Combating Fake Cyber Threat Intelligence using Provenance in Cybersecurity Knowledge Graphs

Shaswata Mitra*, Aritran Piplai†, Sudip Mittal*, Anupam Joshi†
* Department of Computer Science & Engineering, Mississippi State University,
Email: sm3843@msstate.edu, mittal@cse.msstate.edu
†Dept. of Computer Science & Electrical Engineering, University of Maryland, Baltimore County,
Email: {apiplai1, joshi}@umbc.edu

***Abstract*—Today there is a significant amount of fake cybersecurity related intelligence on the internet. To filter out such information, we build a system to capture the provenance information and represent it along with the captured Cyber Threat Intelligence (CTI). In the cybersecurity domain, such CTI is stored in Cybersecurity Knowledge Graphs (CKG). We enhance the exiting CKG model to incorporate intelligence provenance and fuse provenance graphs with CKG. This process includes modifying traditional approaches to entity and relation extraction. CTI data is considered vital in securing our cyberspace. Knowledge graphs containing CTI information along with its provenance can provide expertise to dependent Artificial Intelligence (AI) systems and human analysts.**

***Index Terms*—Cybersecurity, Fake Data, Provenance, Cyber Threat Intelligence.**

I. Introduction

Cyber Threat Intelligence (CTI) is crucial for security organizations to discover and analyze cyber-risks. A recent report [1], mentions data quality as one of the most critical components of CTI. We have also seen that the poor quality of threat intelligence entails huge costs to organizations that use them [2]. Significant threat vectors include generated fake CTI [3]. Recent reports also suggest that the majority of security organizations use CTI that is of no use anymore [4], and that leads to incorrect decisions causing harm to organizations. With news articles talking about the impact of bad quality threat-intelligence becoming rampant, the cybersecurity community is waking up to address these concerns.

The data sources for CTI can be both structured and unstructured. However, using open-source intelligence for CTI is a common practice. An National Security Agency (NSA) report suggests that some countries, like Russia, rely on open-source for 90% of the threat-intelligence that they gather [5]. Unstructured open-source intelligence can come from a variety of places, and it is difficult to verify the information present in them. These sources can be social media posts, blogs, technical reports from organizations with varying degrees of reputation. It is imperative that we record key information about the source of the threat-intelligence. For example, if a CTI originating from a Twitter post suggests that a cyber-threat can be mitigated by updating Adobe Acrobat, it should carry less weight than a CTI from Adobe Security Bulletin that says otherwise. Recording provenance level information provides us with the ability to create additional filters on the threat-intelligence that practitioners use.

In order to effectively use CTI to evaluate and mitigate risks, organizations need to process the raw data feeds and appropriately represent them. Cybersecurity Knowledge Graphs (CKG) are becoming popular to represent CTI. This is because CKG has reasoning capabilities that are applied on the interconnected entities. These reasoning capabilities enforce rules that help in preserving credible information and discarding the rest. Additionally, classes capturing the provenance can be added in the schema of the CKG that can give us more information about the source of the data. In our paper, we improve an existing CKG schema that records CTI with additional classes and relationships that indicate provenance.

The schema of the CKG, as mentioned before, tells us the entity-classes and the relationship-classes. After the schema is defined, various semantic triples are asserted in the CKG. The semantic triples record a specific relationship existing between pairs of entities. Unstructured sources for CTI need to be processed to extract the entities and relationships. We use Natural Language Processing (NLP) techniques to extract the CTI information representing a cyber-attack or a malware, along with the information related to the provenance. The objective of our paper is to associate provenance with the entities asserted in the CKG. Once the entities are associated with the corresponding provenance, we also come up with a provenance score for a set of entities and relationships. In order to improve the current schema to capture provenance related information, we update current CKG schema to include provenance specific entity-classes and relationship-classes. After that, we extract the entities and relationships using Machine Learning (ML) based NLP techniques.

The summary of our key contributions is as follows:

- Creating a novel schema for a CKG that accounts for provenance related entity-classes and relationship-classes.
- Using ML and NLP-based methods for extracting provenance from open-source text.
- Associating entities of existing CKG with provenance.
- Creating a score for provenance quality across a set of entities and relationships representing a CTI.

We organize our paper as follows. In Section II, we discuss some of the relevant papers for our research. The new ontology and the data extraction techniques are discussed in Section

III. In Section IV, we discuss some of the results and the knowledge graph outputs. Finally, we conclude and discuss the next steps for our research in Section V.

II. Related Work

With the rapid growth of knowledge graphs in the domain of cybersecurity, the authenticity of the information becomes prominent. To identify the information source and reliability, we propose to embed such information with the existing solutions. In this section, we talk about relation extraction and similar works in this domain.

