Looking Beyond IoCs: Automatically Extracting Attack Patterns from External CTI

Md Tanvirul Alam
Rochester Institute of Technology
Rochester, New York, USA
tanvirul.alam@mail.rit.edu

Dipkamal Bhusal
Rochester Institute of Technology
Rochester, New York, USA
db1702@rit.edu

Youngja Park
IBM Research
Yorktown Heights, New York, USA
young_park@us.ibm.com

Nidhi Rastogi
Rochester Institute of Technology
Rochester, New York, USA
nxrvse@rit.edu

ABSTRACT

Public and commercial organizations extensively share cyberthreat intelligence (CTI) to prepare systems to defend against existing and emerging cyberattacks. However, traditional CTI has primarily focused on tracking known threat indicators such as IP addresses and domain names, which may not provide long-term value in defending against evolving attacks. To address this challenge, we propose to use more robust threat intelligence signals called attack patterns. LADDER is a knowledge extraction framework that can extract text-based attack patterns from CTI reports at scale. The framework characterizes attack patterns by capturing the phases of an attack in Android and enterprise networks and systematically maps them to the MITRE ATT&CK pattern framework. LADDER can be used by security analysts to determine the presence of attack vectors related to existing and emerging threats, enabling them to prepare defenses proactively. We also present several use cases to demonstrate the application of LADDER in real-world scenarios. Finally, we provide a new, open-access benchmark malware dataset to train future cyberthreat intelligence models.

CCS CONCEPTS

• **Computing methodologies → Information extraction**; **Neural networks**; *Knowledge representation and reasoning*.

KEYWORDS

Threat Intelligence, Attack Patterns, Knowledge Graph, LADDER

ACM Reference Format:
Md Tanvirul Alam, Dipkamal Bhusal, Youngja Park, and Nidhi Rastogi. 2023. Looking Beyond IoCs: Automatically Extracting Attack Patterns from External CTI. In *The 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23), October 16–18, 2023, Hong Kong, China.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3607199.3607208

RAID '23, October 16–18, 2023, Hong Kong, China

ACM ISBN 979-8-4007-0765-0/23/10…$15.00
https://doi.org/10.1145/3607199.3607208

1 INTRODUCTION

Cyber Threat Intelligence (CTI) offers crucial insights into the rapidly evolving cyber threat landscape. This information includes any evidence to identify and assess the associated threats, such as indicators of compromise (IOCs), IP addresses, domain names, and file hashes, and any associated tactics, techniques, and procedures (TTPs) used by the attacker(s). For instance, CTI can provide comprehensive, contextual information on emerging threats like the advanced persistent threat (APT), ScarCruft [58]. Also known as APT37, the cyber threat intelligence on ScarCruft reported that the APT targets "individuals in South Korean organizations" with the primary objective of "cyber espionage." The CTI reports that the APT achieves this through "data exfiltration of selected file formats" and uses MD5 hashes, known as IoCs. Therefore, organizations need to leverage CTI to understand adversary tactics and goals, prevent future attacks, and shorten the time for remedial measures. Security analysts aggregate, clean, analyze, evaluate, and contextualize cyberattack information to produce comprehensive reports. The goal is to enhance cybersecurity-related decision-making for organizations facing similar threats [46]. Later, the CTI report can be disseminated to interested parties through paid subscriptions or free resources such as blogs, bulletins, news, and reports [8]. However, once the CTI is received, it can still present significant challenges for organizations that want to make this information actionable. Some of the key challenges are:

(1) **Timely identifying and extracting pertinent information and integrating threat signals into internal defense infrastructure.** While traditional forms of intelligence collected from static indicators such as malware hash and IP address can be obtained through pattern matching, they fail to detect new generations of cyber threats since attackers can modify them to evade detection [64]. For example, changing a single bit in the binary can alter the malware hash. As a result, there is a growing demand for more tactical threat intelligence extraction from CTI sources that are more robust to evolving adversary evasion TTPs.
(2) **Extracting pertinent information from CTI sources due to the unstructured, semi-structured format and presence of noisy information.** While some research focuses on automating the detection and extraction of Indicators of Compromise (IoC) [19], tracking IoCs is not the same as threat hunting as there is little overlap of shared

IoCs among organizations, and potential delay in IoCs usage and detection [8]. To address these issues, researchers have focused on extracting attack patterns from CTI [19, 29].

