Cyber Threat Intelligence for SOC Analysts

Nidhi Rastogi
Rochester Institute of Technology, NY
nidhi.rastogi@rit.edu

Md Tanvirul Alam
Rochester Institute of Technology, NY
ma8235@rit.edu

***Abstract*—Cyber threat intelligence (CTI) has been valuable to SOC analysts investigating emerging and known threats and attacks. However, the reach is still limited, and the adoption could be higher. While CTI has consistently proven to be a rich source of threat indicators and patterns collected by peer security researchers, other researchers have occasionally found them helpful. Challenges include intelligence in the CTI documented in an unstructured format, embedded in a large amount of text, making it challenging to integrate them effectively with existing threat intelligence analysis tools for internal system logs. In this paper, we detail ongoing research in threat intelligence extraction, integration, and analysis at different levels of granularity from unstructured threat analysis reports. We share ongoing challenges and provide recommendations to overcome them.**

I. Introduction

Cyber threats represent a continuously evolving set of potential attacks and attackers on computers, networks, and connected devices [24]. Detection and mitigation of sophisticated cyber threats require knowledge of a system's threat environment (to determine what attacks *might* occur) and its system state (to determine whether an attack *has* occurred). To provide situational awareness, known as Cyber Threat Intelligence (CTI)– knowledge about cyber attacks and emerging threats from various external sources is distributed to defenders [25], [38]. Unfortunately, the volume of cyber threat knowledge can overwhelm the best SOC analysts. Additionally, the non-uniform and informal structure of cyber threat knowledge feeds makes it infeasible to fully automate processing. System defenders lack the time and resources to detect and mitigate every possible attack. These limitations mean potential attacks, or ongoing threats, must be triaged, understood, and either acted upon or ignored. Worse, specific cyber threat knowledge feeds may or may not be relevant in the context of the target system. Context incorporates current and potential future system states, the associated risks, and policies that express which states are not allowed.

This research will contribute to a new technological paradigm for individualized threat intelligence based on contextual data to manage threat knowledge even when data sources are sparse. This research improves the ability of defenders to protect their systems by building semantic knowledge-driven threat intelligence models that incorporate domain knowledge and relevant contextual information. This work will also result in faster attack detection and mitigation and better allocation of defensive resources. Toward these goals, this project is exploring the following two research directions:

1) Enable the inclusion of domain security knowledge catering to different context vs. data relevance needs. A key challenge is to combine external abstract threat knowledge with internal, domain-specific knowledge.
2) Enable robust adaptation of time and trust varying, dynamically changing cyber threat knowledge in domain-specific information sources. A key challenge is to extract meaningful information with confidence.
3) Overcome the lack of evaluation mechanisms for context. A key challenge is combining ML's precision and accuracy to the ranked inferences of knowledge graphs applicable to given organizational policies and trustworthiness.

This paper discusses ongoing research capturing the first goal and leaves the other two for future work.

II. Motivation

A paradigm shift has been observed in understanding and defending against evolving and complex cyber threats, from predominantly reactive detection of attacks to proactive prediction [40]. Research shows that signature and CTI knowledge sources have historically provided data on individual threat indicators [7], [9], [14]. This approach misses the context that can help organizations understand how to put those indicators together to identify a real attack or what action to take. Cyberthreat knowledge needs to be extracted to analyze cyber threats at scale. In the past decade, we have witnessed unprecedented growth of publicly available knowledge sources such as websites, forums, and social media, as well as non-public sources such as the Dark Web [19], [23], where attackers congregate in underground discussion forums and chat rooms. CTI knowledge sources can provide a stream of IoC and analysis, often driving reactive cyber security strategies. On the other hand, SOC analysts synthesize and contextualize comprehensive threat intelligence to provide reasoned and data-driven predictions, identify underlying trends or motivations, and deliver deeper insights that are more relevant to each organization.

To elaborate further, we describe the CTI of malware Cerberus– a trojan horse that targets Android mobile phone

Workshop on Security Operation Center Operations and Construction (WOSOC) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-90-8
https://dx.doi.org/10.14722/wosoc.2023.23014
www.ndss-symposium.org

banking credentials. We use the same example throughout the paper for consistency. The Cerberus CTI describes how the malware works and how to recognize its tactics, techniques, and procedures (called TTPs or attack patterns). The CTI also covers vulnerabilities of specific business technologies, such as email, domains, and mobile devices. Consider the following snippet from the CTI on Cerberus written by ThreatPost [1]: *"...A malicious Android app has been uncovered on the Google Play app marketplace that is distributing the banking Trojan,* Cerberus*. The app has 10,000 downloads. Researchers said that the trojan was found within the last few days, as it was being spread via a Spanish currency converter app (called "Calculadora de Moneda"), which has been available to Android users in Spain. Once executed, the malware has the capabilities to steal victims' bank-account credentials and bypass security measures, including two-factor authentication (2FA)..."*

In Figure 1, we show the transition from an unstructured threat analysis report to a structured sub-knowledge graph for "Cerberus". The entity extraction (e.g. application), triple generation ⟨malware, targets, application⟩, and knowledge graph construction (all triples combined) are detailed in a later section. It isA *"banking Trojan"*, targets *"Spain, class:Location"*, *"Android, class:OS"* devices, and uses attack patterns such as *"Bypass security measures"*, and *"Steal victim's bank account credentials"*.