A. Named Entity Recognition

Malware Entity Extractor (MEE) is the first step that helps in identifying the entities required for assertion in the CKG. MEE is a malware-specific Named Entity Recognizer (NER), is a subtask of information extraction and classification. It is a 'sequence-to-sequence' classification task. The goal of this task is to parse an unstructured text and map each word to one of the predefined classes that we derive from an ontology. Piplai et al. in their paper [6], describe an MEE using Conditional Random Fields (CRF) and Regular Expressions. There have been other NERs that were built using Support Vector Machines (SVM) [7], neural networks [8], [9], LSTM [10], and Bi-directional LSTM [11]. LSTM and Bi-directional LSTM are widespread for language processing tasks due to context evaluation. But for entity identification, context is not of primary choice. The process constitutes two phases, detection of desired entities and classification of such entities accordingly. Ekbal et al. have proposed in their paper [7] classifying language-independent named entities using SVM. Piplai et al. [6] have demonstrated a similar task for cyberthreats using CRF. In our case, we configure the Stanford NER to classify provenance from Cyber Threat Intelligence (CTI).

B. Relationship Extraction

The cybersecurity knowledge graphs (CKGs) are asserted with semantic triples. Relation extraction establishes a relationship between pairs of entities that were found by the MEE. This yields semantic triples that form the building block of the CKG. Obtaining a relation between entities is not as straightforward as it seems. Not all entities always hold a relation. The ambiguity of data and its schema structure makes relationship extraction necessary to connect the interlinked entity pairs hidden underneath. Relationship extractors can be classified into global and mention levels depending on their approach. Mention level determines if a relation holds between two entities. On other hand, the global level determines all entity pairs that hold a specific relation. The approach behind such can be either supervised, semi-supervised or unsupervised. Based on all that, a relationship extractor can be either binary or n-ary. Binary determines if a relation holds between two entities and, n-ary determines if any relation from the relation set holds between two entities.

TransE(h,r,t) model [12] is used to extract one-to-one relation(r) between head(h) and tail(t) using L1 or L2 norms on vector plane. But TransE cannot determine many-to-many relations. Thus TransH(h,r,t) [13] is used to find many-to-many relations by projecting the entity-pair(h,t) on a hyper-plane. As TransH uses the same hyperplane for entity-pair(h,t) relation determination, thus the aspect of the relationship is overlooked. To address the context of the relationship, TransR(h,r,t) [14] is used to find similar many-to-many relations by using multiple relation hyper-plane for an entity-pair(h,t) so that the aspect of that relationship is also addressed. While addressing the aspect, for better accuracy the entities are also clustered at entity space. That is called CTransR, which is another variant of TransR. But for the heavy computation requirement of TransR, the TransD approach is also used. In TransD, entity-pair(h,t) are projected using different projection matrices on hyperplane, making more computation efficient and similarly accurate. [15]

Pingle et al. [16] developed a supervised relationship extractor that classifies pairs of entities into one relationship class of a predefined ontology. Word2Vec [17] was used as an embedding model to generate vectors of fixed dimensions for the extracted Named Entities. The supervised model is a feed-forward neural network and it produces an entity-relationship set as an output based on an underlying UCO ontology [18]. Some dependent systems include [6], [16], [19]–[27]. The output entity relationships are constructed using many-to-many, many-to-one, and one-to-one relations. Obtained relation set can be further used to fetch necessary real-time information by various sources, e.g., malware detection, using CKG. However, this model fails to incorporate provenance scores of the semantic triples. In our paper, we redefine the schema of UCO 2.0 [16]. The updated schema helps to find provenance scores of the semantic triples using a more refined relationship extraction.

C. Cybersecurity Knowledge Graphs(CKGs)

A knowledge graph represents a collection of interlinked descriptions of subjects, predicates, and objects. Knowledge graphs put data in context via linking subject and object using a predicate. In this way, it provides a framework for data integration, unification, analytics, and sharing. Cybersecurity Knowledge Graphs have widely been used to represent Cyber Threat Intelligence (CTI). This gives cybersecurity analysts a significant edge in querying the system over the information. The cyber threat reports obtained from After Action Reports (AARs) published by organizations and open-source blogs get stored in CKG as semantic triples.

A CKG has mainly two parts, A schema or ontology, and numerous cybersecurity-related semantic triples. Pingle et al. in their paper [16], have developed a cybersecurity ontology called Unified Cybersecurity Ontology (UCO 2.0). UCO 2.0 is developed on top of UCO 1.0 [18] using Structured Threat Information Expression (STIX 2.0) [28]. Such improvements have opened new possibilities for CKG. Piplai et al. in their

paper [6], showcased that further malware analysis using machine learning algorithms is possible using CKG.