(3) **Extracting TTPs due to their natural language descriptions and the evolution of various sub-techniques over time.** Mapping extracted attack patterns to a standard format can decrease redundancy during further analysis. However, this is often difficult due to the ambiguity inherent in the text. Rule-based systems for attack pattern extraction may be inadequate when an attack pattern (or TTP) lacks associated trigger words. Consequently, a machine-learning-based approach is required to extract higher-level tactical intelligence from unstructured CTI texts.

We present LADDER, a framework to automatically extract Tactics, Techniques, and Procedures (TTPs) and other relevant information from CTI sources related to the malware and APTs. We restructure this information using an ontology [11] and TTPClassifier into a knowledge graph (KG) to enable predictive analysis (see Figure 1). TTPClassifier utilizes a novel machine-learning algorithm for TTP extraction from CTI reports (includes IoCs and TTPs). It categorizes the TTPs into standardized MITRE ATT&CK [42] pattern IDs, as shown in Section 4.3. The TTPClassifier enables analysts to learn and analyze attack campaigns for existing or emerging threats, ultimately helping to preempt potential attacks on their organizations. Our proposed framework addresses a critical gap in the automated extraction and analysis of CTI, providing a valuable tool for organizations to enhance their threat detection capabilities. The main contributions are:

1. We propose LADDER, a threat intelligence aggregation and analysis framework that automatically extracts and restructures attack patterns and IoCs as evidence of attacks found in diverse, unstructured CTI. LADDER also classifies them according to ATT&CK pattern techniques described by MITRE. This is the first work to include standardized ATT&CK patterns in the KG with other forms of threat intelligence.
2. We demonstrate the effectiveness of LADDER and its security applications for security analysts by accurately extracting attack patterns, performing predictive analysis of malware behavior, threat conducting threat hunting, and attributing APT groups. To the best of our knowledge, this is the first work to utilize KG for malware attack pattern prediction.
3. We provide a new, open-access benchmark malware dataset to train future cyberthreat intelligence models. It consists of 140,447 tokens, including manually annotated 11,555 named entities and 5,499 relations. The dataset and code are publicly available.[1]

2 MOTIVATING EXAMPLE

To inspire our research, We present an example of how LADDER can be utilized by security analysts to leverage CTI reports. We study the CTI of malware Cerberus, a trojan horse that targets Android mobile phone banking credentials. This CTI provides a comprehensive description of the malware's capabilities and a detailed account of its tactics, techniques, and procedures (TTPs), also called attack patterns. Furthermore, the CTI encompasses different vulnerabilities in various business technologies, including email, domains, and mobile devices. The following excerpt from a Cerberus CTI posted on ThreatPost [47] illustrates this:

"...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)...".

Figure 2 shows the transformation from a CTI report to a knowledge graph for "Cerberus". In Section 4, we detail the process of entity extraction (e.g., application), triple generation ⟨malware, targets, application⟩, and knowledge graph construction (combination of all triples). The table 2(b) shows entity classes and relationships following our ontology described in Section 3.2. 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 entity class definitions for Malware, Attack Pattern, Location, OS, Application (see Section 3.2) map to existing threat intelligence ontology classes [11, 51]. However, they have been adapted for CTI and security logs and follow the STIX2.1 framework for TTP and IoC exchange. Relationships between them are pre-defined within the same ontology. We extract triples from CTI reports to utilize historical malware information, including attack patterns, to train a threat intelligence model.