The class definitions for entities– Malware, Attack Pattern, Location, OS, Application, are mapped to existing threat intelligence ontology classes [11], [33] but modified for CTI and also follow the STIX2.1 framework. Relationships between them are defined by the same ontologies.

Our research proposes to extract triples from open-access CTI sources to learn and train a threat intelligence model using past malware information, including attack patterns. The triple formed for attack patterns is: $\langle \text{Cerberus}, uses, AttackPattern\rangle$. For example, $\langle \text{Cerberus}, uses, \text{“}Steal\ bank\ account\ credentials\text{”}\rangle$. Our approach also maps the attack pattern in natural language to the MITRE ATT&CK ID: T1636.

Benefits for SOC analysts: Automating the process of attack pattern extraction can assist in threat hunting and protection against APT campaigns. However, the same APT campaign may manifest in different forms in different settings due to various factors, such as differences in operating systems, targeted applications, variations of the threat, and so on. As a result, relying solely on IoCs for exact threat hunting is unreliable since attackers can modify them to evade detection [19]. To address this issue, it is important to use a combination of techniques for threat hunting and protection against APT campaigns. In addition to using IoCs, it is also important to use behavioral and anomaly-based detection techniques. These techniques involve monitoring network traffic, system activity, and user behavior to detect abnormal activity that may indicate an APT campaign. Our research focuses on the behavior technique detection aspect, which we describe in the next section.

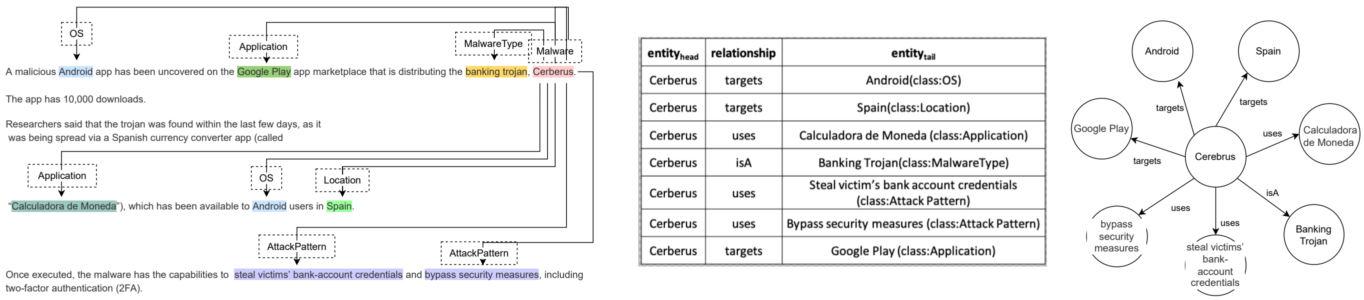
III. Approach Overview

The main idea of the proposed framework, LADDER, is to use past threat intelligence to train a threat intelligence model that can detect semantically similar attack patterns in threat analysis reports and infer attack tactics or techniques for emerging attack vectors. LADDER analyzes various malware attack patterns and maps them to the MITRE ATT&CK pattern framework. Although the low-level details of each step are detailed, the high-level definition of all attack patterns can be abstracted. The ultimate goal of the framework is to use past attack knowledge to improve threat detection and response in the future. LADDER not only uses past attack patterns to improve threat detection but also captures additional information such as locations, IoCs, and system information. This additional information helps to identify the similarity between malware and threat actors, which enhances the effectiveness of the threat intelligence model. By combining multiple sources of information, the model becomes more robust and can better identify emerging threats and potential attacks. This approach enables security analysts to detect and respond to new threats quickly and efficiently, reducing the risk of successful attacks.

LADDER comprises four modules for using threat intelligence from CTI to provide actionable analysis to SOC analysts. The first module aggregates unstructured threat and attack information from diverse CTI sources. The second module uses a knowledge graph-based approach to structure and analyze threat intelligence. The third module trains a threat intelligence model using the structured and analyzed CTI. Finally, the fourth module provides inferred intelligence to SOC analysts. Once the system is trained and ready, a security analyst submits a corpus of unstructured threat information into the framework. The trained system returns inferred intelligence, but instead of simply classifying the ingested data, it returns classes and matching instances. The predefined classes are Malware, Malware Type, Application, Operating System, Organization, Person, Time, Threat Actor, Location, and Attack Pattern. Nine of the ten classes return a list of classes and their respective instances found in the CTI. This approach enables security analysts to quickly and easily identify and understand potential threats and attacks by providing a comprehensive and structured view of the threat landscape.

A. Cyberthreat Intelligence (CTI)

CTI is typically available as after-action reports [30], unstructured blogs, security bulletins [2], and technical reports. To examine CTI reports, a standard approach involves perusing the most prevalent techniques, trends, and threats observed by the author(s), detecting, mitigating, and simulating specific threats and techniques, and mapping ideas, guidance, and priorities to their security measures and general strategy. SOC analysts can use IoCs to track down and defend against known attacks. However, it's also important to extract information like tactics, techniques, and procedures (TTPs) written in a semantic format to characterize a threat attack for analysis and future predictions.



$entity_{head}$	relationship	$entity_{tail}$
Cerberus	targets	Android(class:OS)
Cerberus	targets	Spain(class:Location)
Cerberus	uses	Calculadora de Moneda (class:Application)
Cerberus	isA	Banking Trojan(class:MalwareType)
Cerberus	uses	Steal victim's bank account credentials (class:Attack Pattern)
Cerberus	uses	Bypass security measures (class:Attack Pattern)
Cerberus	targets	Google Play (class:Application)

Fig. 1. For malware Cerberus, (a) Annotated CTI using BRAT, (b) Triples created from the annotated snippet, (c) Knowledge graph from the triples.