D. Provenance and Trust Aware Inference Framework

Provenance is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability, or trustworthiness [29]. With the rapid growth of the internet, data, and computation, generating automated information and relation from open source and official data becomes troublesome due to the trust factor. To process data in an orderly manner and generate authentic information, the authenticity of the data classification needs to be addressed. The work of Ding et al. [30] proposes to filter and classify data based on provenance. The framework rates authenticity of an RDF graph based on its origin. It encodes the origin information by drawing relation from the triples and then calculating with trust indexing. The processed output data helps to discover facts from the given data set with more authenticity and confidence. The kind of provenance information and level of granularity are two major design issues considered in their work. The RDF graph is decomposed based upon grounded nodes into RDF molecules using naive and functional decomposition and heuristic merge-operations. RDF molecules are then converted from triple to quad format (subject, predicate, object, source) to add provenance. [31]

III. Methodology

We use our pipeline to assert malware entities and the corresponding provenance to add in CKG. Our Malware Entity Extractor (MEE) extracts cybersecurity entities from data sources like blogs, tweets, After Action Reports (AARs) [6]. We have extended MEE to enable provenance extraction. The re-trained Provenance Extractor (ExP) extracts and represents provenance triples from the data source according to our ontology. The generated provenance sub-graph is then merged with CKG.

Our pipeline contains four components:

- *Cyber Threat Intelligence (CTI)*: We represent CTI in our Cybersecurity Knowledge Graph. For our experiments in this paper, we consider threat information present in AARs and cybersecurity blogs.
- *CKG Extraction Pipeline*: We use the existing CKG extraction pipeline, developed by Piplai et al. to process the CTI and populate the CKG. It involves 3 major parts, UCO, MEE, and RelExt [6], [16].
- *Provenance System*: The provenance system has two parts, *Provenance Extractor (ExP)* and *Provenance Encoder (EnP)*. Provenance Extractor (ExP) extracts the triples according to our ontology classes (described below). Then we generate the provenance graph using Provenance Encoder (EnP).
- *Provenance Fusion System*: A Provenance Fusion System (PrF) merges the extracted provenance sub-graph with existing intelligence present in the CKG. The merged CKG includes the provenance graph together with the intelligence details.

Next, we describe each of the 4 components in detail.

A. Cyber Threat Intelligence (CTI)

An After Action Report (AAR) is a form of retrospective analysis on a given sequence of security-oriented actions complied by a cybersecurity professional [6]. The purpose of an AAR is to analyze the management or response to an incident, exercise, or event by identifying strengths to be maintained and built upon, as well as identifying potential areas of improvement [32]. There are two forms of AAR, 'Literary AAR' intended for recreational use, and 'Analytical AAR' exercised as part of a process of performance evaluation and improvement. In most cases, AARs include both. The Homeland Security Exercise and Evaluation Program by the United States Department of Homeland Security has mandated the format to share exercises and evaluations to make the information sharing consistent across jurisdictions [6].

In the domain of cybersecurity, an AAR contains a detailed report about cyber-attacks and associated entities. It also includes mechanisms to detect such events and mitigating strategies. Security organizations like, Kaspersky [33], FireEye [34] also publish AAR along with government agencies. It includes 'malware', 'threat-actors' used in the campaign, with targeted applications and attack vectors. The detailed overview gives a clear picture of the attack and how a user should detect and react. An AAR includes much more detailed insights compared to internet blogs. When written according to the mandated format it becomes a vital source of intelligence. Along with this, most AARs are published by reputed organizations, making them a more reliable source. Such nature makes these reports a credible source of intelligence, as opposed to mining data from dark web logs, or social media as demonstrated by Mittal et al. [19], [20].

Piplai et al. [6] also include in their solution internet blogs and Open Source Intelligence (OSINT) [35] [36] to improve data and knowledge diversity. These blogs are commonly written by independent authors and might not be authentic from an organizational standpoint. On top of that hackers try to poison training models using data poisoning to satisfy their malicious intent [3], [37]. Therefore, our solution adds a provenance information to the CKG. We extract provenance entity information using our Provenance Extractor and include it in our CKG.