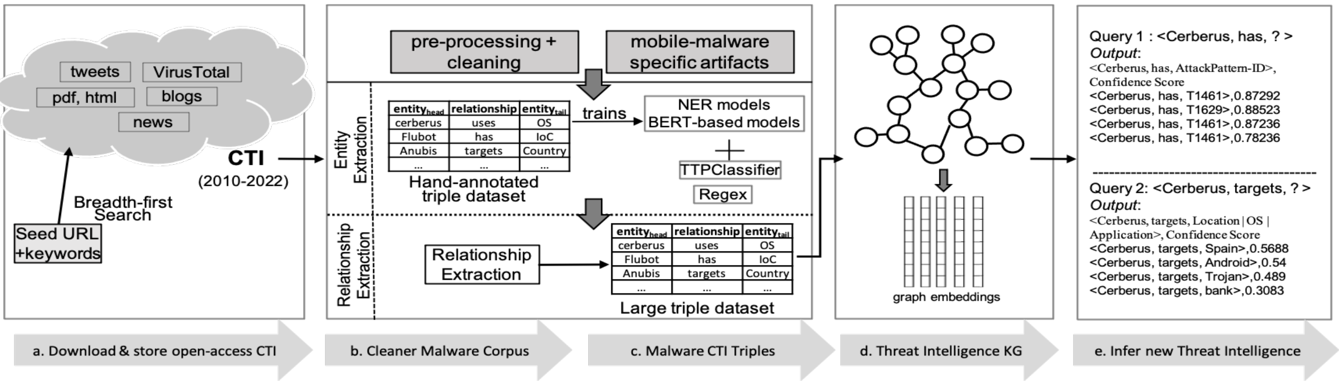
For instance, in Section 6.2: use cases, we show how an analyst maps attack patterns based on external CTI to internal security logs. Section 6 shows how an analyst queries our knowledge graph using the attack patterns extracted from the CTI. The graph can "infer" potential attack patterns that the same malware might attempt, even if these patterns have not previously been reported or observed. Knowledge of the MITRE ATT&CK is advantageous to the analyst as mitigation techniques are provided for each attack pattern, enabling even a less experienced security analyst to take timely actions. Using this knowledge, an analyst can take proactive measures to prevent or deter adversaries from causing damage to the internal network.

3 BACKGROUND

3.1 Cyber Threat Intelligence

Cyber Threat Intelligence (CTI) is evidence-based knowledge about existing or emerging cyber threats, which can facilitate decision-making processes in response to cyberthreats. CTI should be relevant (related to an objective), actionable (prompts a response to a threat), and valuable (contributes to a business outcome) [14]. CTI can be collected both internally within the organization and externally. Organizations can gather internal intelligence from the system and network endpoint logs. External threat intelligence is acquired (freely or at a cost) from sources outside the organization.