B. Concepts of threat intelligence

We extract threat intelligence concepts in the form of a triple $\langle entity_{head} - relationship - entity_{tail} \rangle$, also known as a semantic triple. This triple encodes a statement in a threat report about the malware in an RDF format $\langle subject, predicate, object \rangle$ expression. The entity in the triple represents one of the ten classes from CTI reports – Malware, Malware Type, Application, Operating System, Organization, Person, Time, Threat Actor, Location, Attack Pattern. These triples aggregate in a graph called the threat intelligence knowledge graph, which transforms the multiple data structured, open-access CTI into a structured format. SOC analysts can use this graph to systematically query known threats and emerging threats and forecast trends and attack patterns using evidence from existing knowledge in the graph.

C. Generating Threat Intelligence Graph

Open-access CTI reports are ingested into information extraction and machine learning models to extract relevant entities and assert the relationship between entities to generate a graph. These information extraction models have been pre-trained on triples collected from hand-annotated CTI reports. This approach allows us to construct a knowledge graph from unstructured text sources rather than relying on pre-existing databases constructed by domain experts.

D. Inferring Attack Patterns

The information extraction module also comprises an algorithm we propose that automates attack pattern extraction from CTI reports. As demonstrated in Figure 1(b), our algorithm extracts all known malware attack behaviors and guides us to a list of previously unseen attack behaviors. The triple formed from extracting a new attack behavior is $\langle Cerberus, uses, conduct\ covert\ surveillance \rangle$.

The end-to-end process of inferring attack patterns using our approach works in the following manner: (a) **Training:** Our algorithm and other information extraction models extract attack behaviors, classes, and relationships of all known, existing, and dormant malware, forming a knowledge graph. This includes data on all known and existing malware. Additionally, our algorithm also maps these behaviors to MITRE ATT&CK pattern IDs. (b) **Inference:** After training is complete, the model is ready to infer IoCs and attack patterns of emerging threats, which can be for known and unknown malware. A SOC analyst can query the knowledge graph by inputting a triple with a missing tail entity. Our framework infers all blank tail entities, including attack behaviors and all other classes, along with a confidence score. (c) **Validation:** In this step, the inferred attack patterns and IoCs are compared to those identified by human analysts from CTI.

IV. System Design

A. Dataset Collection

We collected a meticulously curated set of 150 threat reports from the MITRE website. These reports feature 36 Android malware such as *Cerberus, Rotexy, SpyNote RAT*. Attack patterns from many of these reports exist and map to the MITRE ATT&CK framework, alibi are paraphrased. Using the BRAT annotation tool [39], we manually annotated different threat concepts and their relationships, as explained in Section III-B.

B. Information Extraction from CTI for Threat Intelligence Graph

Extracting information from CTI to create a threat intelligence graph [32] involves several steps, such as identifying relevant sources of CTI, including open-source intelligence (OSINT), closed-source intelligence (CSINT), social media, dark web forums, and threat intelligence platforms (TIPs). After determining the most valuable sources for data collection, the following steps involve data normalization, relationship creation between entities [4], and visualizations to create the threat intelligence graph.

Crawling Open-access CTI sources: To ensure high quality and reliability, we scrape threat reports only from security companies, technology companies, and technology news reporting companies. Many threat reports contain their publication date within the first few sentences of the report, typically after the title. Using this information, we prioritize threat reports published after the year 2010, as this information can help with tracking the progression of threats over time and identifying patterns or trends in threat activity.

Extracting and Labeling concepts (entities): After preprocessing and cleaning threat information collected from multiple sources, we use state-of-the-art natural language processing techniques to extract entities. We use attention mechanism [27] models like the *tner* [42] to focus on specific concepts such as threat patterns and malware information and capture the relationships between different entities. For further fine-tuning, we used the *tner* library to help improve the accuracy of the entity extraction process, as the library contains pre-trained models that have already learned to recognize common security concepts and their relationships. In contrast, some cybersecurity concepts like URL, IP address, email, and file name are extracted using pattern matching techniques [31], [46], which involves searching for specific patterns or regular expressions within the CTI text. In addition to this approach, we use a heuristic-based method to extract different indicators, including URL, IP address, hash, and CVE ID, to automate the entity extraction process further.

C. Attack Pattern Extraction

Extracting attack patterns from CTI differs significantly from other entities as they often consist of larger blocks of contiguous text and do not represent a single *named entity*. Instead, they describe a sequence of actions a cyber threat takes. Our proposed approach consists of three sub-tasks: relevant sentence extraction, attack phrase extraction, and mapping attack patterns to the MITRE ATT&CK framework, and we describe them next.

(a) Relevant sentence extraction. The first sub-task involves identifying sentences that describe one or more attack patterns, such as *"exploit," "command and control,"* or *"credential harvesting."* We formulate the problem as a binary sentence classification task where the sentences containing one or more attack patterns are labeled positive while the rest are negative. During training, the positive sentences comprise all the annotated sentences containing attack patterns. An equal number of negative instances are randomly sampled from the remaining sentences to create a balanced dataset. To perform this classification task, we use a pre-trained transformer model, RoBERTa, fine-tuned on a large corpus of text. We add a hidden layer before the linear layer consisting of two neurons for the classification task to improve the model's performance. Once trained, the model classifies new sentences as either positive or negative and extracts relevant sentences containing attack patterns from CTI.