B. CKG Extraction Pipeline

Piplai et al. in their paper [6] described a CKG extraction pipeline to represent malware-specific information. The CKG schema is based on UCO 2.0 [18]. Some of the classes have been modified to better capture malware-specific information. A CKG for each unstructured text report about malware is constructed. The pipeline has three components:

- MEE: A Malware Entity Extractor that is based on Conditional Random Fields and Regular Expressions. The

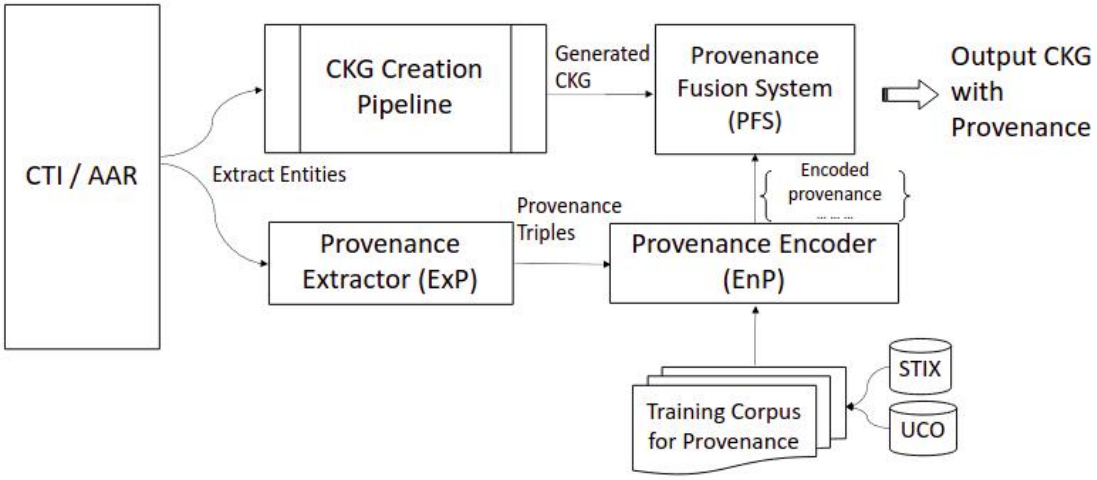

Fig. 1. Architecture Pipeline Diagram.

extractor maps words to known classes of the predefined schema.

- Relationship Extractor (RelExt): The Relationship Extractor establishes a relationship between pairs of entities extracted by the MEE. Pairs of entities that can have no credible relationship between them are filtered out and are not passed to the Relationship extractor. This is a neural network classifier that takes word embeddings of pairs of entities and maps it to one of the 6 relationship classes of the predefined schema. Another class for 'no relationship' is added to the list of possible classes.
- Knowledge Graph Assertion and Fusion: Once the relationship extraction has found out the correct relationship between pairs of entities, we have semantic triples of the form 'Entity - Relationship - Entity'. Each unstructured text report leads to a CKG, that is merged with others if they share the same or similar entities. The 'owl: SameAs' assertion is used for merging entities of two different CKGs and this step is called the 'fusion'. If two entities are not exactly the same, the Edit Distance algorithm is used to see if the entities are similar enough to be fused.

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, hasHash, mitigates, hasVulnerability, uses

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.

In our paper, we use this pipeline to generate CKGs from unstructured text reports about malware. The CKGs that result from this pipeline have a schema that does not capture the provenance level information. The schema is a modified version of UCO 2.0, and we use this to represent the threat information. We keep the existing schema of this paper, and we include more classes and relationships to create a more enriched schema that helps us capture the information we can use for provenance.

C. Provenance System

The provenance system is to gather CKG sources to address the authenticity of that data. We consider URL, Publisher, etc. as a source. In Provenance Extractor we extract triples mentioned in UCO 3.0 from the data source to draw relations among them using the Provenance Encoder.

1) Provenance Extractor (ExP): Syed et al. [18] in their paper have developed the Unifed Cybersecurity Ontology (UCO) 1.0, which was updated to UCO 2.0 [16]. These updates in the ontology have been in line with the updates in STIX from 1.0 to 1.2 [28]. We extend UCO 2.0 to add provenance information into the CKG by redefining the existing schema and possible relations. Next, we explain various provenance classes that have been added in UCO 3.0, which inherits the cybersecurity schema and domain knowledge from UCO 2.0.

- *Intelligence-Id*: An entity class that refers to a unique CTI represented in our CKG.
- *URL*: An entity class that refers to the originating URL from where the CTI was collected.
- *Organization-Name*: An entity class that refers to a name of an organization that published the CTI.
- *Author*: An entity class that refers to the name of the author of the CTI.
- *Country*: An entity class that refers to the originating country. Such as USA, Canada, Israel, etc.
- *Origin-Type*: An entity class that refers to the type of CTI, such as AAR or Blog etc.
- *Creation-Date*: An entity class that refers to the date of the publishing.

- *Provenance-Score*: A value that refers to the amount of knowledge the CKG posses on that entity.

ExP is an updated version of MEE. It was trained using UCO 3.0 to identify the provenance entities from various CTI sources. As UCO 3.0 is developed on top of UCO 2.0, it contains all the entity classes in UCO 2.0 along with the above-listed classes. While executing, the Provenance Extractor parses the CTI sources and classifies the intelligence according to UCO 3.0. While parsing each CTI, we generate a unique *Intelligence-Id* to identify cybersecurity knowledge and allow graph fusion. Based on the Provenance Extractor results and entity vector embeddings obtained using Word2Vec [17], we determine provenance relationships for the particular CTI intelligence. We also keep the count of an entity occurrence to calculate the CKG confidence score for a specific CTI, this informs the value of the *Provenance-Score*.

