these sub-matrices represent the features derived from the interconnections of the set of entities represented by the sub-matrices, which basically represents a sub-graph. Repeated feature extraction on various sub-graphs that contain each entity results in the embedding vector for each entity that is represented by each row of the output matrix $K[i]$.

IV. Experiments and Results

We implement the GCN using Keras and Tensorflow 2.0. Keras is a python-based library that is executable on Tensorflow and Theano (https://keras.io). All models were trained using Keras deep learning API in TensorFlow - on a Google Colab instance with an NVIDIA Tesla T4 GPU and up to 12GB RAM/128GB disk space. To create the training set, we take two approaches. The first approach is deliberately changing some entities and relationships from a correct dataset to incorrect ones randomly chosen from our list of entities and relationships. For example if we have a semantic triple that says a *Malware* 'Pegasus' *uses* *Exploit-Target* 'Android OS', we can choose to change 'Android OS' to 'Windows'. This entity 'Windows' is another *Exploit-Target* that we have in our list of entities, for all CKGs. We can also change the relationship *uses* to any randomly chosen relationship, say,



(a) CKG populated from an After Action Report about Pegasus published in 2018

(b) CKG modified by GCN after being trained with new information

Fig. 3: Comparison of CKGs about a malware Pegasus before and after training

hasVulnerability. We make this change for 15% of the entities and relationships in one graph. This forms noisy CKGs that form K, as an input to our CKG. We retain the unchanged CKG to form the label K^*.

The second approach is generating K^* from updated CTI.For example, we may have a CKG K, that comes from a particular CTI source describing a cyber-attack. We come across a more recent CTI that describes the same cyber-attack. We use the algorithm described as Algorithm 1, to create a new K^* to be used as labels. For any CKG K constructed from fake CTI sources, we also use the same approach to form K^*. We do this wherever possible in our dataset.

We evaluate the results across three parameters : Hits@k, Precision and Recall. For Hits@k, after 50 epochs of training, h and t of a triple in the test data are given to the GCN as input, and we receive another CKG, say, K' as an output with modified scores for all the relationships that exist between the entities. If we look into $K(i, j, :)$ we get the output an array of probabilities for all the relationships that exists between entity E_i and entity E_j. The relationship with the highest probability between 1 and 0 is chosen and is matched with the relationship in $K^*(i, j, :)$ which signifies the true relationship. We use the aforementioned metrics to obtain the scores. We can see the scores in Table II. Given the small number of positive labels for relationships, as compared to the negative labels, we avoid using metrics like Accuracy since their values exceed 0.99 for all experimental settings. This is because for most entity pairs, there is no relationship. The scores are high, due to the small number of training samples for CKGs. Most of the evaluations are correct, because we randomly replace correct entities/relationships with incorrect ones. Some of them may be obvious,and that leads to high scores. It is difficult to identify incorrect relationships that have been asserted from sophisticated fake CTI. We may need more robust models in future to deal with the incorrect assertions from fake CTIs that are realistic.

TABLE II: Scores for our GCN model

Model	**Hits@1**	**Precision**	**Recall**	**F-1**
GCN	0.98	0.986	0.964	0.975

For example, in Figure 3 for the given entities ‘Pegasus’ and ‘iOS’, the predicted relationship output array is as follows

- hasProduct : 0.31
- mitigates : 0.54
- hasVulnerability : 0.36
- uses : 0.82
- modifies : 0.51

Since the predicate ‘uses’ has the highest probability, the predicted triplet is ‘Pegasus - uses - iOS’, which is compared to the true triplet ‘Pegasus - uses - iOS’, thus evaluating to a true prediction. All the test data is used and results are averaged for evaluation.

In Figure 3, we see two CKGs. One of the CKGs (Fig 3a) is asserted with triples extracted from an older report on Pegasus. We see a relatively larger number of triples in this graph that talks about entities like ‘memory corruption’, ‘Facebook’, and ‘Gmail’. Recently, there was another report on Pegasus, specifically talking about how Pegasus used ‘zero-touch’ to get installed in people’s smartphones. This recent version of Pegasus uses a vulnerability in iPhones. Specifically, an old vulnerability in iMessage is used for this attack. This version of Pegasus received a lot of media coverage, and that led to 5 tech reports about Pegasus in our training set for extraction.

If we look at Figure 3b, we see some of the relationships between ‘Pegasus’ and entities derived from older reports have been discarded. This was done by placing a threshold of 0.5 for the top relationship to be retained. It should also be noted that the relationships related to ‘iOS’, specifically ‘zero-click’, ‘iMessage’ had an increase in the relationship scores we receive after passing the original CKG through the GCN. This says that those relationships pertaining to ‘iOS’ and the recent CTIs available for ‘Pegasus’ are to be trusted more, because their probabilistic scores have increased. As we mentioned in Section I, we want to retain information that are new, and we want to score the triples relating to new information higher than triples relating to outdated CTI. Since recent reports about ‘Pegasus’ talk about the iPhone hack, information related to this receive a higher score. The scores for older relationships are lowered. Adaptive thresholding can help us retain more or less of the older relationships depending upon the use-case.

V. Conclusion and Future Work

We discuss a GCN based algorithm that scores semantic triples in a CKG. These scores are high for correct and useful information and are low for outdated and incorrect information. Our work provides a new approach to automatically correct CKGs that may contain wrong information. With the recent developments in ML based automatic CKG generation, it becomes likely that some of the relationships that are asserted are incorrect. This is because even state-of-the-art ML systems are far from being a 100% accurate, and even less so on unseen data. This problem becomes even more challenging in cybersecurity because we come across tech reports about novel attacks and zero-days frequently. ML or NLP based techniques are likely to fail in extracting data that contains new malware-specific information . As CKGs grow in size, it will become impossible for human fact-checkers to correct the information present in them. Our method can be applied in cleaning up CKGs that contain information that have been misclassified by ML algorithms.