[1] https://github.com/aiforsec/LADDER

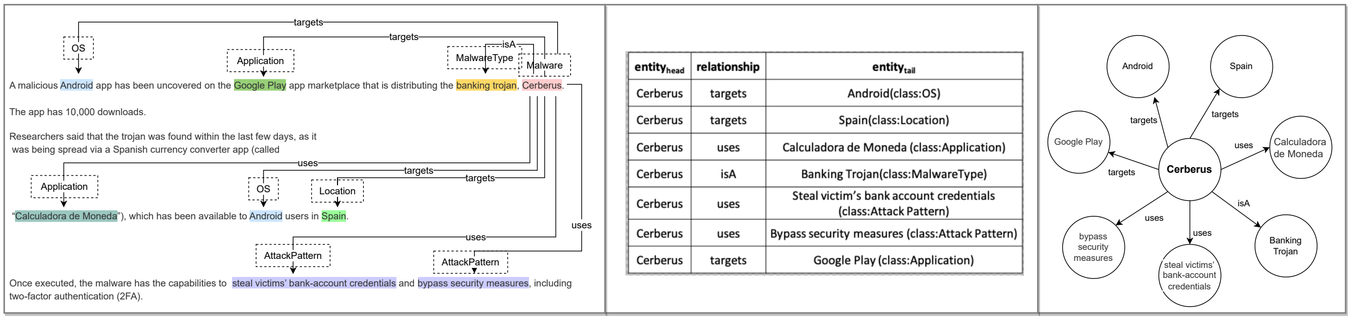

tweets
VirusTotal
pdf, html
blogs
news
CTI
(2010-2022)
Breadth-first Search
Seed URL +keywords
pre-processing + cleaning
mobile-malware specific artifacts
Entity Extraction
trains
NER models BERT-based models
TTPClassifier
Regex
Hand-annotated triple dataset
Relationship Extraction
Relationship Extraction
Large triple dataset
graph embeddings
a. Download & store open-access CTI
b. Cleaner Malware Corpus
c. Malware CTI Triples
d. Threat Intelligence KG
e. Infer new Threat Intelligence

$entity_{head}$	relationship	$entity_{tail}$
cerberus	uses	OS
Flubot	has	IoC
Anubis	targets	Country
...	...	...

$entity_{head}$	relationship	$entity_{tail}$
cerberus	uses	OS
Flubot	has	IoC
Anubis	targets	Country
...	...	...

```
Query 1 : <Cerberus, has, ? >
Output:
<Cerberus, has, AttackPattern-ID>,
Confidence Score
<Cerberus, has, T1461>,0.87292
<Cerberus, has, T1629>,0.88523
<Cerberus, has, T1461>,0.87236
<Cerberus, has, T1461>,0.78236
------------------------------------------
Query 2: <Cerberus, targets, ? >
Output:
<Cerberus, targets, Location|OS |
Application>, Confidence Score
<Cerberus, targets, Spain>,0.5688
<Cerberus, targets, Android>,0.54
<Cerberus, targets, Trojan>,0.489
<Cerberus, targets, bank>,0.3083
```

Figure 1: Proposed framework: LADDER and the five component modules. (a) Extracts CTI using crawlers, (b) Pre-processes and prepares for entity and relationship extraction, (c) generates data in the form of triples, (d) creates a knowledge graph by combining the triples and uses graph embedding methods to pack every triple's properties into a vector with smaller dimensions, and (e) returns instances when to queried by the analyst. (a)-(d) are training step, (e) is inference step.



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).

$entity_{head}$	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)

Figure 2: For malware Cerberus, (a) Annotated CTI using BRAT, (b) Triples created from the annotated snippet, (c) Knowledge graph from the triples (best viewed when zoomed).

CTI is collected from security bulletins, the dark web, hacking-related websites, public threat reports, source code repositories, and social media platforms like GitHub and Reddit [66].

Unstructured CTI. Open-source CTI comes in both **structured** and **unstructured** forms. Many standards have developed over the years for structured CTI sharing– STIX (Structured Threat Information eXpression) [6], TAXII (Trusted Automated eXchange of Indicator Information) [13], ATT&CK (AdversarialTactics, Techniques & Common Knowledge) [60] and many others. Structured CTIs allow efficient automation and collection of information from diverse sources of CTI. However, generating such structured threat reports is time-consuming and requires a lot of manual labor. As such, many public threat reports are provided in an unstructured format by different security farms such as Symantec [61], McAfee [35], Kaspersky [22]. Although these reports are more readily available, extracting relevant information for a specific organization or security analyst is still challenging. Without information extraction tools, security analysts may become overburdened with the large volume of information available [30]. This issue becomes even more significant when dealing with tactical threat intelligence, such as adversary tactics, techniques, and procedures. As such, there is a need for automatic information extraction from diverse unstructured CTI sources to make the information actionable.

