IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 6, 2020, accepted November 10, 2020, date of publication November 19, 2020, date of current version December 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3039234

Creating Cybersecurity Knowledge Graphs From Malware After Action Reports

ARITRAN PIPLAI[1], SUDIP MITTAL[2], (Member, IEEE), ANUPAM JOSHI[1], (Fellow, IEEE), TIM FININ[1], JAMES HOLT[3], AND RICHARD ZAK[3,4]

[1]Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County, Baltimore, MD 21201, USA
[2]Department of Computer Science, University of North Carolina at Wilmington, Wilmington, NC 28403, USA
[3]Laboratory of Physical Sciences, Maryland, MD 20740, USA
[4]Booz Allen Hamilton, McLean, VA 22102, USA

Corresponding author: Aritran Piplai (apiplai1@umbc.edu)

This work was supported in part by a United States Department of Defense grant, in part by a gift from IBM research, and in part by a National Science Foundation (NSF) grant, award number 2025685.

ABSTRACT After Action Reports (AARs) provide incisive analysis of cyber-incidents. Extracting cyber-knowledge from these sources would provide security analysts with credible information, which they can use to detect or find patterns indicative of a cyber-attack. In this paper, we describe a system to extract information from AARs, aggregate the extracted information by fusing similar entities together, and represent that extracted information in a Cybersecurity Knowledge Graph (CKG). We extract entities by building a customized named entity recognizer called 'Malware Entity Extractor' (MEE). We then build a neural network to predict how pairs of 'malware entities' are related to each other. When we have predicted entity pairs and the relationship between them, we assert the 'entity-relationship set' in a CKG. Our next step in the process is to fuse similar entities, to improve our CKG. This fusion helps represent intelligence extracted from multiple documents and reports. The fused CKG has knowledge from multiple AARs, with relationships between entities extracted from separate reports. As a result of this fusion, a security analyst can execute queries and retrieve better answers on the fused CKG, than a knowledge graph with no fusion. We also showcase various reasoning capabilities that can be leveraged by a security analyst using our fused CKG.

INDEX TERMS Artificial intelligence, computer security, cyber threat intelligence, after action reports, knowledge graphs, semantic web.

I. INTRODUCTION

Every year thousands of malware are created and subsequently used to attack different organizations. In March 2018, Nuance Technologies announced that it lost 92 million US dollars to an attack caused by the *Notpetya* malware [6]. Around the same time, a group of hackers launched a series of cyber-attacks on multiple universities [48]. The attackers stole 31 terabytes of data and the total estimated financial loss stood at about three billion US dollars. Spear-phishing emails were used to steal login credentials and ex-filtrate sensitive data [48]. To combat these malware-based attacks, security researchers retrieve malware samples from the 'wild'. These samples are then 'detonated' in a controlled environment and its behavior 'logged' [26], [27]. Using this behavioral data, security analysts map various malware to known indicators and means of attacks. Analysts and forensic experts from companies like Mandiant are also involved in the aftermath of attacks, where they analyze what malware was used, the means and indicators of attack, and the results of the attack. As a result of these studies , these security analysts produce 'After Action Reports' (AARs), which describe in great detail a particular malware sample, its means and consequences. These technical AARs are vital source of Cyber Threat Intelligence (CTI) and can augment other Open Source Intelligence (OSINT) sources to create a holistic picture of an attack. Subsequently, these AARs are used to create and or modify the defensive approaches deployed to protect the infrastructure, and identify and prevent future attacks. They are also sometimes used to identify perpetrators and attribute these attacks to known hacker groups. AARs contain data that is found after an exhaustive investigation of an attack. If security analysts base their primary source of knowledge on AARs, they will be able to

The associate editor coordinating the review of this manuscript and approving it for publication was Ilsun You.

access a lot of relevant information about attacks which will not only help them to correctly identify the various aspects of an unknown attack, but also draw similarities between new attacks and previously known attacks. This ability to draw similarities between previously encountered attacks is greatly amplified if we have the knowledge from a number of AARs dealing with similar malware, or malware variants. In addition, the extracted knowledge can be used to automate defenses [35].

In this paper, we aim to mine knowledge from these AARs produced by security analysts. Going through a large number of cybersecurity blogs and AARs is not only cumbersome but also infeasible for a time constrained security professional. We extract cybersecurity knowledge from these reports, and populate it in a knowledge graph. This can be then used to assist the defensive systems and is queryable by analysts [35]. The cybersecurity domain is a niche area in terms of terms/language used. As such, compared to general natural language processing tasks, there is not a lot of data available for creating data-driven knowledge extraction approaches. Moreover, approaches trained on generally available corpus do not work well for cybersecurity related text. Thus, one significant contributon of our work is a cybersecurity textual corpus which helps build machine learning algorithms to extract information from textual data about cybersecurity. The cybersecurity corpus that we create contains processed AARs, along with other blogs and technical reports. (See Section III-A). We created a novel annotated dataset for cyber-knowledge mining. We curated 474 AARs and APT (Advanced Persistent Threat) Reports, 550 security blogs from Microsoft [28] and Adobe [1] , numerous descriptors from Common Vulnerabilities Enumerations (CVE) [30].

Our second major contribution is the creation of a set of tools, used in a pipeline, to extract cybersecurity knowledge from text sources. Our cybersecurity knowledge extraction and processing pipeline has 3 components, a Malware Entity Extractor (MEE), a Relationship Extractor (RelExt), and the final Cybersecurity Knowledge Graph (CKG). MEE has been trained over annotated pieces of cybersecurity text, to predict cybersecurity entities in an AAR. MEE is a cybersecurity Named Entity Recognizer (NER) (See Section III-B1). It has been built using Conditional Random Fields, Gibbs' sampling, and regular expressions. RelExt (See Section III-B2) is a novel deep learning based relationship extractor that has been trained over cybersecurity data, to predict a relationship between pairs of entities extracted by MEE. Once we have extracted the named entities we filter out pairs of entities that cannot have a relationship between them according to the schema of our CKG. We then pass the remaining pairs of entities to the relationship extractor. The relationship extractor takes two named entities ('A' and 'B') as input and tries to predict a particular relationship as output. We then populate our CKG (See Section III-C) with the entity relationship sets from the extracted data. The CKG represents the unstructured data present in the AAR into a structured ontology. Once we have an entity-relationship set from RelExt, we can populate the CKG with the triple ('Entity A'–has relationship 'R'–'Entity B').

To improve the quality of the CKG, we also fuse knowledge from different AARs that describe the same entity. This fusion greatly helps security analysts aggregate different pieces of cybersecurity knowledge from multiple sources and understand how those pieces are related. A security analyst can then use this fused knowledge to make better-informed cybersecurity decisions. This helps her to query a single *fused* knowledge graph, and retrieve entities from different sources. This also helps her to discover previously unknown relationships. This fusion of knowledge from different AARs is one of the main contributions of this work. We demonstrate this capability in Section III-D.

Our reasoning and search capabilities are further improved because of our CKG fusion. Fusion of knowledge from different AARs helps a security analyst to retrieve better results for her queries. This emphasises the need for our fusion step to improve the underlying CKG. We demonstrate this capability by executing the same query on the fused CKG, and also on the knowledge graph before fusion. Some example queries can be found in Section IV-D.

Our system can be used by security analysts to look-up cyber intelligence data, and also ascertain similarities between a new cyber-incident and some previously encountered cyber-incidents already asserted in our CKG. A security analyst can use a SPARQL [46] endpoint to query over the graph and ask the system to compute answers to complex queries and scenarios (See Section IV-D).

