Knowledge Graph Construction for Detecting Cybersecurity Attacks

Shraddha Makwana
University of Alberta
Edmonton, Alberta, Canada
smakwana@ualberta.ca

Pranjal Naringrekar
University of Alberta
Edmonton, Alberta, Canada
naringre@ualberta.ca

Abstract

Security analysts in a 'Security Operations Center' (SoC) play a critical role in guaranteeing the organization's security. They have a substantial quantity of background knowledge about the evolving and new attacks. Their ability to detect attacks differs. The dangers of open source Text descriptions of cyber-attacks, are a classic example of intelligence sources. The importance of a cybersecurity knowledge graph can't be overstated because it assists a security analyst in detecting cyber threats. Cybersecurity knowledge graph can be stored in a systematic manner in the form of a database, that keeps track of a wide range of cyber-threat data. This information is stored as semantic triples which contains a link between two cybersecurity entities and can be queried. This paper focuses on extracting the entities and relations from the cybersecurity text, which are basic building blocks for constructing the cybersecurity knowledge graph (CKG). Here, we built the relation extraction model using BERT which will be a classifier that predicts a relation 'r' for a given pair of entities e1, e2. We evaluate our work using the F1 score measure. The work done is publicly available at: https://github.com/pranjal080598/KG_Cybersecurity_Attack

Keywords: Knowledge Graph (KG), Cybersecurity, Transformer and Non-Transformer Model, BERT, Neo4j, UBIAI

1 Introduction

In this task, we performed joint Named Entity Recognition and Relation Extraction between the entities to build a system that can help the security analyst predict the cybersecurity attacks.

Cyber-attacks target individuals, small and medium enterprises with an aim to compromise confidentiality, integrity, and availability of information. Every year cyber defense professionals and researchers find millions of attack and malware variants.Security Analysts hence, have to be up to date with these attacks. Therefore, this idea arises from the fact that a Security Analyst's background knowledge is sometimes insufficient for detecting evolving attacks as they are complex and varied. In order to help them with a decision about cyber-attack, all open source threat intelligence sources need to be stored in a structured way in a cybersecurity knowledge graph that can be queried [7].

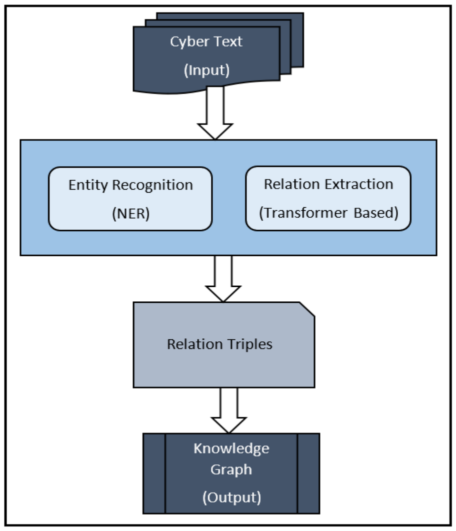


Figure 1. System Architecture

The major motivation of developing this system comes from a news article called 'Attack Samba Server' which specifies that an attack can be caused due to some vulnerabilities exposed on port 445. The attackers usually take advantage of the Log4j vulnerability and port 445 to get access to the network. The analysts need to process the information of these attacks with respect to their local defensive setup. Therefore, our system would take the two entities as input and then derive the relationship between them to aid the security analyst with knowledge of possible attacks. For instance, two entities like Log4j and port 445 are given as input and an exploit relationship is derived and a knowledge graph with other relationships is also formed.

As shown in Figure 1, our project is divided into three sub-categories. In the first part, we would gather data and perform filtering process based on the required cybersecurity text, in second part we will perform cybersecurity entity recognition using Named Entity Recognition and extract the relationship amongst these entities using Transformer Based Model to create a semantic triplet and lastly we would construct a knowledge graph that can be queried. Our aim

UofA'22, April 2022, Edmonton, AB, Canada
2022.

here is to depict a relation extraction model which will be a classifier that predicts a relation 'r' for a given pair of entities e1, e2.