3.2 Collecting and Structuring CTI Concepts

We create knowledge graphs from CTI because these graphs can transform unstructured information about CTI into a structured format. They can also store a vast amount of domain-specific information in the form of triples representing pairs of entities and the relationship between them [45]. This approach necessitates capturing context from threat intelligence information and representing it in a structured format using RDF expressions, <*subject, predicate, object*>. To facilitate this process, we adapt from existing ontologies on malware [11, 51]. The cyber threat concepts in the knowledge graph include Malware, Malware Type, Application, Operating System, Organization, Person, Time, Threat Actor, Location, and Attack Pattern. These concepts are connected through ten relations, including isA, targets, uses, hasAuthor, hasAlias, indicates, discoveredIn, exploits, variantOf, and has. Figure 3 shows a part of the ontology. Some triples may convey low-level threat intelligence, like the hash or IP addresses associated with the malware. Others can capture higher-level intelligence, such as the applications targeted

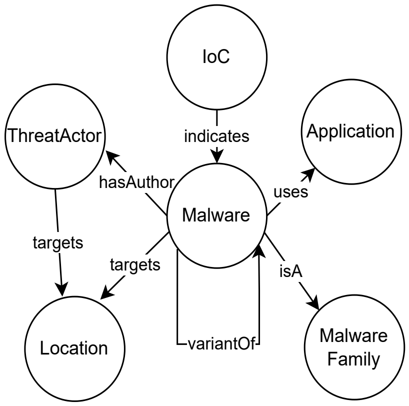


Figure 3: Part of the ontology used in the study

by malware or their attack patterns. We provide details of cybersecurity concepts in the Appendix. By structuring the open-access CTI in this manner, we enable efficient analysis and automated information extraction for security analysts.

3.3 Attack Patterns

Attack patterns depict an attacker's methods to achieve a tactical goal offering a high-level insight into the motivation behind an attack. For instance, an adversary may encrypt files on a device to prevent access until a ransom is paid. MITRE's Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework enumerates common attack patterns with 66 unique techniques for mobile platforms alone.[2] Unlike Indicators of Compromise (IoCs), which may be short-lived, Threat Tactic Techniques and Procedures (TTPs) deliver long-term intelligence for cyber threat analysis. TTPs can assist organizations in evaluating the effectiveness of their security measures against current threats and aid attackers' emulation for testing and validating defenses against prevalent techniques [4, 39].

4 SYSTEM DESIGN

4.1 Dataset Collection

To our knowledge, no public dataset containing triples extracted from CTI sources exists. To ensure the highest quality of triples for our knowledge graph creation, we curated CTI reports related to 36 malware, including *Cerberus, Rotexy, Judy, Gooligan,* and *SpyNote RAT* listed on the MITRE website; representative of Android malware during 2015-2022. Many attack patterns in these reports were paraphrased and mapped to the MITRE ATT&CK framework. These reports are authored by security analysts from reputable organizations such as McAfee, Symantec, and Kaspersky and provide natural language descriptions of the malware emergence, propagation, attack patterns, and IoCs.

Annotation. We employed the widely used open-source annotation tool, BRAT [59] to manually annotate threat concepts and their relationships, as explained in Section 3.2. An example of our annotation is shown in Figure 2. It is important to note that some entities may classify into different classes depending on their context. For instance, the terms *Facebook, Twitter* and *Instagram* can be classified as either Organization or Application depending on the context. We also annotated attack patterns as concepts, but these often include other concepts within them due to their extended length. In such cases, we annotated the larger text and the smaller entities within the text as an attack pattern. For example, the attack pattern *"break Android's application sandbox"* includes the annotation for *Android* as *OS*. We show the distribution of different entities in Figure 4 (l), with AttackPattern being the most common entity type, followed by Malware and Application. The distributions of relationships between entities are shown in Figure 4 (r). The relationship *uses* has the highest count because it links attack patterns with the associated Malware. The *Indicates* relation connects compromised indicators with the representative Malware. The *targets* relationship links Malware with ThreatActor and other entities. Finally, the *isA* relationship indicates a broader category or family of Malware.

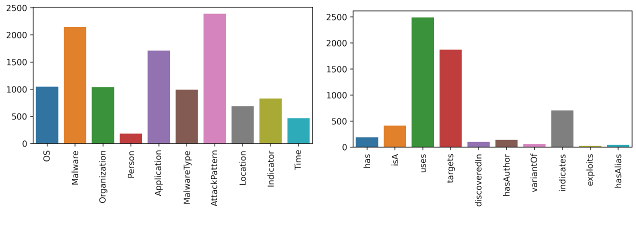


Figure 4: Entity (l) and relationships (r) distribution.