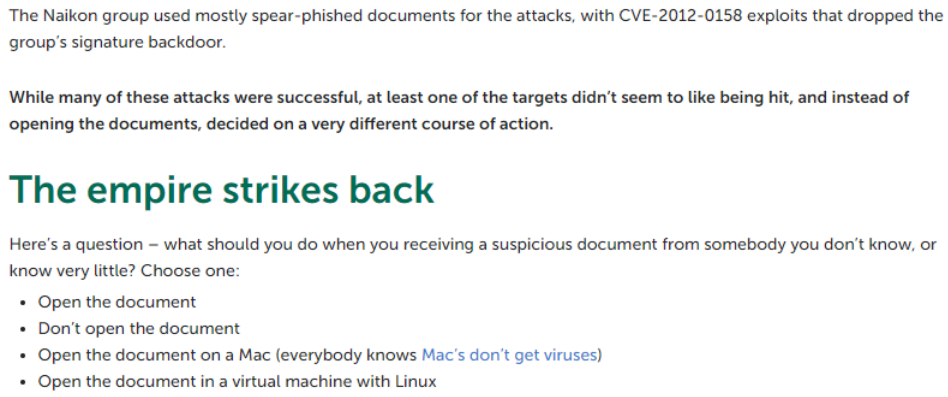
We organize the paper as follows - Section II, talks in detail about AARs and similar research conducted by other researchers. Section III, talks in detail about the architecture of our system and how we form the pipeline. Section IV, talks about our findings from our experiment and the overall quality of the individual components of our pipeline. Section V, talks about the scope of further research in this area.

II. BACKGROUND AND RELATED WORK

In this section we describe AARs and talk about similar research conducted in this area.

A. AFTER ACTION REPORTS (AAR) AND OPEN SOURCE INTELLIGENCE (OSINT)

An AARs contain details about a particular exercise presented in a fixed format as mandated by the United States Department of Homeland Security [44]. The format to share exercises and evaluations has been standardized by the Homeland Security Exercise and Evaluation Program, to make intelligence information sharing consistent across jurisdictions. In the domain of cybersecurity, an AAR contains various details about an exercise to detect a cyber-attack, and information about the means to mitigate the cyber-attack. Apart from government agencies, security companies also publish AARs. Some examples of such companies are 'Kaspersky' [20] and 'FireEye' [9]. These AARs include detailed analysis of various cyber-attacks. An AAR about a particular


The Naikon group used mostly spear-phished documents for the attacks, with CVE-2012-0158 exploits that dropped the group's signature backdoor.

While many of these attacks were successful, at least one of the targets didn't seem to like being hit, and instead of opening the documents, decided on a very different course of action.

The empire strikes back

Here's a question – what should you do when you receiving a suspicious document from somebody you don't know, or know very little? Choose one:

- Open the document
- Don't open the document
- Open the document on a Mac (everybody knows Mac's don't get viruses)
- Open the document in a virtual machine with Linux

FIGURE 1. An excerpt from an after action report describing Naikon and Hellsing malware attacks.

'malware','threat-actor', or a 'campaign' includes the means to identify the attack from 'indicators' which are nothing but 'fingerprints' left at the attack site by the agent carrying out the attack. It may also talk about what 'vulnerabilities' were targeted by the attacker agent, and also what a user, or a potential victim, can do to prevent the attack.

An AAR about a cyber-attack provides security analysts with knowledge, extracted and captured in a mandated format, so that they are able to use this information to analyze future attacks. AARs, also make a reliable source of information to mine intelligence, because they are published on official security agency websites. This 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.* [32]–[34], where the authenticity of the information, cannot be verified. Another reason why we choose AAR to be a source of our knowledge is that these are open source. Security companies release full length reports on their websites, available for users to download and read.

After Action Reports (AARs) are different from technical blogs and include more technical details about malware behaviour. An AAR gives security analysts technical details and knowledge about a cyber incident. Cybersecurity blogs, differ from AARs and do not capture ample technical information. Many of these blogs are intended for a general audience, and only have superficial information about an attack. Another factor that is important, in the context of our source of knowledge, is that there might be blogs coming from individuals who may be unknown and do not possess the required technical skills. Compared to those reports and blogs, AARs are released by known cyber-security researchers and organizations. They also have in depth technical details which might be a useful source of knowledge for security analysts.

B. CYBERSECURITY KNOWLEDGE GRAPHS

A knowledge graph is a set of semantic triples, which are pairs of 'entities' with 'relationships' between them. Cybersecurity Knowledge Graphs (CKGs) have long been used to represent Cyber Threat Intelligence (CTI). To represent CTI in a CKG, the first step is to identify what entities and relationships need to be asserted. We also use an ontology called 'Unified Cybersecurity Ontology' (UCO) [45] to provide our system with cybersecurity domain knowledge. UCO is based on Structured Threat Intelligence Language (STIX 2.0) [12] which provides a schema to represent cyber-threat intelligence. CKGs have also been developed from other open-source information by Mittal *et al.* [32], [33]. CKGs and knowledge graphs have also been used to create various analyst augmentation systems [18], [19], [21], [36], [40], [42]. Next, we discuss the 3 main components of our system- a named entity recognizer, a relationship extractor, and a system to compare the various malware nodes in the CKG.

1) NAMED ENTITY RECOGNITION

Named Entity Recognition (NER) for cyber-threat information extraction have been built using Conditional Random Fields (CRF), Support Vector Machines [7], and neural networks [5]. Ekbal *et al.* in their paper [7] have proposed a language independent algorithm for detecting named entities. Recently, Bi-directional LSTMs are being used to recognize named entities. Even in the field of CTI, entity extraction has been done with the help of deep learning [4], [11]. Some of the approaches, in the field of CTI, have also demonstrated the use of neural networks on hand-picked features which yielded better results [24], [41]. Despite the widespread applications of Long Short Term Memory (LSTM), the use of CRF-based classifiers for entity extraction have continued to remain state of the art [43]. Using BiDirectional LSTMs, aids in the process of capturing (or forgetting) long term context, which is necessary to predict an entity class. However, some of the entities that we are interested in, like 'filenames', 'IPAddresses', or 'hashes', do not need contextual information to be predicted into the correct class. Capturing diverse context for these classes may even be detrimental for the task of entity classification. Moreover, BiLSTMs overfit with limited data. Thus, we build our own entity extractor for cybersecurity text based on CRFs and regular expressions (See Section III-B1).

2) RELATIONSHIP EXTRACTOR

Relationship extraction predicts the links or the relationships existing between pairs of entities extracted by our system. There has been significant work done in the field of relation extraction between entities. Relationship mapping can be many-to-many, many-to-one, or one-to-one. TransH models [47] have worked on extracting many-to-many mapping by shifting vector spaces in hyperplanes. TransE models [3] have used head and tail entities to predict one-to-one mapping. Sparse vectors have also been used [15] to predict relationship mapping, which was an improvement over TransH and TransE models. We use the Relationship Extraction method proposed by Pingle *et al.* [39], which is our previous work in this area. The Relationship Extraction algorithm uses vector encodings of individual entities created using word2vec [29]. These vector representations help capture the context of the cybersecurity entities in text, which aids in the task of relationship prediction.