(b) Attack phase extraction. For the sentences predicted as positive (i.e., comprising at least one attack pattern), the second sub-task involves identifying only the relevant part(s) of the sentence that contains the description. To achieve this, we use a sequence tagging model similar to those used for extracting other entities for this task. The sequence tagging model is trained on a hand-annotated labeled dataset, where each word in the sentence is annotated as either part of an attack pattern description or not. We can train the sequence tagging model on a labeled dataset, where each word in the sentence is annotated as either part of an attack pattern description or not. In this case, we have two classes: whether or not a word is part of an attack pattern description, and the model can identify all the words that belong to each attack pattern, even if the words are not contiguous. For example, consider the sentence:

"The malware can covertly send and steal SMS codes, open tailored overlays for various online banks, and steal 2FA-codes".

The model identifies three distinct attack patterns: "send and steal SMS codes," "open tailored overlays for various online banks," and "steal 2FA codes."

(c) Mapping Attack Patterns. The final sub-task maps each attack pattern to standardized MITRE ATT&CK techniques. MITRE ATT&CK provides a comprehensive framework comprising techniques and sub-techniques for understanding and categorizing cyber threats. However, we do not consider mapping to sub-techniques in this work as mapping each extracted sequence to its corresponding sub-technique MITRE ID requires significant annotation effort, as the number of classes is significantly large. For mapping to the techniques, we adopt a semantic similarity-based approach for the mapping task. We compute the embeddings for extracted attack pattern phrases using a pre-trained sentence transformer model [34]. Then, we compute the embeddings of both the title and description of the MITRE ATT&CK ID described on the MITRE website. We use a cosine similarity metric to find the closest match between the extracted attack pattern and the pre-computed embeddings of the MITRE ATT&CK IDs. Once found, the embedding maps to the corresponding MITRE ATT&CK technique.

D. Adding Relationship to Concepts

To incorporate entity information for relation classification, we introduce four tokens that capture the position information of the entities. Let s be the input sentence, e_1, and e_2 be the two entities in s for which we want to classify the relation. We denote the starting and ending position of e_1 and e_2 in s as $(s^{\text{start}}_{e_1}, s^{\text{end}}_{e_1})$ and $(s^{\text{start}}_{e_2}, s^{\text{end}}_{e_2})$, respectively. The input sentence is:

$$[CLS][E1]s_{e1}[E1][SEP]s_{between}[SEP][E2]s_{e2}[E2]$$

where, *[CLS], [SEP]* are special tokens used in pre-trained transformer models; *[E1], [E2]* are special tokens to mark the starting and ending position of e_1 and e_2, respectively, and s_{between} is the text between the two entities, which may contain relevant contextual information for relation extraction.

A pre-trained transformer-based relation classification model takes sentences from CTI reports in this format as input data and predicts the relation between e_1 and e_2 through a linear layer. Our relation extraction approach only considers pairs of entities that may have a valid relationship based on the adopted ontology [11], [33]. For instance, we may establish a relationship between a pair of entities of type Malware and Application (e.g., $\langle Malware, targets, Application\rangle$) but not consider a relationship between entities of type Application and Time. Our work builds on [45], which incorporates entity information for relation classification.

E. Querying the framework

We adopt a triple format for queries to enable SOC analysts to query the threat intelligence graph. This querying process is akin to knowledge graph link prediction, which involves predicting the best candidate for a missing entity in a knowledge graph [35]. Our framework uses a link prediction approach to infer the missing entity. We leverage the knowledge generated

TABLE I
RESULTS FOR NER USING DIFFERENT TRANSFORMERS (BOLD INDICATES THE BEST RESULT)

Model	Precision	Recall	F1-score
BERT-base	73.34	77.88	75.14
BERT-large	75.30	79.23	77.12
RoBERTa-base	41.55	41.01	40.84
RoBERTa-large	35.95	36.23	35.49
XLM-RoBERTa-base	75.32	79.06	76.98
XLM-RoBERTa-large	**76.97**	**81.57**	**78.98**

TABLE II
ENTITY EXTRACTION RESULT FOR DIFFERENT CLASSES USING XLM-ROBERTA-LARGE MODEL

Class	Precision	Recall	F1-score
Malware	78.45	83.08	80.70
MalwareType	65.64	87.18	74.89
Application	70.13	73.26	71.66
OS	89.95	96.24	92.99
Organization	73.68	74.12	73.90
Person	88.24	75.00	81.08
ThreatActor	58.33	37.84	45.90
Time	85.51	89.39	87.41
Location	93.55	89.92	91.70
Average	76.97	81.57	78.98

from the modules in our framework to learn a scoring function for predicting the missing triple. The tensor- factorization-based model, TuckER [6], infers the missing entity in the threat intelligence graph.

V. EXPERIMENTS AND RESULTS

For the large-scale knowledge graph, we extract concepts from open CTI using the modules described in our framework. The experimental choices, setup, and results are described below.

A. *Information Extraction*

We use the PyTorch [28] deep learning framework to implement the models for the information extraction and subsequent tasks. We fine-tune the pretrained transformer models available in Huggingface's transformers library [44]. We use a sequence length of 128 for training the NER models. We train the models for 20 epochs using 32 samples per mini-batch. We use a learning rate of 1e-5 for the base models and 1e-6 for the large models and optimize the models using the AdamW optimizer [21].