This study contributes in the following terms:

- We provide the KG in the cybersecurity area that does not restrict data extractions to a pre-defined set of data. Our model combines NER, Spacy and BERT to construct a powerful open cyber threat intelligence KG model.
- We show that a KG can be graphically depicted using the simple Neo4J architecture that avoids issues with memory consumption
- We demonstrate that a CKG can be queried easily to find appropriate responses

2 Motivating Example

Figure 2 depicts an example article that specifies that vulnerability 'CVE-2022-27666 787' is found in Linux that affects the heap causing an attacker with a normal privilege to overwrite kernel heap objects that can cause a local privilege escalation threat. In this article the OS called Linus is linked with the CVE vulnerability that cannot be interpreted by a normal data mining system and hence we construct a KG that will find the relation 'Has_Vulnerability' between 'Linux' and 'CVE-2022-27666 787' which are our identified entities. Also, if these entities are searched on google it would not be able to relate it to an attack, however, our relation extractor will make sure to find the relation between these two entities.

3 Overview of Approach

We aim to solve the problem by using a three stage approach: Named Entity Recognizer (NER), Relationship Extraction using transformer based model (BERT), and storing the constructed KG in Neo4j. We aim to chose this method as it divides a complex task into three simple stages. It uses a pre-annotated text that can be easily generated from UBIAI to depict the entities and relations that can be given to the model and also be used to construct the KG.

3.1 Data

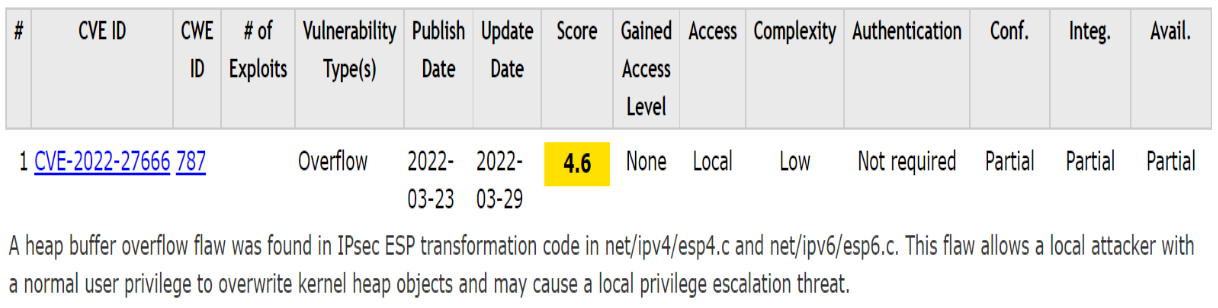
For our implementation, we used the dataset of CVE descriptions as shown in Table 1, which is publicly available at Kaggle. The dataset is divided as 15% dev data, 70% as train data and 15% as test data. We utilise the UBIAI text annotation tool to collect training data because of its flexible interface, that allowed us to quickly switch between entity and relation annotation. UBIAI is an Oregon based startup that provides cloud-based solutions and services in the field of Natural Language Processing (NLP) to help users extract actionable insights from unstructured documents. We converted the annotated data to a binary spacy file before we can train the model and separated the UBIAI annotations into train/dev/test and saved them separately.

Field	Value
CVE_Id	CVE-2019-16548
Mod_Date	2019-11-21 15:15:00
Pub_Date	2019-11-21 15:15:00
CWE_Code	352
CWE_Name	CSRF
Summary	A cross-site request forgery...