3) UTILIZING CKG FOR MALWARE COMPARISON

There has been significant research done in the area of comparing malwares. Some of these approaches use machine learning after extracting features about these malwares [22]. One interesting approach generates graphs on instruction traces of target executables [2]. The authors subsequently used machine learning algorithms to classify various softwares as benign or malicious. Other graph-based approaches include building behavioral graphs of malwares, based on system calls as demonstrated by Park *et al.* [38]. After constructing these graphs, the authors propose a method of subgraph matching to calculate similarity between malwares. However, this approach would encapsulate only the system behaviour at the site of attack. It does not necessarily capture broader details like, what is the target software, or if this malware is a part of a bigger campaign. It will not capture, the attack pattern of the malware in natural language. The CKG that we are building has technical details, as well as broader details about campaigns launched, tools used, softwares targeted, etc. Thus we can use the CKG, extract triples about a malware, and compare it with the triples about another malware and simply compare them, to cluster similar malwares. Jiang *et al.* in their paper [16], have proposed a method of recreating the semantic view of the host machine in a virtual machine and running malware analysis in the VM to tackle the problem of malware hiding from detection software. The paper demonstrates how, by recreating the semantic view of the host machine in the VM, it is possible to identify 'self-hiding' malware by using file comparisons. Although it is possible to detect newer varieties of malware using the semantic view, it does not give us higher level details about the malware. The semantic view that has been recreated in the paper is focused on system specific metrics. Some examples of these metrics are processes, memory, files. Our CKG based on STIX, captures not only system specific information, like filenames and hashes, it is also able to capture a wide range of details which aids security researchers. For example, by performing analysis in an infected machine, we may be able to gather the information that a file is suspiciously trying to connect to a remote IP Address and is trying to transmit some information. AARs, having already analyzed this may be able to assert that this is due to the 'command and control infrastructure' that the malware is using. Since we gather our information from AARs, we identify these keywords and populate the CKG. So a security researcher can easily search for all malwares using 'command and control' infrastructure in our CKG and will be presented with appropriate results.

III. METHODOLOGY

Our proposed pipeline takes as input, an AAR as shown in Figure 2. The trained Malware Entity Extractor (MEE), extracts entities from the AAR, and sends the extracted set of entities to the Relationship Extractor (RelExt). The trained relationship extractor then predicts the best relationship that should exist between a pair of entities. The output of the relationship extractor is the entity-relationship set which is then used by the CKG module to assert it into a knowledge graph.

If two AARs talk about the same malware, the entity-relationship sets are fused together. The fused output of the system is then asserted in the CKG. The main components of our system are:

- ***MEE***: A Malware Entity Extractor which has been trained over annotated pieces of cybersecurity text, to predict cybersecurity entities in an AAR.
- ***RelExt***: A relationship extractor which has been trained over cybersecurity data, to predict a relationship between pairs of entities extracted by MEE.
- ***CKG***: Cybersecurity Knowledge Graphs which are populated with the entity relationship sets of the extracted data. The CKG represents the unstructured data present in the AAR into a structured ontology.

A. AFTER ACTION REPORT CORPUS

The dataset for our system includes AARs from various sources, including cybersecurity companies mentioned in Section II-A. We have curated a total of 474 reports, each of which contains details about a cyber-attack. The total size of our dataset is about 1 GB. All of our curated reports used in our pipeline, are in English. Since, our MEE and the vector embeddings for the entities used in RelExt, were trained in a corpus of cybersecurity text in English, we only curated reports which were consistent with the language of training those components. Our corpus contains detailed analysis reports from cybersecurity companies mentioned in Section II-A. We also have notes on 'Advanced Persistent Threats' which include reports from government agencies, like 'Intelligence Research Team'. The reports were in PDF format and were converted to raw text for processing. Some of the information embedded in the PDF files are in the form

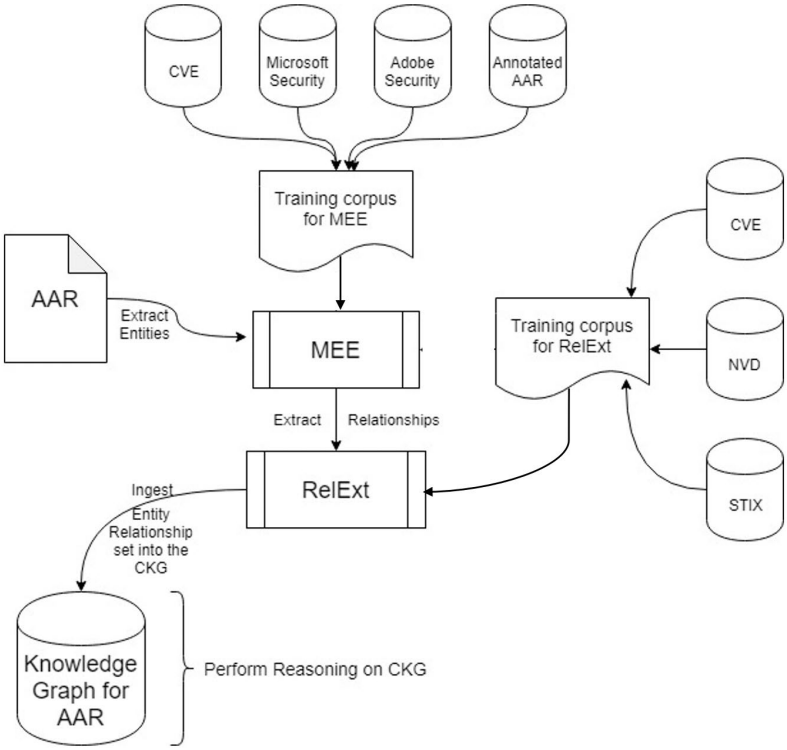


FIGURE 2. System Architecture including MEE, RelExt, and CKG.

of images. Detecting the information present in those images is out of scope for our pipeline. We extract the raw text from the Portable Document Format (PDF) files, using industry standard tools and assert into a CKG.

B. CYBERSECURITY KNOWLEDGE EXTRACTION FROM AFTER ACTION REPORTS

Syed *et al.* in their paper about the Unified Cybersecurity Ontology [45] proposed a schema to represent cyber-threat intelligence. UCO 1.0 was based on STIX 1.2 [13], which is a standard to share cyber-threat information. UCO 2.0 was based on STIX 2.0 [12], and it has refined the previous version of STIX [39]. We build the schema of our CKG based on the classes and relationships specified in UCO 2.0.

Some important classes present in UCO 2.0 which have been used in our CKG, along with some additional new classes to better represent cyber-threat intelligence from an AAR are:

- *Software*: An entity that relates to a piece of code usually used as tool such as Office or Adobe.
- *Exploit-Target*: An entity that relates to the site of the attack usually targeted by a malware such as Android or an operating system like Windows.
- *Malware*: An entity that refers to malicious code and/or software which is inserted into a system.
- *Indicator*: An entity that contains a pattern which helps the administrator to indicate an ongoing attack or malicious activity.
- *Vulnerability*: An entity that refers to a patch of bug or weakness that could be exploited by ill-intended users.
- *Course-of-action*: An entity that refers an action or set of actions that either prevents or responds to an attack.
- *Tool*: An entity that refers to legitimate software that can be used by threat actors for malicious activities.
- *Attack-pattern*: An entity that refers to steps that could result in an active attack on an individual or group of users.
- *Campaign*: An entity that refers to grouping of activities that could lead to a malicious attack.
- *Filename*: A file which is used by the malicious software to execute the attack
- *Hash*: A SHA-256 hash of an executable which may be used to identify an attack
- *IP Addresses*: An IP Address or addresses which a malicious software may be using in the course of the attack

In our previous work about CKG improvement, we have defined the classes and the relationships, necessary for the schema of our CKG. [39] Next, we discuss the 3 pipeline systems- MEE, RelExt, and CKG in detail.