We split the annotated datasets based on the malware under discussion so that the same malware-related report is not present in two different splits. Out of 36 total malware, the training dataset contains reports for 26; validation and the test dataset contain reports for five malware each. The number of documents in the train, validation, and test split are 104, 21, and 25, respectively.

TABLE III
RESULT FOR THE RELEVANT SENTENCE EXTRACTION SUBTASK FOR ATTACK PATTERN EXTRACTION

Model	Precision	Recall	F1-score
BERT-base	86.50	85.20	85.84
BERT-large	86.06	85.80	85.93
RoBERTa-base	87.42	86.10	86.76
RoBERT-large	**89.22**	**90.03**	**89.62**
XLM-RoBERTa-base	83.00	88.52	85.67
XLM-RoBERTa-large	84.73	88.82	86.73

TABLE IV
RESULT FOR THE ATTACK PHRASE EXTRACTION SUBTASK FOR ATTACK PATTERN EXTRACTION

Model	Precision	Recall	F1-score
BERT-base	87.67	90.55	89.09
BERT-large	87.74	87.81	87.78
RoBERTa-base	88.53	90.12	89.32
RoBERT-large	**89.19**	**92.14**	**90.64**
XLM-RoBERTa-base	86.82	90.72	88.73
XLM-RoBERTa-large	88.55	91.77	90.13

1) Entity Extraction: We show the results for the entity extraction task using different transformer models in Table I. XLM-RoBERTa large model performs the best for this task with an average F1 score of 78.98%. We show the class-specific results for this model in Table II. We generally obtain better results for classes with more samples in the training dataset. Operating System and Location yield better results than the other classes as they do not have much variation in forms. Application and Organization have some overlap between them (e.g., Facebook and Twitter can be both Application and Organization) which reduces the performance. The most challenging class appears to be the ThreatActor. This is because ThreatActor is usually an Organization or Person with malicious intent. So, this requires a thorough understanding of the context to detect them properly. Having a limited input length means this may not be possible to infer from a single sentence. There are also not enough samples for this class in the training data, further reducing the performance.

2) Attack Pattern Extraction: We use the same malware split as in the NER task for attack pattern extraction. There are usually more sentences in a report that do not contain an attack pattern description than those that do. So, we randomly sample the same amount of negative sentences as the number of sentences that include attack patterns to create a balanced dataset for the sentence classification task. We fine-tune the transformer models for sentence classification and attack phrase extraction subtasks. We use a sequence length of 256 to train these models. We use a mini-batch size of 32, and the optimal learning rate from [1e-5, 5e-5, 1e-6], optimized using the validation set. We train the sentence classification models for 20 epochs and the attack phrase extraction models for 30 epochs. The models are trained using

TABLE V
RESULT FOR RELATIONSHIP EXTRACTION

Model	Precision	Recall	F1-score
BERT-base	93.75	92.22	92.60
BERT-large	**93.78**	**92.46**	**92.62**
RoBERTa-base	81.66	83.88	82.27
RoBERT-large	77.47	81.06	77.97
XLM-RoBERTa-base	81.77	83.86	81.74
XLM-RoBERTa-large	72.64	78.69	75.29

TABLE VI
CLASS-SPECIFIC RESULTS FOR RELATIONSHIP EXTRACTION

Class	Precision	Recall	F1-score
noRelation	100.0	100.0	100.0
isA	98.6	97.3	98.0
targets	96.3	88.1	92.0
uses	46.2	33.3	38.7
hasAuthor	87.5	93.3	90.3
has	100.0	70.0	82.4
variantOf	100.0	46.2	63.2
hasAlias	26.3	71.4	38.5
indicates	75.9	97.6	85.4
discoveredIn	100.0	100.0	100.0
exploits	100.0	50.0	66.7

the Adam optimizer [18].

We show the result for the binary sentence classification task in Table III. We get greater than 85% F1-score for all the models with RoBERTa large performing best with an average F1-score of 89.62%. The results for the attack phrase extraction task are in Table in IV. Here also, we obtain the best result using the RoBERTa-large model, with an average F1-score of 90.64%. These results suggest that our algorithm can detect the relevant sentences for attack patterns from the text and extract the part of the sentence that mentions the attack patterns with a high degree of accuracy.

To learn the optimal parameter values w_t and tau, We manually map 80 randomly selected attack pattern descriptions annotated in our training corpus with their corresponding MITRE ID. The optimum values for the two parameters were 0.4 and 0.6, respectively.

B. Relation Extraction

When extracting relationships from threat reports, we need to consider every pair of entities and infer the relationship between them. However, many pairs of entities may not have any meaningful relationship in the context, even if they may be valid according to the ontology. To identify such cases, we add the class *NoRelation* that predicts that there is no relation between the entities under consideration. We randomly sample such plausible entities from the annotated documents that do not have any annotated relation. We perform an 80:20 split for training and inference. Since there may be a relation between entities far apart in the document, we use a larger sequence length of 512. Due to GPU memory constraints, we use a mini-batch size of 16 for the large models and 8 for the smaller models. We use the optimal learning rate from [1e-5, 5e-5, 1e-6] and train the models for ten epochs using the AdamW optimizer.