2) Provenance Encoder (EnP): In this phase, we create a provenance relation encoder that helps represents provenance in semantic triples. Then we merge the encoded provenance sub-graph with CKG using our Provenance Fusion System (PFS). The following is the description of our defined schema to add provenance into CKG. Each class and possible relations have been defined in UCO 3.0:

- *mentionedIntelligence*: Domain: All UCO 2.0 entities, Range: Intelligence-Id
- *mentionedUrl*: Domain: Intelligence-Id, Range: URL
- *wasGeneratedBy*: Domain: URL, Range: Organization-Name
- *authorName*: Domain: URL, Range: Author
- *wasDerivedFrom*: Domain: URL, Range: URL
- *countryOfOrigin*: Domain: Organization-Name, Range: Country
- *isTypeOf*: Domain: URL, Range: Origin-Type
- *createdOn*: Domain: URL, Range: Creation-Date
- *usedBy*: Domain: URL, Range: Organization-Name
- *hasProveance*: Domain: Provenance-Score, Range: "URL", "Organization-Nam", "Author", "Origin-Type"

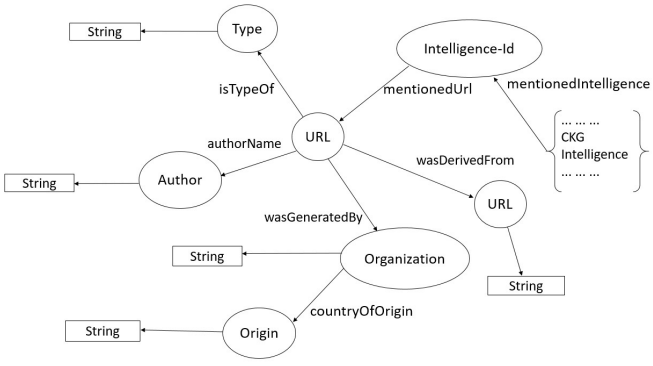

Fig. 2. Provenance Relationship diagram.

Our encoder was trained by modifying the existing Word2Vec based RelExt developed by Pingle et al. [16]. While determining relations among UCO 3.0 entities, we also generate triple between all identified UCO 2.0 entities with Intelligence-Id using *mentionedIntelligence* relation. The main reason behind creating such a relation is to enable graph fusion in CKG and maintain entity-specific origins. Along with this we also relate intelligence with its origin URL using *mentionedUrl* relation.

In our approach, each AAR/CTI source is mapped with a unique Intelligence-Id. Thus, all entities coming from a similar source should point to the same Intelligence-Id. But, any entity can point to multiple Intelligence-Id as the same entity can be derived from various data sources. Once the relation triples generation among the provenance entities, we calculate entity weight. The entity weight is calculated using the softmax function. After calculating the weights of each entity, we calculate the entity confidence score by computing the mean score of all the associated entities. While calculating, we exclude "URL", "CreationTime", and "Type" from keeping the count. All URLs are considered separate entities in our provenance model. We only pre-set the "Type" weight values for AAR, OSINT, and Blogs. We set the provenance score for AAR as 0.7 and Blog as 0.3. Using this method we compute the CKG confidence score of an URL entity node. We set the value based on our available data-set comparison. The outcome of our Provenance Encoder is a Provenance Knowledge Graph. We then fuse the graph with CKG in the Provenance Fusion System.

D. Provenance Fusion System (PFS)

Once relation extraction and confidence calculation have completed, we fuse the provenance graph with CKG by identifying the correct entities. We fuse the graph by relating a particular entity to Intelligence-Id using *mentionedIntelligence* relation. As the output of our encoder system "EnP" is an existing provenance graph therefore we merge the two graphs using the union finding method. Ding et al. in their paper [30] has proposed molecular mechanism while graph decomposition. Their solution includes lossless decomposition of a graph by removing BNodes using molecules with their characteristics. In our solution, while executing graph fusion, we keep the same molecular join without adding any BNodes. This ensures efficiency in our representation. Figure 3, describes the relation set among UCO 3.0 entities after graph fusion.