1) MALWARE ENTITY EXTRACTOR (MEE)

After compiling the AAR corpus, we pass each report to the Malware Entity Extractor (MEE). The MEE has been trained separately, with data from different sources, like, Common Vulnerability and Exposures (CVE), security blogs, STIX datasets, AARs, dark web posts, etc. The MEE is trained in

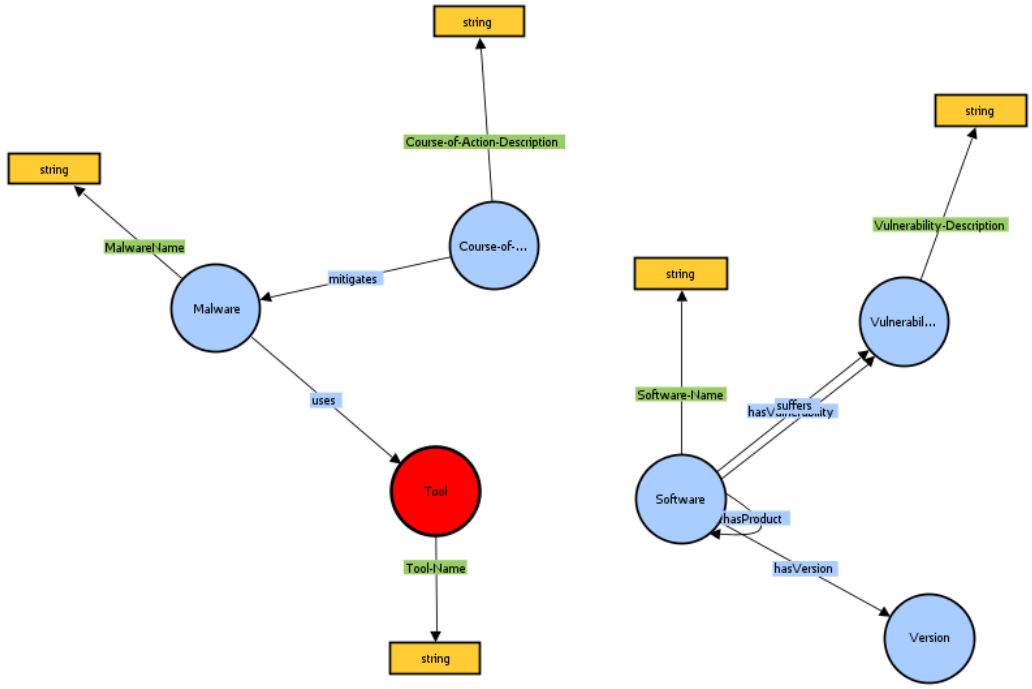


FIGURE 3. Some of the classes and the relationships as shown by the VOWL visualizer.

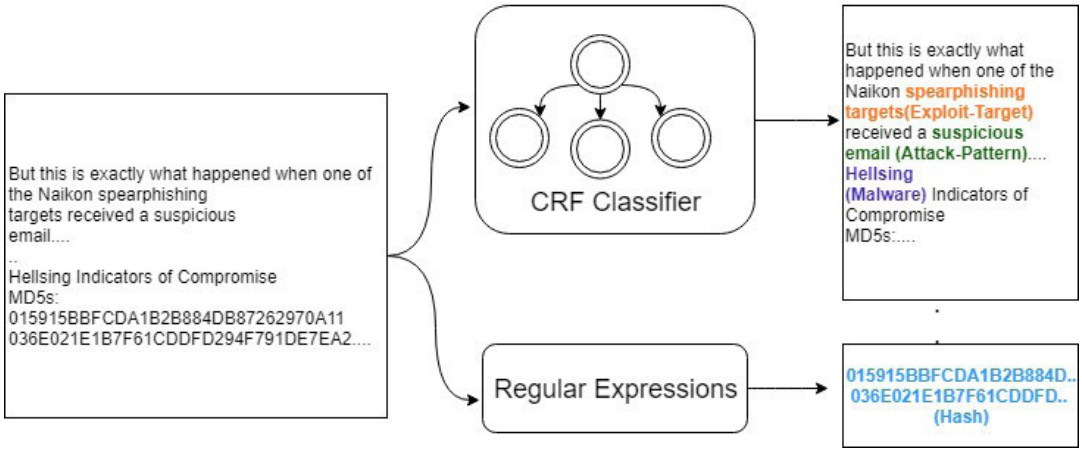


FIGURE 4. Components of the Malware Entity Extractor (MEE) and its performance on an AAR excerpt.

such a way, that given a piece of text, it will predict an entity class as output for every word in that piece of text. MEE aims to detect the type of knowledge present in the AAR. Our knowledge graph schema is based on Unified Cybersecurity Ontology (UCO) 2.0 [45].

MEE is based on the Stanford NER [8], it takes as input, the context of a particular word, for which it has been tasked to find the label, and then tries to match possible labels it can find for a word with a similar context in the training set. It also factors in the structure of the sentence. From the way MEE has been trained, it is clear that the model performs better as a class predictor if it has seen that particular word before. For AARs, we add sentences in the training set, which gets appended to the training corpus. The corpus also has sentences from many other cybersecurity sources and not just AARs. We discuss the performance of the MEE in subsequent sections, the model performs well although it has not encountered much of the text from an AAR in its training set. This basically means that our MEE can perform well on unseen data, which is critical as we want our MEE to perform well on new AARs. We use these named entities to assert the entities and their classes in the Cybersecurity Knowledge Graph (CKG). Our MEE model has two sub-components- A *regular expression model* to extract file names, hashes and IP Addresses from the files and a *Conditional Random Field* (CRF) component to identify various named entities and

classes (See Section III-B). Next , we describe these two sub-components.

a: REGULAR EXPRESSIONS

The first component uses regular expressions to find pieces of text, which correspond to classes which do not need a context to be identified, some of these are-

- *Filename*
- *Hash*
- *IP Address*

We form simple regular expressions which help us detecting IP Addresses :

\.(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])

and a regular expression we to detect file hashes :

[0-9a-f] {5,40} \[0-9a-f]{10,64}

b: CONDITIONAL RANDOM FIELDS

We use a conditional random field (CRF) named entity recognizer (NER) system from CoreNLP [25] to identify the malware entities, which requires contextual information. We used 6 ,184 sentences, which contained about 53,829 words from various sources, including:

- Microsoft Security Bulletins [28]
- Adobe Security Blogs [1]
- CVE descriptors [30]
- Annotated After Action Reports (AARs)

We annotated a portion of them to form a training set. We describe the various splits for training in Section IV-A. Manning *et al.* in their paper [8] have described how the use of Gibbs' sampling instead of the traditional Viterbi encoding facilitates capturing of non-local information. CRF is a probabilistic graphical model, which encapsulates bidirectional flow of information. In order to form Markov chains, for Gibbs' sampling to work, the CRF model converts the graphical model into linear chains. This helps in capturing non-local information, and ensuring label consistency. The word, 'Adobe Reader', used in a sentence may not always have ample contextual information for our model to infer that it is a 'Software'. But with the help of a CRF model augmented by Gibbs' sampling, even if there was non-local evidence to suggest that 'Adobe Reader' is a 'Software', our MEE is successfully able to detect the correct class of the word. We discuss the performance of MEE and its components in Section IV.