We show the aggregate result for relation extraction in Table V. The BERT-based model outperforms the other two models for this task. The best result is obtained using the BERT-large model with a 92.62% average F1-score. We show the results for specific classes in Table VI. We do not include attack patterns for the relation extraction task as they are part of a single relation type (*Malware uses AttackPattern*). So, we can infer the relation for attack pattern once we identify the malware under discussion. Although *discoveredIn* performed well for the annotated corpus, it was primarily because the reports were cleaned and usually did not contain redundant time information of malware discovery. Among other classes, *uses* and *hasAlias* performs much worse. Context is challenging with $uses$ as it is usually confused with *targets* since both contain the same type of head-tail entity pairs (e.g., Malware and Application. Relationship between two malware (*variantOf* and *hasAlias*) is difficult to detect since they may be expressed at a distant position in report. We note that these results are obtained on the manually cleaned test dataset, and the performance drops on more noisy texts.

To the best of our knowledge, there is no open-source dataset for cybersecurity relation extraction. An early work in [15] used semi-supervised learning with a bootstrapping algorithm for extracting the relation between security entities. They achieved an average F1-score of 82% on a dataset containing 8 different relations. The work in [29] used a word embedding model for relation extraction in cybersecurity texts. They considered six different types of relations – hasProduct, hasVulnerability, uses, indicates, mitigates, related-to. They achieved an average F1-score of 92%, which is comparable to ours. Another work in [13] introduced a relation extraction model using the pre-trained BERT model and bidirectional GRU and CRF and achieved an average F1-score of 80.98%. Recent work on open information extraction [36], i.e., where the set of relations is not predetermined, achieved an average F1-score of 59.4%.

C. Threat Intelligence Knowledge Graph

The trained information and relation extraction models allow us to generate triples from new threat reports. We combine prediction from the best-performing NER model and heuristics to extract the concepts. We perform some post-processing to remove noisy entities extracted by the approach. For Malware and ThreatActors, we do not include them if they are predicted as a different class elsewhere or only mentioned once. For example, organizations like *ThreatFabric* are sometimes detected as *ThreatActor* instead of *Organization*. We do not use the entity node for relation extraction in such cases. Next, we use the best relation extraction model to identify the relationship between entity pairs following the ontology. We extract and map attack patterns to their corresponding MITRE IDs using our proposed algorithm. We identify the malware being discussed in each report (if any) and include the relationship with that malware and the attack patterns. We

TABLE VII
INFERENCE (LINK PREDICTION) RESULTS FOR DIFFERENT TRAINING AND TEST DATASETS

KG	TestSet 1				TestSet 2			
	Hits@3	Hits@10	Hits@30	MRR	Hits@3	Hits@10	Hits@30	MRR
KG1	0.209	0.365	0.497	0.186	0.090	0.195	0.322	0.093
KG2	0.221	0.353	0.516	0.211	0.215	0.359	0.501	0.203
KG3	0.202	0.353	0.507	0.190	0.204	0.356	0.498	0.193

TABLE VIII
CLASS-SPECIFIC INFERENCE (LINK PREDICTION) RESULTS.

Class	**KG**	**TestSet 1**				**TestSet 2**			
		Hits@3	Hits@10	Hits@30	MRR	Hits@3	Hits@10	Hits@30	MRR
AttackPattern	KG1	0.354	0.634	0.847	0.314	0.212	0.441	0.657	0.210
	KG2	0.444	0.700	0.940	0.420	0.453	0.694	0.936	0.415
Location	KG1	0.042	0.096	0.205	0.048	0.018	0.033	0.096	0.024
	KG2	0.036	0.096	0.247	0.044	0.018	0.092	0.225	0.034
Application	KG1	0.100	0.165	0.230	0.086	0.032	0.094	0.178	0.040
	KG2	0.026	0.083	0.126	0.030	0.040	0.102	0.129	0.034

do this by identifying the malware with the most frequent mention in the report.

We also include triples from Virus Total (VT) to create a larger knowledge graph to evaluate link prediction performance in the presence of more IoCs. VT provides academic API to access intelligence on a database of several million malware samples. We use the VT API V3.0 [3] to extract IoCs (contacted domains and IP addresses) and permission requests for each malware hash collected from a larger corpus of OpenCTI reports. We map each permission to attack pattern and MITRE ID using our algorithm and evaluate attack pattern prediction from querying the larger knowledge graph.

D. Inferring entities with TuckER

We create two different test sets from the hand-annotated documents with a varying number of triples for the link prediction task. We considered triples involving Malware e.g., AttackPattern, Location, Application, Organization for testing. The first test set (TestSet 1) consists of 25% of the annotated triples and the second (TestSet 2) consists of 40% of the triples. We experiment with three training knowledge graphs for the link prediction task. The first one (KG1) consists of the remaining triples in hand-annotated documents. The second one (KG2) consists of the triples generated using our proposed framework from 12k documents. We also extract triples from the VirusTotal and incorporate them with KG2 to obtain the last knowledge graph (KG3). We train the link prediction models using TuckER to predict the tail entities. We train the models with 50 embedding dimensions with a mini-batch size of 64. The models are trained for 1000 iterations with an initial learning rate of 0.001.

We use Mean Rank, Mean Reciprocal Rank (MRR), and Hits@n for evaluation. These are calculated from the ranks of all true test triples which TuckER returns. Mean reciprocal rank (MRR) is the average of the inverse of the ranks of all the true test triples. Hits@n denotes the percentage of test-set ranking where a true triple is ranked within the top n positions of the ranking. A higher score is considered better.