IV. Experimental Result

We assert the extracted entities and relationships relating to provenance to a CKG that contains CTI information. We generate a CKG with malware-specific information and provenance. Figure 4 is a parsed output of CKG with a provenance on two AAR generated by "Lookout" organization on "Dark Caracal" and "Pegasus" malware. In the diagram, the fused CKG with provenance using UCO 3.0 is shown. The existing CKG adhering UCO 2.0 remains the same. Here we fuse the Provenance graph with CKG by intelligence-id: "Intl-101721" and "Intl-102121". Each intelligence refers to the URL by the mentionedURL attribute. "Intl:

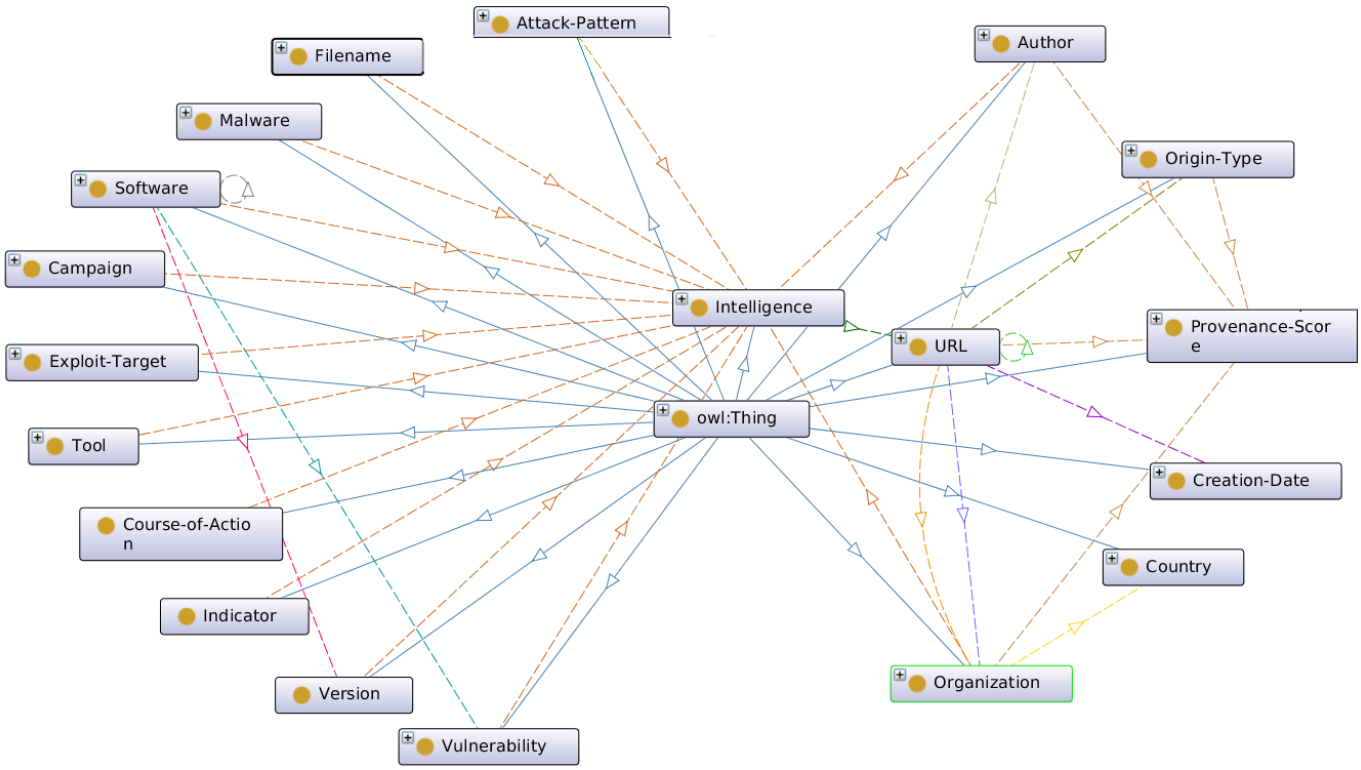

Fig. 3. UCO 3.0 Ontology relation diagram.

101721" points to "Lookout-Dark-Caracal" AAR URL. "Intl-102121" points to the "Lookout-Pegasus-Android-technical-Report" AAR URL. Now "Dark-Caracal" AAR Url is derived from multiple URLs. Those can be of type either AAR or Blog. For example, "Lookout-Dark-Caracal" points to "i-got-a-letter-from-the-government" using *wasDerivedFrom* relation, which is type AAR. Similarly points to "T1093", "trident-pegasus", "pegasus-android" with the same *wasDerivedFrom* relation but these are of type Blog. In the same manner "Lookout-Pegasus-Android-technical-Report" points to "Lookout-Pegasus-Technical-Analysis" which is another AAR generated by Lookout. A URL also keeps track of its creation date using *createdOn* relation. Here we can see that "Lookout-Dark-Caracal" was *createdOn* "01/18/2018" and "Lookout-Pegasus-Android-technical-Report" was *createdOn* "04/01/2017". Along with an URL keeps its origin information using *wasGeneratedBy* attribute. Here we can see "Lookout-Dark-Caracal" and "Lookout-Pegasus-Android-technical-Report" both *wasGeneratedBy* the "Lookout" organization, which has "countryOfOrigin" "USA". An URL also keeps track of its author using *authorName* attribute. For example, here we can see "Mike Murray" is the author of the blog "pegasus-android".