2) RELATIONSHIP EXTRACTOR (RelExt)

We take pairs of malware entities captured by our MEE and pass it to the next stage of our pipeline, which establishes a relationship existing between a given pair of entities. The RelExt is essentially a neural network that takes two vectors representing two entities extracted by our MEE and predicts a relationship between them. We train a Word2vec [29] model separately and the extracted entities are passed to the model to generate their respective embeddings. The dimension of the embeddings is 200. The RelExt neural network takes a pair of embeddings as input. It has ab input layer, 3 hidden layers, and a softmax layer as output. The dimension of the input layer is 400 and the hidden layers have dimensions 200, 100, and 50 respectively. The output softmax layer has a dimension of 6.

Before we send entity pairs to RelExt, we perform post-processing of candidate entity pairs based on the pre-defined schema of our CKG. We only pass the entity pairs, which can have a credible relationship between them, and we discard the pairs of entities which, according to the schema of our CKG, cannot have any relationship. For example, there can be no direct relationship between a 'Software' and a 'Filename', according to our CKG, so we do not pass entity-pairs which are of the type 'Software' and 'Filename' as candidates to our next stage. Since we have already defined a schema for our CKG, we automatically filter out pairs of entities which do not have credible relationships between them. This component of our pipeline, is a neural network, which takes in two entities as input and then predicts a particular relationship which exists between the two entities.

The first stage for relationship extraction is to represent the input entities in a vector form, which is suitable for neural networks to perform matrix multiplication. The output of the MEE (See Section III-B1), merely produces a class type, or entity type, for a particular word, which is not sufficient for neural networks. So we take the corpus we used for training the MEE, and generate word2vec embeddings [29] for each word. Word2Vec is a technique to represent words present in a corpus. Word2vec algorithm captures the context of each word in the corpus, and represents each word with a vector of a specified length. We train word2vec on a corpus of cybersecurity text from NVD [37], CVE [30], and STIX data from TAXII servers [39], to represent each entity captured by our MEE.

The second part of the relationship extractor is to train the neural network with the help of a training set made from NVD, CVE and STIX datasets. We have labeled pairs of entities and the relationship which exists between them. Our dataset consists of a total of 33,000 labeled relationships. We split the dataset into a training set and validation set. We use different split ratios: 80 (train), 20 (validation); 70 (train), 30 (validation); 90 (train), 10 (validation). Table 3, and Table 4, contain the experimental results of RelExt on the validation set. After we have a trained model that is capable of producing a vector output for any input word, we take the entities predicted from the AARs and generate the embeddings for various entities. We then train the neural network model RelExt to produce the output class of relationship that exists between the pairs of embeddings. We evaluate our Relext model in Section IV-B.

Sample output for the RelExt component, here the neural network produces the relationship between cyber entities -

- 'Hellsing'(Malware)-*uses*-'suspicious email'
- 'Hellsing' (Malware)- *uses* - '015915BB..' (Hash)

This component's output is an Entity-Relationship Set, which we will use in the next stage to populate the CKG.

C. CYBERSECURITY KNOWLEDGE GRAPH ASSERTION

After we generate the entity-relationship set, which is the output of the relationship extraction component of our system, we are able to assert this data to our Cybersecurity Knowledge Graph. We base our knowledge graph schema on STIX (Structured Threat Intelligence) [12] and use UCO 2.0 [45] to provide cybersecurity domain knowledge to the system. The output of the relation extraction is a set of semantic triples and the types of the individual entities. Entity classes, relations, and the specific classes which act as a domain or range of each relation describe the schema of a knowledge graph. The following is a description of the schema for our CKG. Each relations in our CKG is mentioned in the following list, along with its domain and range.

- *attributedTo*: Domain:*Malware* or *Tool* or *Vulnerability*. Range:*Campaign*.
- *indicates*: Domain:*Indicator*. Range:*Malware* or *Tool*.
- *hasProduct*: Domain:*Software*. Range:*Software*
- *mitigates*: Domain:*Course-of-Action*. Range:*Malware* or *Tool* or *Vulnerability*.
- *hasVulnerability*: Domain:*Software* or *Exploit-Target*. Range:*Vulnerability*.
- *uses*: Domain:*Malware* or *Tool* or *Attack-Pattern*. Range:*Malware* or *Tool* or *Vulnerability*.

D. CYBERSECURITY KNOWLEDGE GRAPH (CKG) FUSION

In our dataset, we encounter multiple AARs describing the same attacks or malwares. Ideally, we would like to fuse knowledge extracted from different AARs describing the same malware to create a more robust CKG.

If an AAR entity has been already asserted in our CKG and an exact match is available, we simply declare a '*owl:SameAs*' assertion and fuse the graph entities. Here we declare that the two entities nodes and subgraphs are the same. If there is no exact match for a newly discovered AAR entity, we calculate the term frequency–inverse document frequency (TF-IDF) scores to calculate similarity between the current document and previously processed AARs. We also calculate string similarity between new entities and entities which are already asserted using various 'edit-distance' metrics. If there is a close match, we *fuse* them. Fusion helps us discover knowledge about a malware or a cyber-entity which is found in multiple AAR sources. This makes our CKG more robust, with more information about cyber-entities ingested. Here is an example query on the 'unfused' CKG.

```
SELECT ?x WHERE {
 ?x a CKG:Malware;
    CKG:uses
     CKG:588f41bbc117346355113f.}
```

The above query returns:

```
Hellsing
```

The same query on the 'fused' CKG returns:

```
SELECT ?x WHERE {
 ?x a FusedCKG:Malware;
    FusedCKG:uses
     FusedCKG:588f41bbc117346355113f.}
```

The above query returns:

```
Hellsing, XWeber
```

In figure 5 we see how the *Hellsing* malware and its attributes are identified and asserted in the CKG. We can see that 'Hellsing' 'uses' different files like 'cmd.exe', 'test.exe', and 'xKat.exe'. This was created using the entity relationship sets from two reports about the same malware. One of them mentioned the filename 'xkat.exe' and the other report mentioned a filename called 'xKat.exe'. An edit-distance algorithm helped us calculate the similarity between the two entities and we used an 'owl:SameAs' relation to assert that those two individuals were identical. All relationships held by one of the nodes will hold for the other as a result of the 'owl:SameAs' assertion. We also see other entities identified. For example, '7zip archive' is a 'Tool' which is used by the Hellsing malware.

IV. EXPERIMENTS AND RESULTS

To evaluate our entire system, we ran experiments on various individual components. The first two components in our system are MEE and RelExt. Since both these components are classification tasks we use standard classification scoring metrics like precision, recall, and F-1 scores.

A. MEE EVALUATION

The corpus for our Malware Entity Extractor (MEE) comprises of security bulletins from Adobe [1] and Microsoft [28], CVE descriptions [30], and annotated After Action Reports. We performed evaluations on two types of datasets:

- *On a split of the entire corpus*. This evaluation set contains a mixed bag of sentences from all the sources mentioned. There is no overlap between the training set and the test set. We perform a 10 fold evaluation and averaged the results to produce precision, recall, and F-1 scores. The entire dataset consisted of about 3600 sentences. We used 720 sentences (20%) for testing and the rest for training. In Table 1, we can see the precision,recall, and f-1 scores for each individual class that is present in the test set.
- *Specifically on a set of sentences from After Action Reports*. We annotate some more sentences from various