We show the results for the inference (link prediction) task in Table VII. For TestSet 1, we have similar performance using the different training datasets for all Hits@(n) values. However, for TestSet 2, KG1 performs much worse than KG2. We have a difference of 16.4% @Hits 10 between KG1 and KG2. This suggests that the additional triples obtained from a larger knowledge graph enable the model to make better predictions. The performance for KG2 is similar on the two test datasets suggesting better generalization ability. Adding additional triple from VirusTotal in KG3 decreases performance slightly compared to KG2, suggesting that the inclusion of additional IoCs may not help with the inference (link prediction) task. This is most likely because IoCs are usually unique across different malware.

We show class-specific prediction result in Table VIII using KG1 and KG2 for three different tail entities - AttackPattern, Location and Application where the head entity is Malware. Here the most promising result is obtained for predicting AttackPattern. The primary reason is that we have 66 unique attack patterns, but the number of possible tail entities is much larger for relations involving other classes. This result suggests that malware with similar properties may exhibit similar attack patterns. Similar to the aggregate result above, we see no significant drop in performance for the two test datasets for KG2 as opposed to KG1, where there is a significant drop.

VI. RELATED WORK

This research is closely related to three areas that support threat intelligence, especially attack pattern inference using open-access CTI, information extraction from unstructured and semi-structured corpus, and prediction.

CTI: Prior research has centered around building rules or models for searching and analyzing IoCs. Most shared intelligence has been limited to tracking known threat indicators such as IP addresses, domain names, and file hashes [20], [25], [38]. [5], [8], [37] use internal enterprise logs to capture threat intelligence and produce attack patterns, which is not scalable. Therefore none of these can be directly compared to our framework and inference-generating threat intelligence knowledge graphs. The limitation of this approach is that there is little to no overlap in the shared information, and no long-term analysis is possible using this intelligence. Also, none of this research captures detailed attack patterns in a standardized format nor utilizes CTI reports for analysis. In [41], Thein et al. propose a neural network approach for classifying sentences of a document into five phases of the cyber kill chain, which are broad categories for all the phases of a cyberattack. Similarly, [14] combines dependency parser and heuristics to extract threat actions from a document and map them to kill chain phases. Luo et al. [22] formulate the event extraction as a sequence tagging task and use a bidirectional LSTM network to solve it.

Discovering and Extracting Threat concepts: [10] implement a maximum entropy model for automatically labeling cybersecurity concepts. However, more recently, named entity recognition (NER) models have been built to adopt a hybrid approach combining multiple methods for NER [46]. Kim et al. [17] built a NER system using a deep bidirectional

LSTM-CRF network trained on a combination of features. The work in [12] provides a comparative study of different neural networks, including CNN, LSTM, BERT, and CRF, for cybersecurity NER. OpenCTI reports come from diverse sources, and therefore ambiguity and duplication in threat concepts are research-worthy problems. Word embeddings have been used for entity disambiguation in [43] by clustering entities based on their similarity in the embedding space.

Threat Intelligence Knowledge Graph: Knowledge graph construction in the cybersecurity domain is limited compared to other domains, such as biomedical studies. The study in [26] implements a collaborative framework with the help of semantically rich knowledge representation for the early detection of cybersecurity threats. This system assimilates ontologically defined concepts from multiple sources, such as security bulletins, CVEs, and blogs, and then represents it as a knowledge graph (KG) connected by these concepts. A cybersecurity KG is constructed from open-access CTI in [30], which contains an analysis of various cyber-attacks and is prepared from the investigation of attacks. This system consists of a custom NER and an entity fusion technique to merge concepts extracted from multiple reports. SEPSES [16] is a cybersecurity KG populated from multiple heterogeneous cybersecurity data sources and frequently updated. An Extraction, Transformation, and Loading (ETL) periodically checks and updates the KG as new security information becomes available.

VII. Conclusion

Our ongoing research presents a framework to infer attack patterns and other threat intelligence related to emerging threats. By inferring attack pattern steps and other classes such as Location, Application, OS, our framework provides valuable evidence to security analysts when dealing with new attacks. The proposed framework can be utilized by SOC analysts to proactively gain insights into potential ways that an emerging threat, described in open-access CTI, can affect their organization's internal network. We discuss the challenges of extracting threat information from CTI and models that have effectively analyzed cyber threat datasets.

References

[1] Cerberus unleashed. threatpost.com/cerberus-banking-trojan-unleashed-google-play/157218/.

[2] Msft security bulletin. https://msrc.microsoft.com/update-guide/.

[3] Virustotal. https://developers.virustotal.com/reference/overview.

[4] Md Tanvirul Alam, Dipkamal Bhusal, Youngja Park, and Nidhi Rastogi. Cyner: A python library for cybersecurity named entity recognition. *arXiv preprint arXiv:2204.05754*, 2022.

[5] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. {ATLAS}: A sequence-based learning approach for attack investigation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3005–3022, 2021.

[6] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. pages 5185–5194, 2019.

[7] Sean Barnum. Standardizing cyber threat intelligence information with the structured threat information expression (stix). *Mitre Corporation*, 11:1–22, 2012.

[8] Xander Bouwman, Harm Griffioen, Jelle Egbers, Christian Doerr, Bram Klievink, and Michel Van Eeten. A different cup of {TI}? the added value of commercial threat intelligence. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 433–450, 2020.