4.2 Information Extraction from CTI for Threat Intelligence Graph

Dataset Crawler. The annotated dataset enables us to train machine learning models for information extraction tasks. The trained models can then be used for knowledge graph extraction from a broader set of CTI reports. To this end, we developed a high-performance web crawler that scraped over 12,000 relevant unstructured open-access CTI reports from public URLs. We focused our search on security and technology companies and technology news reporting companies to ensure report quality. The crawler used a breadth-first search (BFS) starting from a seed URL belonging to a security or technical CTI website. It saved all URLs mentioned on the starting page and assessed their text for relevance. A relevant page included detailed descriptions of malware, such as the presence of malware-related keywords within the first n words of the article (where $n < 100$). The crawler saved the URL and its text if the page was relevant. Then, the text cleaner processed the content by removing HTML tags and images. We also extracted temporal information from the reports for longitudinal analysis using a heuristic-based approach. Specifically, we used the NER model provided in the Flair [2] NLP library to extract the first *DATE* entity found in the top five sentences, if present. We verified the date and extracted the year using the python datefinder [26] library. We limited our research to threat reports published between 2010-2021. Details of web crawling Algorithm 1 are in the Appendix.

[2] https://attack.mitre.org/techniques/mobile/

Entity Extraction. Once the threat reports are pre-processed and cleaned, we extract different entity classes using state-of-the-art natural language processing techniques. We fine-tune Transformer-based [68] pretrained language model using our hand-annotated dataset for classes including Malware, AttackPattern, Application, OS, Organization, Person, Time, Location. We adopt this approach because such large-scale language models are highly effective in numerous downstream NLP tasks with limited labeled data [16]. Specifically, we use three transformer variants in our experiments: BERT [16], RoBERTa[31] and XLM-RoBERTa[12]. These models use powerful attention mechanisms to capture context and extract evolving security concepts like attack patterns and malware. While BERT and RoBERTa are pre-trained on English corpora, XLM-RoBERTa is multilingual. These models leverage their powerful attention mechanisms to capture the context and extract evolving security concepts such as attack patterns and malware. We use the *tner* library [67] to fine-tune the models, which enables us to achieve high accuracy in entity recognition. Specifically, we take the hidden layer representation from the transformer model and add a classification layer with ten neurons corresponding to the nine entity classes and a special no entity (O) token, illustrating that a token does not belong to any entity classes. On the other hand, we use pattern matching to extract IoCs such as URL, IP address, email, and file name. We describe the regular expressions used for pattern matching and their corresponding entity types in the Appendix.

4.3 Attack Pattern Extraction

Attack pattern extraction and labeling pose unique challenges. Attack patterns comprise a larger block of text that describes a cyber threat action rather than a single named entity. Sometimes, an attack pattern may include other entity types within its description. For example, in the sentence "Cerberus is capable of generating an instance of TeamViewer on mobile," the attack pattern phrase "capable of generating an instance of TeamViewer on mobile" contains the entity "TeamViewer" of type "Application" within it. These nuances necessitate a different approach to extraction and labeling, considering the broader context of the attack pattern and its relationship to other entities in the text. To address these challenges, we present TTPClassifier, a novel approach for extracting attack patterns from threat reports. This approach comprises three sub-tasks: relevant sentence extraction, attack phrase identification & extraction, and mapping attack patterns to the MITRE ATT&CK. We discuss these sub-tasks below:

(1) **Relevant Sentence Extraction.** Firstly, we identify sentences that contain an attack description. We approach this task as a binary sentence classification problem, where sentences containing one or more attack patterns are labeled positive and the rest negative. During training, we use all annotated positive instances and randomly select an equal number of negative instances from the remaining sentences to form a balanced dataset. We fine-tune pre-trained transformer models for this classification task, adding a linear layer with two neurons. For RoBERTa models, we add a hidden layer before the linear layer.