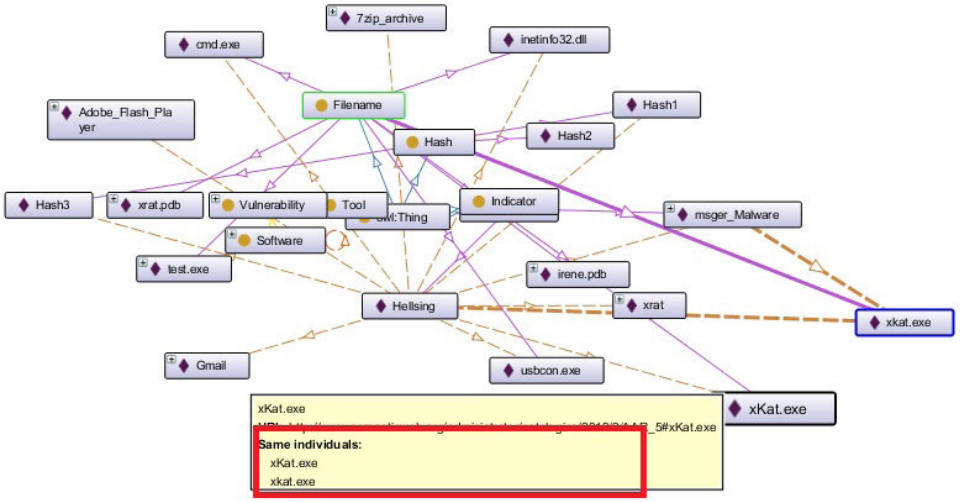


FIGURE 5. Hellsing malware entities in the Cybersecurity Knowledge Graph.

TABLE 1. Precision, Recall, and F-1 score for Entity Classes evaluated across the test sets of the corpus.

Entity Classes	Precision	Recall	F-1 Score
Attack-Pattern	77	63	69
Campaign	92	74	82
Course-of-Action	80	62	70
IP Addresses	100	88	93
Hashes	100	100	100
Exploit-Target	77	90	83
Filenames	80	90	85
Malware	83	83	83
Software	86	88	87
Tool	80	90	85
Vulnerability	73	53	62
Average	91	92	91

TABLE 2. Precision, Recall, and F-1 score for Entity Classes evaluated across the test sets of after action reports.

Entity Classes	Precision	Recall	F-1 Score
Attack-Pattern	85	52	65
Campaign	100	100	100
Course-of-Action	81	80	80
IP Addresses	100	100	100
Hashes	100	100	100
Exploit-Target	71	38	50
Filenames	85	80	83
Malware	83	83	83
Software	40	67	50
Tool	81	70	75
Vulnerability	80	80	80
Average	78	78	77

After Action Reports to form a test set specifically to evaluate MEE's performance on this type of data. The test set consisted of 50 annotated sentences chosen at random from After Action Reports. These were not present in the training set, which had the 3600 sentences. In Table 2, we can see the precision, recall, and F-1 scores of the classes which were present in the test set of the AAR corpus.

It should be noted here that the average scores for the test set of the AAR corpus, is slightly lower than the scores for the test set of the entire training corpus. This is because the sentences in blogs, CVE, Microsoft and Adobe Security are more structured and follows a fixed format. This allows the CRF based MEE to easily detect patterns in the sentences, as compared to detecting sentences from the AAR test set. The sentences in the AAR reports are much more complex and diverse, making entity detection for the MEE, more challenging.

B. RelExt EVALUATION

We evaluated the RelExt in accordance with the method stated by Pingle *et al.* [39]. The total dataset comprises of 90,000 annotated relationships from Common Weakness Enumeration [31], triples extracted from Trusted Automated eXchange of Indicator Information (TAXII) [14] servers, annotated triples from the same corpus of Microsoft Security Bulletins, Adobe Security Bulletins, and After Action Reports.

Although there is very limited prior work for relation extraction in the domain of cybersecurity [17] , it is difficult for us to compare our results with the results obtained by other approaches. This is because there is no common dataset for comparison, nor are there agreed upon relationships to extract. Since we define our own schema, we have a set of pre-defined relationship classes for the RelExt pipeline.

Jones *et al.* [17], discuss a relation extraction pipeline for cybersecurity. However, the dataset that they have chosen is comprised of NVD [37] feeds and CVE [30] data. However, these data feeds are simple and they have an underlying structure present in them. This is not the case for our dataset, which not only comprises of NVD and CVE feeds but also sentences from long technical reports which are much more complex, making it more difficult for basic NLP techniques to successfully perform relation extraction on our dataset.

TABLE 3. Precision, Recall, and F-1 score for relationship classes.

Relationship Classes	Precision	Recall	F-1 Score
hasProduct	49	97	65
hasVulnerability	92	74	82
uses	100	88	93
indicates	80	90	85
mitigates	55	70	62
related-to	92	74	66

TABLE 4. Accuracy for various splits on training and testing data.

Splits	Accuracy
80-20	96.21%
70-30	91.88%
66-34	91.34%

In Table 3, we can see the precision, recall, and F-1 scores of each relationship class, averaged across all types of splits (80-20, 90-10, 70-30). In Table 4, we can see the accuracy scores of all classes on each type of split.

C. CKG EVALUATION

Evaluation metrics for knowledge graphs are an open challenge. Some of the related work that has been done in this area, concentrated on validating RDF triples, against supporting or contradicting knowledge found from various other sources [23]. Another work [10] describes the cost of knowledge graph evaluation. It talks about various sampling methods which can help us choose triples at random and verify the correctness of those triples. The paper calculates the cost of different sampling and evaluation methods, based on the number of samples we choose for evaluation. In our evaluation methodology, we use a fixed number of triples for verification, selected at random. It should be noted that the components of our system leading to the CKG (MEE and RelExt) have been evaluated separately. The evaluation scores of these components give us an idea about the correctness of the CKG triples. However, since the CKG is the final output of our pipeline, we want to separately evaluate the CKG. So, in order to evaluate the quality of the knowledge graph created, we use human evaluators. The evaluator asks questions in natural language. We translate these questions to CKG queries, and we retrieve the results of these queries and map them to the information a human can find, from a particular AAR document. This method of evaluation a knowledge graph using a domain expert human evaluator is considerably expensive and slow.

We made a list of about 25 queries and asked the evaluator what entities have been correctly identified by the system. An example of such query is as follows-

Is there any software that is mentioned which has some exposed vulnerabilities?
Descriptive Logic Query: Software, hasVulnerability some Vulnerability
Answer: Adobe Flash Player, WindowsOSX
SPARQL Query: SELECT ?x WHERE { ?x a Software;:hasVulnerability ?y. ?y a Vulnerability.}

TABLE 5. CKG evaluation.

Correctly identified	Incorrectly identified	Missed	Total
25(0.48)	9(0.17)	17(0.35)	52(1)

Table 5, shows the average evaluation results from the After Action Reports. Each of these queries produces a set of entities as result. We see how many of these entities correctly follow the ground-truth knowledge present in various AARs. 48% of correctly identified entities essentially means that 48% of all entities returned by our set of queries have captured correctly the knowledge presented in the report. Only 17% of the data is incorrect in our CKG.

D. REASONING

Our pipeline's objective is to capture data from AARs so that security engineers can use it without spending additional time reviewing these reports to extract the data manually. Once the CKG has been populated, we can leverage a robust query mechanism that is available to the end users to retrieve desired entities. The standard way of querying a CKG is through SPARQL [46] queries which we show in this section.

Suppose an end user wants to know what 'Tool' two malwares use in the fused CKG. The query for that in SPARQL would be as follows:

```
SELECT DISTINCT ?x ?y ?z where {
 ?x a FusedCKG:Malware;
    FusedCKG:uses ?z.
 ?z a FusedCKG:Tool.
 ?y a FusedCKG:Malware;
    FusedCKG:uses ?z.
 FILTER(?x != ?y). }
```

The above query returns:

```
'Dark Caracal' 'CrossRAT' 'Java'
'Hellsing' 'Trojan' 'Windows 7 Driver'
```

The query asks for distinct pairs of entities in the same namespace 'FusedCKG', which use the same 'Tool'. The response says both 'Dark Caracal' and 'CrossRAT' use 'Java', and 'Hellsing' and 'Trojan' use 'Windows 7 driver'.