Apart from the problems of ML algorithms underperforming, we also run into the risk of retaining information in CKGs that are old. This problem is even more acute in cybersecurity as mentioned in Section I. Our method can be used to quantify the relationships in the CKG such that older relationships receive lower score, and newer information receives a higher score. Thresholding on the scores can lead to the retention of only new information. We demonstrate this capability in Figure 3. We also experimented on fake CTI that are trivial, and our model was successful in identifying them. However, detecting fake information is still challenging. Recent developments with Transformer architectures in neural networks made it possible to generate fake CTI that are difficult to identify even by human experts. In future, our research would be on identifying ‘human-like’ CTI with more sophisticated models.

References

[1] Priyanka Ranade, Aritran Piplai, Sudip Mittal, Anupam Joshi, and Tim Finin. Generating fake cyber threat intelligence using transformer-based models. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2021.

[2] Zareen Syed, Ankur Padia, Tim Finin, Lisa Mathews, and Anupam Joshi. Uco: A unified cybersecurity ontology. In *Workshops at the thirtieth AAAI conference on artificial intelligence*, 2016.

[3] Sudip Mittal, Anupam Joshi, and Tim Finin. Cyber-all-intel: An ai for security related threat intelligence. *UMBC Faculty Collection*, 2019.

[4] Gaurav Sood. virustotal: R client for the virustotal api, 2017. R package version 0.2.1.

[5] External Data Source. Virusshare dataset, 2018.

[6] Oasis group. Stix 2.0 documentation. https://oasis-open.github.io/cti-documentation/stix/examples.html, May 2013.

[7] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating cybersecurity knowledge graphs from malware after action reports. *IEEE Access*, 8:211691–211703, 2020.

[8] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019.

[9] Duncan Riley. *Survey finds CISOs are relying on outdated, report-based threat intelligence*, 2021.

[10] Aritran Piplai, Sudip Mittal, Mahmoud Abdelsalam, Maanak Gupta, Anupam Joshi, and Tim Finin. Knowledge enrichment by fusing representations for malware threat intelligence and behavior. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2020.

[11] Aritran Piplai, Priyanka Ranade, Anantaa Kotal, Sudip Mittal, Sandeep Nair Narayanan, and Anupam Joshi. Using knowledge graphs and reinforcement learning for malware analysis. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2626–2633. IEEE, 2020.

[12] Sudip Mittal, Prajit Das, Varish Mulwad, Anupam Joshi, and Tim Finin. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2016.

[13] Sudip Mittal, Anupam Joshi, and Tim Finin. Thinking, fast and slow: Combining vector spaces and knowledge graphs. *arXiv preprint arXiv:1708.03310*, 2017.

[14] Soham Dasgupta, Aritran Piplai, Anantaa Kotal, and Anupam Joshi. A Comparative Study of Deep Learning based Named Entity Recognition Algorithms for Cybersecurity. In *IEEE International Conference on Big Data 2020*. IEEE, December 2020.

[15] Injy Sarhan and Marco Spruit. Open-cykg: An open cyber threat intelligence knowledge graph. *Knowledge-Based Systems*, page 107524, 2021.

[16] Xu LIANG. Summary of translate model for knowledge graph embedding. https://towardsdatascience.com/summary-of-translate-model-for-knowledge-graph-embedding-29042be64273.

[17] Nidhi Rastogi, Sharmishtha Dutta, Mohammed J Zaki, Alex Gittens, and Charu Aggarwal. Malont: An ontology for malware threat intelligence. In *International Workshop on Deployable Machine Learning for Security Defense*, pages 28–44. Springer, 2020.

[18] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J. Doug Tygar. Can machine learning be secure? In *ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[19] Benjamin Rubinstein, Blaine Nelson, Ling Huang, Anthony Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. Doug Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *ACM SIGCOMM Conference on Internet Measurement*, pages 1–14, 2009.

[20] Marius Kloft and Pavel Laskov. Online anomaly detection under adversarial impact. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 405–412. JMLR Workshop and Conference Proceedings, 2010.

[21] Marius Kloft and Pavel Laskov. Security analysis of online centroid anomaly detection. *The Journal of Machine Learning Research*, 13(1):3681–3724, 2012.

[22] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11957–11965, 2020.

[23] Thabo Mahlangu, Sinethemba January, Thulani Mashiane, Moses Dlamini, Sipho Ngobeni, and Nkqubela Ruxwana. Data poisoning: Achilles heel of cyber threat intelligence systems. In *Proceedings of the ICCWS 2019 14th International Conference on Cyber Warfare and Security: ICCWS*, 2019.

[24] Nitika Khurana, Sudip Mittal, Aritran Piplai, and Anupam Joshi. Preventing poisoning attacks on ai based threat intelligence systems. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

[25] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training

data poisoning? In *international conference on machine learning*, pages 1689–1698. PMLR, 2015.

[26] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[27] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[29] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*, 2015.

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

[31] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

[32] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[34] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 world wide web conference*, pages 1775–1784, 2018.

[35] Federico Monti, Michael M Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *arXiv preprint arXiv:1704.06803*, 2017.

[36] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[37] Yuexin Wu, Hanxiao Liu, and Yiming Yang. Graph convolutional matrix completion for bipartite edge prediction. In *KDIR*, pages 49–58, 2018.

[38] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 968–977, 2019.