Apart from the provenance graph, all entities, which were found from a specific data source are also pointed to their corresponding Intelligence-Id to keep track of its origin. For example we can see that malware "Pegasus" *uses* tools like "Skype", "C2-Server", "SMS", "SSH" etc. to execute its operation. Thus after provenance fusion, all the entities point to "Intl-102121" to keep track that all the data was mentioned in URL "Lookout-Pegasus-Android-technical-Report" using one-to-one mapping. In the same manner "Dark-Caracal" uses "Pallas", thus it points to "Intl-01/18/2018". In this manner all entities from UCO 2.0 point to its origin intelligence. The main advantage of such implementations is that while querying CKG about the relations, we can get not only the data but the source where it was derived from. Which is the main purpose of our solution. Refer to the following diagram for further details on the Provenance Graph.

According to Figure 4, if we want to find the names of the malware which are mentioned in AAR, then we can place query like following:

```
SELECT ?x WHERE {
  ?x a CKG:Malware ;
  ?x a CKG:mentionedIntelligence ?y .
  ?y CKG:Intelligence-Id ?z.
  ?z CKG:mentionedURL ?q.
  ?q CKG:isType AAR . }
```

The query would return "Dark Caracal" and "Pegasus".

Similarly, to find the CTI URL that refer to malware "Pegasus" and was reported on 01/01/2017, then we can place query like following:

```
SELECT ?y WHERE {
 ?x a CKG:Intelligence-Id ;
 ?x a CKG:mentionedUrl ?y.
```

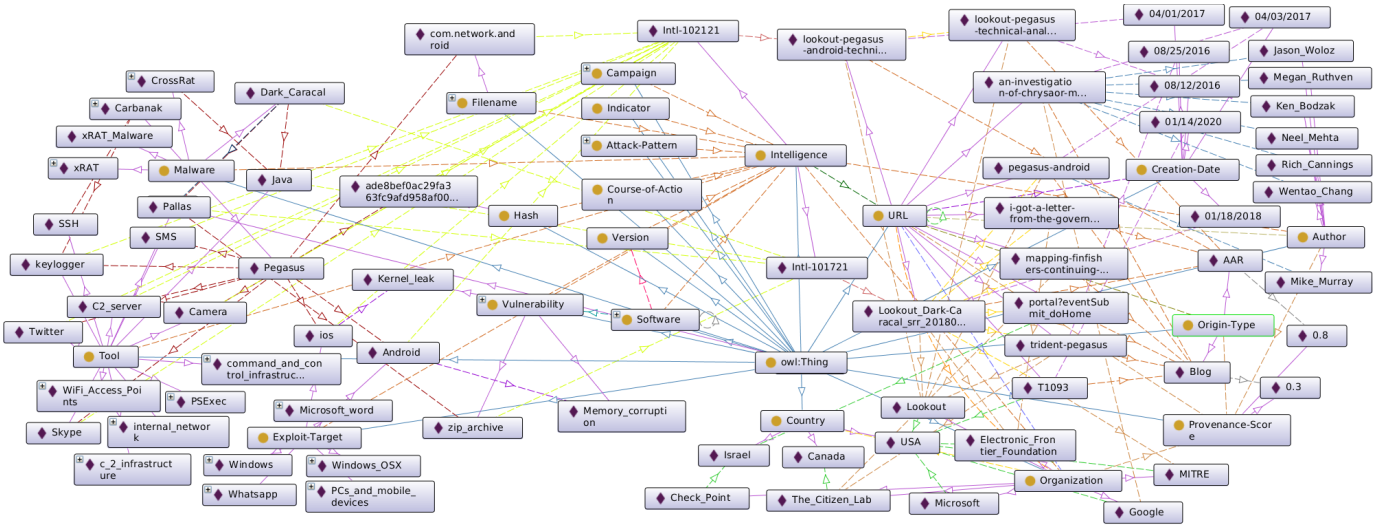

Fig. 4. Data diagram of Fused CKG with Provenance.

```
?y a CKG:createdOn ?z .
?z a CKG:Creation-Date : 04/01/2017;
?m a CKG:Malware .
CKG: Pegasus and
?m a CKG:mentionedIntelligence ?i ;
?i a CKG:Intelligence-Id: ?x . }
```

The query would return "Lookout-Pegusas-Android-technical" URL: "https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-android-technical-analysis.pdf" as the creation date is "04/01/2017" and "Pegasus" is mentioned in the CTI.