Another example can be about a user who wants to focus on a particular filename. The user wants to list out all malwares which use a particular file 'test.exe'. The query that the user can use, is as follows:

```
SELECT ?x WHERE {
 ?x a FusedCKG:Malware;
    FusedCKG:uses FusedCKG:test.exe. }
```

The above query returns:

```
'Hellsing'
'XWeber'
```

The query asks for all entities of the type malware, in the namespace of 'FusedCKG' which 'uses' the entity 'test.exe' in the same namespace.

Several AARs may be released by different organizations describing a particular malware. Each AAR about a particular malware may be the result of some 'focused' analysis done on the malware. For example, we have multiple AARs about the spyware 'Pegasus'. Some of the AARs focus only on the behavior of the malware in Android operating system. On the other hand some reports are focusing on the behavior of the malware on all mobile operating systems. Fusing knowledge from different AARs helps us to aggregate knowledge acquired from individual AARs. If we run the following query on a CKG with knowledge from just one AAR about 'Pegasus':

```
SELECT ?x WHERE {
 CKG:Pegasus CKG:uses ?x.
 ?x a CKG:Tool. }
```

The above query running on a CKG with no fusion returns:

```
'SMS'
'WiFi'
```

The same query when ran on the fused CKG returns:

```
'SMS'
'WiFi'
'Camera'
'Keylogger'
```

Similarly, we can discover more knowledge about the Pegasus malware by merging knowledge from multiple reports:

```
SELECT ?x WHERE {
 CKG:Pegasus CKG:uses ?x.
 ?x a CKG:Exploit-Target. }
```

On a CKG without fusion this query returns

```
'Android'
```

The same query when run on a fused CKG returns

```
'Android'
'iOS'
```

V. CONCLUSION AND FUTURE WORK

We successfully created a pipeline of novel systems which automatically extracts cybersecurity entities from After Action Reports (AARs). The system identifies how each pair of those entities are related, and asserts them into a Cybersecurity Knowledge Graph (CKG). The pipeline fuses knowledge extracted from one AAR with other AARs that describe the same attack. Our CKG comes with reasoning capabilities, i.e, it helps end users execute queries and find similarities between different cyber-attacks. We trained our MEE and RelExt on cybersecurity text which is relatively shorter and has a fixed structure, like text from CVEs, Microsoft Security Bulletins, Adobe Security Bulletins. We showcased that by appending a small number of sentences from AARs to our corpus of other related cybersecurity text, we can create generalized MEE and RelExt. MEE and RelExt perform well on unseen text data from AARs, are able to extract entities, and establish correct relationships between them. The fact that our training set contains text data from a diverse range of sources, makes our pipeline robust. We also showcased that the results we get by executing the same queries on the fused CKG has more information, than the results we get on the same query executed on the unfused knowledge graph. This proves that the quality of the knowledge graph describing all AARs is improved by the process of fusion. This improvement can then be leveraged by a security analysts, who can use this information to better protect an organization. We have demonstrated that it is possible to extract information from AARs, without having to train our extractor models exclusively on these reports.

In the future, it will be possible to apply neural models to extract vector embeddings of entities in our CKG, and use it for malware attribution. We can also extend the schema of our ontology to capture more information about cyber-attacks, which would lead to more informed resolutions by our CKG reasoner.

REFERENCES

[1] Adobe. (May 2020). *Adobe Security Bulletin*. [Online]. Available: https://helpx.adobe.com/security.html

[2] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *J. Comput. Virol.*, vol. 7, no. 4, pp. 247–258, Nov. 2011.

[3] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, 2013, pp. 2787–2795.

[4] R. A. Bridges, C. L. Jones, M. D. Iannacone, K. M. Testa, and J. R. Goodall, "Automatic labeling for entity extraction in cyber security," 2013, *arXiv:1308.4941*. [Online]. Available: http://arxiv.org/abs/1308.4941

[5] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 357–370, Dec. 2016.

[6] Cyberscoop. (May 2018). *Notpetya Malware*. [Online]. Available: https://cyberscoop.com/nuance-communications-says-notpetya-ransomware-cost-it-98-million-since-june/

[7] A. Ekbal and S. Bandyopadhyay, "Named entity recognition using support vector machine: A language independent approach," *Int. J. Elect. Comput. Eng., World Acad. Sci., Eng. Technol.*, pp. 589–604, 2010.

[8] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by Gibbs sampling," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2005, pp. 363–370.

[9] FireEye. (May 2020). *Fireeye Reports*. [Online]. Available: https://usa.kaspersky.com/enterprise-security/apt-intelligence-reporting

[10] J. Gao, X. Li, Y. E. Xu, B. Sisman, X. L. Dong, and J. Yang, "Efficient knowledge graph accuracy evaluation," *Proc. VLDB Endowment*, vol. 12, no. 11, pp. 1679–1691, Jul. 2019.

[11] H. Gasmi, A. Bouras, and J. Laval, "LSTM recurrent neural networks for cybersecurity named entity recognition," in *Proc. 13th Int. Conf. Softw. Eng. Adv.*, 2018, p. 11.

[12] Oasis Group. (May 2013). *STIX 2.0 Documentation*. [Online]. Available: https://oasis-open.github.io/cti-documentation/stix/examples.html

[13] Oasis Group. (May 2018). *STIX 1.0 Documentation*. [Online]. Available: https://stixproject.github.io/documentation/

[14] Oasis Group. (May 2019). *TAXII*. [Online]. Available: https://oasis-open.github.io/cti-documentation/taxii/intro

[15] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1–7.

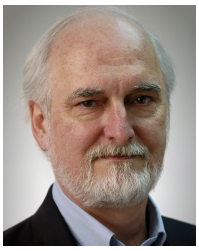
[16] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection through vmm-based 'out-of-the-box' semantic view reconstruction," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 128–138.