[9] Robert A Bridges, Kelly MT Huffer, Corinne L Jones, Michael D Iannacone, and John R Goodall. Cybersecurity automated information extraction techniques: Drawbacks of current methods, and enhanced extractors. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 437–442. IEEE, 2017.

[10] Robert A. Bridges, Corinne L. Jones, Michael D. Iannacone, and John R. Goodall. Automatic labeling for entity extraction in cyber security. *CoRR*, abs/1308.4941, 2013.

[11] Ryan Christian, Sharmishtha Dutta, Youngja Park, and Nidhi Rastogi. Poster: An ontology-driven knowledge graph for android malware. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2435–2437, 2021.

[12] Soham Dasgupta, Aritran Piplai, Anantaa Kotal, and Anupam Joshi. A comparative study of deep learning based named entity recognition algorithms for cybersecurity. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2596–2604. IEEE, 2020.

[13] Yongyan Guo, Zhengyu Liu, Cheng Huang, Jiayong Liu, Wangyuan Jing, Ziwang Wang, and Yanghao Wang. Cyberrel: Joint entity and relation extraction for cybersecurity concepts. In *International Conference on Information and Communications Security*, pages 447–463. Springer, 2021.

[14] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd annual computer security applications conference*, pages 103–115, 2017.

[15] Corinne L Jones, Robert A Bridges, Kelly MT Huffer, and John R Goodall. Towards a relation extraction framework for cyber-security concepts. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, pages 1–4, 2015.

[16] Elmar Kiesling, Andreas Ekelhart, Kabul Kurniawan, and Fajar Juang Ekaputra. The sepses knowledge graph: An integrated resource for cybersecurity. In *SEMWEB*, 2019.

[17] Gyeongmin Kim, Chanhee Lee, Jaechoon Jo, and Heuiseok Lim. Automatic extraction of named entities of cyber threats using a deep bi-lstm-crf network. *International journal of machine learning and cybernetics*, 11(10):2341–2355, 2020.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Max Landauer, Florian Skopik, Markus Wurzenberger, Wolfgang Hotwagner, and Andreas Rauber. A framework for cyber threat intelligence extraction from raw log data. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3200–3209. IEEE, 2019.

[20] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 755–766, 2016.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[22] Ning Luo, Xiangyu Du, Yitong He, Jun Jiang, Xuren Wang, Zhengwei Jiang, and Kai Zhang. A framework for document-level cybersecurity event extraction from open source data. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 422–427. IEEE, 2021.

[23] Arunesh Mathur, Gunes Acar, Michael J Friedman, Eli Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan. Dark patterns at scale: Findings from a crawl of 11k shopping websites. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–32, 2019.

[24] Rob McMillan. Definition: Threat intelligence, May 2013.

[25] Sadegh M Milajerdi, Birhanu Eshete, Rigel Gjomemo, and VN Venkatakrishnan. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1795–1812, 2019.

[26] Sandeep Nair Narayanan, Ashwinkumar Ganesan, Karuna Pande Joshi, Tim Oates, Anupam Joshi, and Timothy W. Finin. Early detection of cybersecurity threats using collaborative cognition. *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 354–363, 2018.

[27] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[29] 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. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 879–886, 2019.

[30] 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.

[31] 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.

[32] Nidhi Rastogi, Sharmishtha Dutta, Alex Gittens, Mohammad Zaki, and Charu Aggarwal. Tinker: A framework for open source cyberthreat intelligence. In *21st International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'22)*, number https://arxiv.org/abs/2102.05571, pages 1–14, 2022.

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

[34] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[35] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.

[36] Injy Sarhan and Marco René Spruit. Open-cykg: An open cyber threat intelligence knowledge graph. *Knowledge Based System*, 233:107524, 2021.

[37] Yun Shen and Gianluca Stringhini. {ATTACK2VEC}: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 905–921, 2019.

[38] Xiaokui Shu, Frederico Araujo, Douglas L Schales, Marc Ph Stoecklin, Jiyong Jang, Heqing Huang, and Josyula R Rao. Threat intelligence computing. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1883–1898, 2018.

[39] Pontus Stenetorp, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou, and Junichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2012.

[40] Nan Sun, Jun Zhang, Paul Rimba, Shang Gao, Leo Yu Zhang, and Yang Xiang. Data-driven cybersecurity incident prediction: A survey. *IEEE communications surveys & tutorials*, 21(2):1744–1772, 2018.

[41] Thin Tharaphe Thein, Yuki Ezawa, Shunta Nakagawa, Keisuke Furumoto, Yoshiaki Shiraishi, Masami Mohri, Yasuhiro Takano, and Masakatu Morii. Paragraph-based estimation of cyber kill chain phase from threat intelligence reports. *Journal of Information Processing*, 28:1025–1029, 2020.

[42] Asahi Ushio and Jose Camacho-Collados. T-ner: An all-round python library for transformer-based named entity recognition. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 53–62, 2021.

[43] Shikhar Vashishth, Prince Jain, and Partha Talukdar. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference*, pages 1317–1327, 2018.

[44] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

[45] Shanchan Wu and Yifan He. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2361–2364, 2019.

[46] Feng Yi, Bo Jiang, Lu Wang, and Jianjun Wu. Cybersecurity named entity recognition using multi-modal ensemble learning. *IEEE Access*, 8:63214–63224, 2020.