(2) **Attack Pattern Identification & Extraction.** Secondly, we identify the relevant parts of sentences containing attack pattern descriptions for those predicted as positive, i.e., having at least one attack pattern. We use a sequence tagging model for this subtask, similar to entity extraction. Using two classes, the model predicts whether each word in the sentence is part of an attack pattern description. Some sentences may contain *more than one* attack pattern, necessitating the combination of each contiguous block tagged with the attack pattern entity into a single attack pattern description. Consider the following example: *"The malware can covertly send and steal SMS codes, open tailored overlays for various online banks, and steal 2FA-codes"*. This sentence contains three attack patterns: (a) *"covertly send and steal SMS codes"*, (b) *"open tailored overlays for various online banks"*, and (c) *"steal 2FA-codes"*. Since TTPClassifier makes predictions for individual tokens, we combine each contiguous block tagged with the attack pattern entity into a single attack pattern description. During post-processing, we discard invalid extractions, e.g., those that do not contain verbs.

(3) **Mapping to ATT&CK ID.** Finally, we map each extracted attack pattern to standardized ATT&CK techniques. Although ATT&CK has both techniques and sub-techniques, we only consider the former and map each extracted sequence to one technique. Due to the large number of potential classes and significant annotation effort required to match an attack pattern to its corresponding ATT&CK ID, we adopt a semantic similarity-based approach for the mapping task. We first compute embeddings for the extracted attack pattern phrases using a pre-trained sentence transformer model [52]. We use embeddings from the title and description of the ATT&CK ID described on the website for improved learning. Sometimes a CTI report mentions an attack pattern that closely resembles the title. However, we need the description at other times to identify the matching technique. For example, for MITRE ATT&CK ID: Location Tracking, we have descriptions of the form *"eSurv can track the device's location"* for one malware and *"Pallas tracks the latitude and longitude coordinates of the infected device"* for another malware.[3] By computing the similarity between the attack pattern title and description, we can correctly match against both. Specifically, we compute a metric, the weighted distance between the extracted phrase and an ATT&CK ID i, as follows:

$$d_i = w_t cos(v_{phrase}, v^i_{title}) + (1 - w_t) cos(v_{phrase}, v^i_{desc})$$

Where v_{phrase} is the vector embedding for the extracted attack phrase, v^i_{title} and v^i_{desc} are the vector embeddings for the i^{th} ATT&CK ID, respectively. cos represents the cosine distance between two vectors u, v computed as

$$cos(u, v) = 1 - \frac{u.v}{||u||_2||v||_2}$$

We iterate over all the different attack patterns present for a platform (66 techniques for mobile platforms and 196 for

[3] https://attack.mitre.org/techniques/T1430/

enterprise platforms) and find the ID with the smallest distance. We output this as the mapped ID if the distance is less than a threshold τ. We identify the optimum value for tau experimentally.

Unstructured threat reports can lead to variations in the description of the same attack pattern across different reports. This redundancy of information can impede the effectiveness of pattern prediction. To mitigate this issue, we map the extracted attack patterns to standardized ATT&CK techniques, allowing us to have a fixed number of attack patterns in the knowledge graph.

Although our model is trained on mobile platform CTI reports, it can be utilized to extract attack patterns for other platforms. Our platform-agnostic algorithm can extract attack patterns as they appear in the text. The mapping steps can be changed to incorporate the appropriate list of attack patterns in MITRE for the target platform. In Section 6, we demonstrate the effectiveness of our approach in extracting attack patterns from enterprise CTI reports.