V. Conclusion

Cybersecurity Knowledge Graphs (CKG) are useful for storing a large number of semantic triples about cybersecurity entities. Encoding the provenance of such CTI data, helps us determine the authenticity and reliability of the included knowledge. In this paper, we have successfully updated the existing CKG generation pipeline and incorporated provenance in CKG. The system extracts provenance data about the mined CTI and fuses such data with CKG entities. Including provenance information helps end-users get cybersecurity information along with its origins. In the future, we can use graph neural models to create node embeddings of entities that incorporate provenance information encoded in our CKG. Such a solution can help incorporate a trust dimension in these representations. We can also extend the schema of our ontology to capture more information on provenance, which would lead to more informed resolutions by our CKG reasoner.

Acknowledgement

This work was supported by a U.S. Department of Defense grant and National Science Foundation grant #2133190.

References

[1] What are the most critical components of threat intelligence and how do you take action on them?, 2021.
[2] Andy Pendergast. The cost of bad threat intelligence, 2015.
[3] P Ranade, A Piplai, S Mittal, A Joshi, and T Finin. Generating fake cyber threat intelligence using transformer-based models. In *International Joint Conference on Neural Networks 2021 (IJCNN 2021)*, 2021.
[4] Threat intelligence: The biggest blind spot for cisos, 2021.
[5] Intelligence collection activities and disciplines.
[6] 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.
[7] Asif Ekbal and Sivaji Bandyopadhyay. Named entity recognition using support vector machine: A language independent approach. *International Journal of Electrical, Computer, and Systems Engineering*, 4(2):155–170, 2010.
[8] Edson Florez, Frédéric Precioso, Michel Riveill, and Romaric Pighetti. Named entity recognition using neural networks for clinical notes. In *International Workshop on Medication and Adverse Drug Event Detection*, pages 7–15. PMLR, 2018.
[9] Anantaa Kotal Soham Dasgupta, Aritran Piplai 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.
[10] James Hammerton. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 172–175, 2003.
[11] Anantaa Kotal Soham Dasgupta, Aritran Piplai 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.
[12] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.
[13] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.
[14] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.
[15] Xu LIANG. Summary of translate model for knowledge graph embedding. https://towardsdatascience.com/summary-of-translate-model-for-knowledge-graph-embedding-29042be64273.
[16] 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.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[18] 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.

[19] 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.

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

[21] 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.

[22] Matthew Sills, Priyanka Ranade, and Sudip Mittal. Cybersecurity threat intelligence augmentation and embedding improvement-a healthcare usecase. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2020.

[23] 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.

[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] Priyanka Ranade, Sudip Mittal, Anupam Joshi, and Karuna Joshi. Using deep neural networks to translate multi-lingual threat intelligence. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 238–243. IEEE, 2018.

[26] Lorenzo Neil, Sudip Mittal, Anupam Joshi, et al. Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports. *IEEE Intelligence and Security Informatics (IEEE ISI) 2018*, 2018.

[27] Ketki Sane, Karuna Pande Joshi, and Sudip Mittal. Semantically rich framework to automate cyber insurance services. *IEEE Transactions on Services Computing*, 2021.

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

[29] W3C. Constraints of the prov data model. https://www.w3.org/TR/2013/REC-prov-constraints-20130430/.

[30] Paulo Pinheiro da Silva Li Ding, Yun Peng and Deborah L. McGuinness. Tracking RDF Graph Provenance using RDF Molecules. Technical report, UMBC, April 2005.

[31] Li Ding, Pranam Kolari, Tim Finin, Anupam Joshi, Yun Peng, Yelena Yesha, et al. On homeland security and the semantic web: A provenance and trust aware inference framework. In *Proceedings of the AAAI Spring Symposium on AI Technologies for Homeland Security*, 2005.

[32] University of Houston Office of Emergency Management. After action report. https://uh.edu/emergency-management/planning-and-response/building-department-preparedness/after-action-report/.

[33] KasperSky. Kaspersky apt intelligence reporting. https://usa.kaspersky.com/enterprise-security/apt-intelligence-reporting.

[34] FireEye. Fireeye reports. https://www.fireeye.com/current-threats/threat-intelligence-reports.html.

[35] Michael Glassman and Min Ju Kang. Intelligence in the internet age: The emergence and evolution of open source intelligence (osint). *Computers in Human Behavior*, 28(2):673–682, 2012.

[36] Robert David Steele. Open source intelligence: What is it? why is it important to the military. *American Intelligence Journal*, 17(1):35–41, 1996.

[37] MITRE ATLAS. Adversarial machine learning. https://atlas.mitre.org/resources/adversarial-ml-101/.