Table 1. Sample Data Record

3.2 Phase 1: Entity Recognition

In first stage, we extracted entities using the pre-trained fine-tuned NER model to find Operating System and Vulnerability. We provided the entities from our golden corpus and used them to train the classifier. Spacy's internal rel_component was used for this purpose. SpaCy 3 uses a config file config.cfg that contains all the model training components to train the model that selects the language of the model, NER and hardware (GPU) to use and download the config file template. Later, auto-filled the config file with the rest of the parameters that the BERT model requires and trained the model and saved it under the folder called 'model-best'.



#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2022-27666	787		Overflow	2022-03-23	2022-03-29	4.6	None	Local	Low	Not required	Partial	Partial	Partial

A heap buffer overflow flaw was found in IPsec ESP transformation code in net/ipv4/esp4.c and net/ipv6/esp6.c. This flaw allows a local attacker with a normal user privilege to overwrite kernel heap objects and may cause a local privilege escalation threat.

Figure 2. Example Article

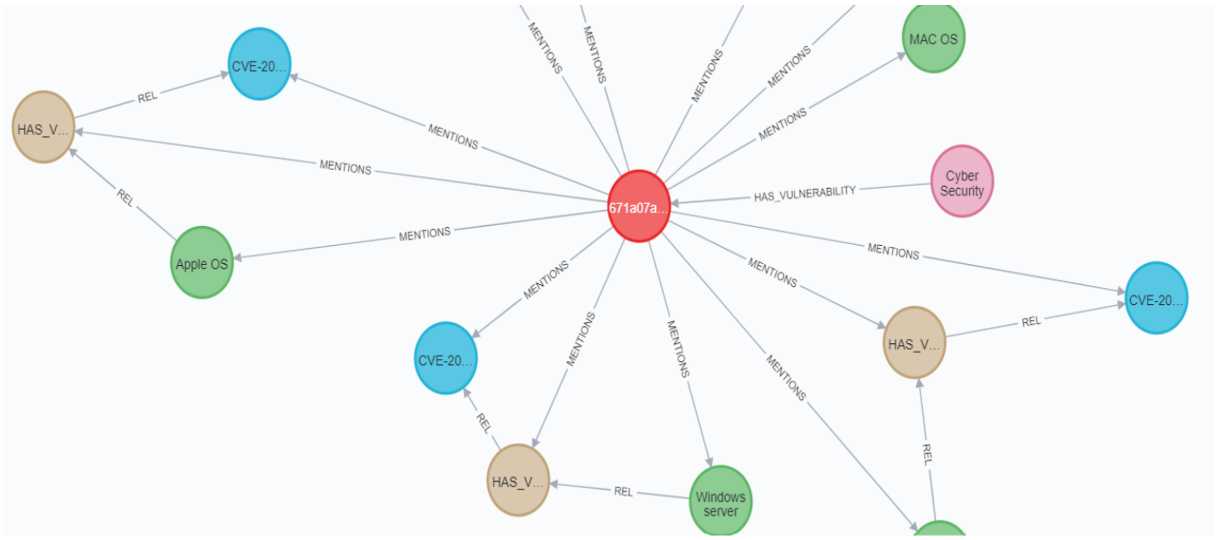


Figure 3. Neo4j Graph

3.3 Phase 2: Relationship Extraction

The second stage focused on implementing the HDSKG solution done in the paper by [8], which uses BERT model to extract the relationships. In this phase we made changes to the model file that helped to decrease the max_length in configs/rel_trf.cfg from the default 100 tokens to 20 tokens so that we could increase the efficiency of our model. The max_length corresponds to the maximum distance between two entities above which they will not be considered for relation classification. As a result, two entities from the same document will be classified, as long as they are within a maximum distance (in number of tokens) of each other. We then trained the transformer and non-transformer model (tok2vec) to predict these relations.

3.4 Phase 3: KG Construction

In third stage, we constructed the KG using Neo4j. While the Python module networkX allowed us to create graphics of our nodes and relationships, the actual graph was saved in Python memory rather than a database. When trying to build scalable systems that must hold an ever-growing knowledge graph, it proved to be an issue. This is why Neo4j is used as it allowed us to save the graph in a fully working database that can handle massive amounts of data and can also be queried. The output can be seen in Figure 2.