[17] C. L. Jones, R. A. Bridges, K. M. T. Huffer, and J. R. Goodall, "Towards a relation extraction framework for cyber-security concepts," in *Proc. 10th Annu. Cyber Inf. Secur. Res. Conf.*, 2015, pp. 1–4.
[18] K. P. Joshi, A. Gupta, S. Mittal, C. Pearce, and T. Finin, "ALDA: Cognitive assistant for legal document analytics," in *Proc. AAAI Fall Symp. Cognit. Assistance Government Public Sector Appl.*, 2016, pp. 1–4.
[19] M. Joshi, S. Mittal, K. P. Joshi, and T. Finin, "Semantically rich, oblivious access control using ABAC for secure cloud storage," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 142–149.
[20] Kaspersky. (May 2020). *Kaspersky Reports*. [Online]. Available: https://usa.kaspersky.com/enterprise-security/apt-intelligence-reporting
[21] N. Khurana, S. Mittal, A. Piplai, and A. Joshi, "Preventing poisoning attacks on AI based threat intelligence systems," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Oct. 2019, pp. 1–6.
[22] J. Liu, Y. Wang, and Y. Wang, "The similarity analysis of malicious software," in *Proc. IEEE 1st Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2016, pp. 161–168.
[23] S. Liu, M. d'Aquin, and E. Motta, "Measuring accuracy of triples in knowledge," in *Proc. 1st Int. Conf. Lang., Data, Knowl.* Cham, Switzerland: Springer, 2017, pp. 343–357.
[24] C. Ma, H. Zheng, P. Xie, C. Li, L. Li, and S. Luo, "Neural sequence labeling with linguistic features," in *Proc. 12th Int. Workshop Semantic Eval.*, 2018, pp. 707–711.
[25] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. Mcclosky, "The stanford corenlp natural language processing toolkit," in *Proc. ACL, Syst. Demonstrations*, 2014, pp. 55–60.
[26] A. McDole, M. Abdelsalam, M. Gupta, and S. Mittal, "Analyzing CNN based behavioural malware detection techniques on cloud IaaS," in *Proc. Int. Conf. Cloud Comput. (CLOUD)*, 2020, pp. 1–17.
[27] A. McDole, M. Abdelsalam, M. Gupta, S. Mittal, and M. Alazab, "Deep learning techniques for behavioural malware analysis in cloud IaaS," in *Malware Analysis using Artificial Intelligence and Deep Learning*. Springer, 2020.
[28] Microsoft. (May 2020). *Microsoft Security Bulletin*. [Online]. Available: https://msrc-blog.microsoft.com/tag/security-bulletin/
[29] T. Mikolov, I. S. K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, 2013, pp. 3111–3119.
[30] MITRE. (May 2020). *CVELIST Project*. [Online]. Available: https://github.com/CVEProject/cvelist
[31] MITRE. (May 2020). *CWE MITRE*. [Online]. Available: https://cwe.mitre.org/data/index.html
[32] S. Mittal, P. K. Das, V. Mulwad, A. Joshi, and T. Finin, "CyberTwitter: Using Twitter to generate alerts for cybersecurity threats and vulnerabilities," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2016, pp. 860–867.
[33] S. Mittal, A. Joshi, and T. Finin, "Thinking, fast and slow: Combining vector spaces and knowledge graphs," 2017, *arXiv:1708.03310*. [Online]. Available: http://arxiv.org/abs/1708.03310
[34] S. Mittal, A. Joshi, and T. Finin, "Cyber-all-intel: An AI for security related threat intelligence," 2019, *arXiv:1905.02895*. [Online]. Available: http://arxiv.org/abs/1905.02895
[35] S. N. Narayanan, A. Ganesan, K. Joshi, T. Oates, A. Joshi, and T. Finin, "Early detection of cybersecurity threats using collaborative cognition," in *Proc. IEEE 4th Int. Conf. Collaboration Internet Comput. (CIC)*, Oct. 2018, pp. 354–363.
[36] L. Neil, S. Mittal, and A. Joshi, "Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2018, pp. 7–12.
[37] NIST. (May 2020). *NVD*. [Online]. Available: https://nvd.nist.gov/
[38] Y. Park, D. Reeves, V. Mulukutla, and B. Sundaravel, "Fast malware classification by automated behavioral graph matching," in *Proc. 6th Annu. Workshop Cyber Secur. Inf. Intell. Res. (CSIIRW)*, 2010, pp. 1–4.
[39] A. Pingle, A. Piplai, S. Mittal, A. Joshi, J. Holt, and R. Zak, "RelExt: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2019, pp. 879–886.
[40] A. Piplai, S. Mittal, M. Abdelsalam, M. Gupta, A. Joshi, and T. Finin, "Knowledge enrichment by fusing representations for malware threat intelligence and behavior," in *Proc. IEEE Int. Conf. Intell. Secur. Inform.*, 2020.
[41] R. Manikandan, K. Madgula, and S. Saha, "Cybersecurity text analysis using convolutional neural network and conditional random fields," in *Proc. 12th Int. Workshop Semantic Eval.*, 2018, pp. 868–873.
[42] P. Ranade, S. Mittal, A. Joshi, and K. Joshi, "Using deep neural networks to translate multi-lingual threat intelligence," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2018, pp. 238–243.
[43] S. R. Vadapalli, G. Hsieh, and K. S. Nauer, "Twitterosint: Automated cybersecurity threat intelligence collection and analysis using Twitter data," in *Proc. Int. Conf. Secur. Manage.*, 2018, pp. 220–226.
[44] Homeland Security. (May 2020). *After Action Report Defintion*. [Online]. Available: https://emilms.fema.gov/IS130a/groups/57.html
[45] Z. Syed, A. Padia, T. Finin, L. Mathews, and A. Joshi, "UCO: A unified cybersecurity ontology," in *Proc. AAAI Workshop Artif. Intell. Cyber Secur.*, 2016, pp. 1–8.
[46] W3. (May 2020). *Sparql Query Language*. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/
[47] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1112–1119.
[48] Wired. (Jul. 2018). *Worst Cybersecurity Breaches*. [Online]. Available: https://www.wired.com/story/2018-worst-hacks-so-far/

ARITRAN PIPLAI received the B.E. degree in computer science from Jadavpur University, India. He is currently pursuing the Ph.D. degree with the University of Maryland at Baltimore County (UMBC). He works with the Ebiquity Laboratory, Department of Computer Science, UMBC. His main research interest is the application of artificial intelligence to the field of cybersecurity.

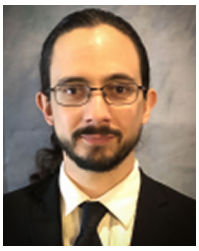
SUDIP MITTAL (Member, IEEE) received the B.Tech. and M.Tech. degrees in computer science from IIIT Delhi and the Ph.D. degree in computer science from the University of Maryland at Baltimore County (UMBC). He worked with the Accelerating Cognitive Cyber Security Research Laboratory (ACCL), Ebiquity Research Lab, and the Center for Hybrid Multicore Productivity Research (CHMPR). He is currently an Assistant Professor of computer science with the University of North Carolina at Wilmington (UNCW). His primary research interests are cybersecurity and artificial intelligence. His goal is to develop the next generation of cyber defense systems that help protect various organizations and people.

ANUPAM JOSHI (Fellow, IEEE) received the B.Tech. degree from IIT Delhi in 1989, and the M.S. and Ph.D. degrees from Purdue University, in 1991 and 1993, respectively. He is currently the Oros Family Professor and the Chair of the Computer Science and Electrical Engineering Department, University of Maryland at Baltimore County (UMBC). He is also the Director of the Center for Cybersecurity, UMBC. His research interests are in the broad area of networked computing and intelligent systems. His primary focus has been on data management and security/privacy in mobile/pervasive computing environments, and policy-driven approaches to security and privacy. He has published over 250 technical articles, filed and been granted several patents, and has obtained research support from NSF, NASA, DARPA, the US DoD, NIST, IBM, Microsoft, Qualcom, Northrop Grumman, and Lockheed Martin.



TIM FININ received the degrees from MIT and the University of Illinois at Urbana–Champaign. He is currently the Willard and Lillian Hackerman Chair of Engineering and a Professor of computer science and electrical engineering with the University of Maryland at Baltimore County (UMBC). He has over 40 years of experience in applying AI to problems in information systems and language understanding. He has held positions at UMBC, Unisys, the University of Pennsylvania, and the MIT AI Laboratory. His current research interests include knowledge graphs, analyzing and extracting information from text, and cybersecurity. He is a fellow of ACM and AAAI. He was a recipient of the IEEE Technical Achievement Award.

JAMES HOLT is currently a Researcher with the Laboratory for Physical Sciences, where he is focused on applying artificial intelligence and machine learning techniques to address cybersecurity problems.



RICHARD ZAK currently works as a Senior Lead Technologist with Booz Allen Hamilton and Laboratory of Physical Sciences. His research work involves using machine-learning algorithms address cybersecurity problems, with a focus on malware.