Lastly, we will evaluated our method on 15% test data. The F1-score (0.28) for relation extraction by [1] who proposed the openIE tool will remain our baseline and our aim was to maximize this F1-score value.

4 Evaluation

For assessing the performance of our work, we will be using the same evaluation metrics used by [8], which is F1 score. It's calculation is the harmonic mean of Precision and Recall, as follows:

$$F1 = 2 \times \left(\frac{(precision \times recall)}{(precision + recall)}\right)$$

The evaluation measure is calculated based on the overlap between the predicted and actual extracted relationships. Our evaluation function will give credit to partial matches between gold and predicted relationships. The partial credit is proportional to the intersection of the two relationships, and it is normalized by the length of the two entities. The gold label standards for this task was handcrafted by us.

Table 3 highlights the result for entity recognition and relation extraction model. We received an F1 score of 0.8 for entity recognition and F1 score of 0.5 for relation extraction.

Along with this, we also ensured the correctness of the constructed KG using various queries.'Find the vulnerability in Windows', 'Most affected OS' and 'Pair of vulnerabilities that Co-occur the Most' are some of the examples of the executed query.

Model	**Precision**	**Recall**	**F1**
Entity Recognition	0.6	0.4	0.8
Relation Extraction	0.6	0.2	0.5

Table 2. Result of Model

Also, for better performance we tried implementing the Dictionary-based named entity recognition and Bi-LSTM-CRF model to extract entities. Dictionary-based named entity recognition categorized datasets based on collected dictionaries or user-defined dictionaries [4]. However, there is a disadvantage of having to manually organize dictionaries, and because it is necessary to deal with constantly changing and emerging new words over time we tried the Bi-LSTM-CRF model which showed meaningful performance in time series data, using supervised learning-based word embedding and non-supervised learning-based word embedding from a large corpus [2].

5 Discussion

The results are build on the existing evidences reported by [1]. Even though our model was able to reach an F1 score of 0.8 and 0.5 for entity recognition and relationship extraction respectively, it still depicted various faults. We were able to identify errors with respect to various scenarios in the constructed KG.

In first scenario, a vulnerability called 'CVE-2019-16547' was considered as 'CVE' and '2019' separately causing an undefined entity to be constructed. In the second scenario, a wrong relationship was predicted amongst these wrong recognized entities. In third scenario, a relationship that existed between an OS and Vulnerability was not predicted. As shown in Figure 3, there exists no edge between Android and CVE-20.. even if the relationship exists. In order to avoid these errors, categorize all types of entities which are not classified, along with the relationships. Include relation extraction to be trained on more precise entities to avoid interpreting wrong relations.

Apart from this, NEO4J helped in fast searching in the KG. This solution can be implemented for any other independent domain as well. With only a hundred of annotated documents, we were able to train a relation classifier with good performance. However, both tasks are pipelined currently and can't be used separately to train both models simultaneously.

6 Related Work

[3] firstly, constructed a knowledge graph for cybersecurity by collecting and analyzing structured and unstructured data. Secondly, they used Named Entity Recognizer (NER) to extract the entities and to train the extractor. Thirdly, they constructed the ontology according to the information that has been obtained and then generated the cybersecurity knowledge base. They deduced rules based on a quintuple model. New attributes were deduced using the attribute value prediction formula and new relationships between instances were obtained using the relational reasoning predictive formula and the path-ranking algorithm. They made use of a quintuple model whereas we used the transformer based and non-transformer based (Tok2Vec) model.

A research for improving the cybersecurity knowledge graph was performed by [5]. RelExt made use of a semantic triple generation technique that consisted of two cybersecurity entities and the relationship between the entities. They made use of deep learning approaches to extract possible relationships and evaluated their technique by asserting the

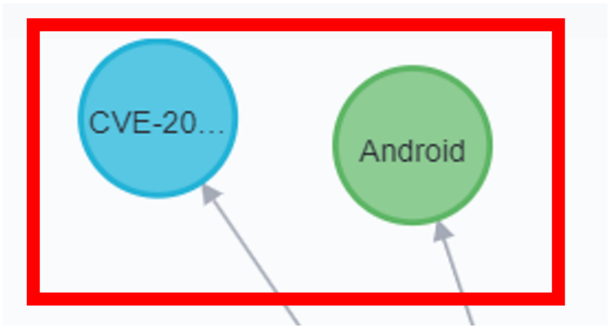


Figure 4. Error Analysis

entity relationship generated by RelExt in the Knowledge Graph to get information about various cybersecurity threats. Their approach combined the existing KG to enhance their system whereas our approach followed a more simple-from-base KG construction.

[6] describes a system that makes use of After Action Reports to extract cyber-knowledge. The entities are extracted using a customized extractor called the Malware Entity Extractor. Later, they construct a neural network unlike our approach that used a simple BERT based Model to predict how pairs of 'malware entities' are related to each other. Furthermore, similar entities are fused to improve CKG that would aid the security analyst to execute queries to retrieve better answers.

7 Conclusion

We were able to solve the problem of matching vulnerabilities in OS's by using Named Entity Recognizer (NER) and transformer based model (BERT) to extract entities and relationships, respectively and store the constructed KG in Neo4j that can be queried. We achieved this by building a KG linking OS and vulnerabilities together. Our results clearly depict that our entity recognition model performed better than the relationship extraction model. As a part of our error analysis we were able to identify few errors in the constructed KG and also proposed methods to resolve them. KGs, while they useful and efficient, they may sometimes result in incompleteness, redundancy, and ambiguity, which can lead to uninformative query results.

In future, we will also try to achieve high performance for this task by exploring multiple state-of-the-art Natural Language processing based techniques such as Markov Logic Networks and experimenting with other different types of feature extraction to understand and evaluate Relation Extraction Triples. This will help in constructing a more precise KG. Combining NLP with Neo4j's graph DB can help us accelerate information discovery in many domains, with more notable applications in healthcare and biomedical.

References

[1] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. 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)*. 344–354.

[2] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv:1508.01991 [cs.CL]

[3] Yan Jia, Yulu Qi, Huaijun Shang, Rong Jiang, and Aiping Li. 2018. A Practical Approach to Constructing a Knowledge Graph for Cybersecurity. *Engineering* 4, 1 (2018), 53–60. https://doi.org/10.1016/j.eng.2018.01.004 Cybersecurity.

[4] Arvind Neelakantan and Michael Collins. 2014. Learning Dictionaries for Named Entity Recognition using Minimal Supervision. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, 452–461. https://doi.org/10.3115/v1/E14-1048

[5] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. 2019. RelExt: Relation Extraction using Deep Learning approaches for Cybersecurity Knowledge Graph Improvement. arXiv:1905.02497 [cs.CL]

[6] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. 2020. Creating Cybersecurity Knowledge Graphs From Malware After Action Reports. *IEEE Access* 8 (2020), 211691–211703. https://doi.org/10.1109/ACCESS.2020.3039234

[7] Injy Sarhan and Marco Spruit. 2021. Open-CyKG: An Open Cyber Threat Intelligence Knowledge Graph. *Knowledge-Based Systems* 233 (2021), 107524. https://doi.org/10.1016/j.knosys.2021.107524

[8] Xuejiao Zhao, Zhenchang Xing, Muhammad Ashad Kabir, Naoya Sawada, Jing Li, and Shang-Wei Lin. 2017. Hdskg: Harvesting domain specific knowledge graph from content of webpages. In *2017 ieee 24th international conference on software analysis, evolution and reengineering (saner)*. IEEE, 56–67.