4.4 Adding Relationship to Concepts

We train a relation classification model to determine the relationship between each pair of entities mentioned in the report. We only consider a pair of entities for relation extraction if a valid relationship may exist between them according to the adopted ontology [11, 51]. For example, we may have a relationship between a pair of entities of type Malware and Application, e.g., $\langle Malware, targets, Application\rangle$. However, we do not have a valid relationship type when two entities are of type Application and Time. Similar to NER, we use transformer-based models for the relation extraction task. Our approach incorporates entity information for relation classification [71]. Given a text s with a pair of entities e_1 and e_2, we introduce four tokens that capture the position information of the entities. Consider the example:

"Cerberus is capable of generating an instance of TeamViewer on mobile."

where, e_1 is *Cerberus* and e_2 is *TeamViewer*. The formatted sentence will be: *[CLS] ⟨e1⟩ Cerberus ⟨/e1⟩ is capable of generating an instance of ⟨e2⟩ TeamViewer ⟨/e2⟩ on mobile.*

[CLS] is the unique start of sequence token for the BERT model. We concatenate the hidden layer representation for the start position of both entities and generate the final vector embedding. We pass the vector through a couple of fully connected layers to predict the relation type between the entities. Once the triples are generated from the large corpus, the threat intelligence knowledge graph is ready for querying.

4.5 Querying LADDER

Knowledge Graph. The threat intelligence graph is a directed knowledge graph, KG = $\{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}, \mathcal{R}$ and $\mathcal{T}$ indicate the sets of entities, relations, and triples, respectively [20]. Each triple $\langle e_{head}, r, e_{tail}\rangle \in \mathcal{T}$ indicates that there is a relationship $r \in \mathcal{R}$ between $e_{head} \in \mathcal{E}$ and $e_{tail} \in \mathcal{E}$. KG link prediction is the task of predicting the best candidate for a missing entity. Formally, the task of entity-prediction is to predict the value for e_{head} given $\langle ?, r, e_{tail}\rangle$ or e_{tail} given $\langle e_{head}, r, ?\rangle$, where "**?**" indicates a missing entity (head or tail).

Table 1: 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 2: 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

Vector Embeddings: All triples are mathematically represented by three vectors, $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^{d_e}$, $r \in \mathbb{R}^{d_r}$, where d_e and d_r are the embedding dimensions of entities and relations, respectively and have relatively low (e.g., 30-200) dimensional vector space embeddings [5, 69]. Embeddings preserve information about the structure and key features of the triple. The embeddings for all the triples in the KG involve factorization of co-occurrence-based tensors [5], which leads to reducing the dimensionality of the triples when creating embeddings for entities and relations in the KG. We use the f_ϕ notation for the scoring function of each triple $\langle e_{head}, r, e_{tail}\rangle \in \mathcal{T}$. The scoring function measures the plausibility of a fact in $\mathcal{T}$, based on translational distance or semantic similarity [28].

Querying KG. With the threat intelligence graph, KG built, security analysts can query the graph where the query is posed in a triple format. Querying a knowledge graph is related to knowledge graph link prediction. Using the link prediction approach, LADDER predicts e_{tail} by learning a scoring function. Entity prediction for KG follows TuckER [5] since it outperforms traditional link prediction models [50]. TuckER is a linear model based on tucker decomposition [65] of entity embedding matrix and relational embedding matrix in a knowledge graph. We also evaluate the accuracy and recommend ranking inferred entities [5].

5 EXPERIMENTS AND RESULTS

5.1 Information Extraction

We used the PyTorch deep learning framework to implement our models for information extraction and subsequent tasks. To achieve high performance, we fine-tuned the pre-trained transformer models from Huggingface's transformers library. For the Named Entity Recognition (NER) models, we set the sequence length to 128 and trained the models for 20 epochs, with 32 samples per mini-batch.

Table 3: 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 4: 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

We used the AdamW optimizer with a learning rate of 1e-5 for the base (BERT-base, RoBERTa-base, XLM-RoBERTa-base) and 1e-6 for large models (BERT-large, RoBERTa-large, XLM-RoBERTa-large). To avoid the repetition of reports written on the same malware, we split the annotated datasets based on the malware under discussion. Out of 36 malware, the training dataset comprised 26, and the validation and test datasets were five malware each. In total, there were 104 large CTI reports in the training split, 21 in the validation split, and 25 in the test split.

5.1.1 Entity Extraction. The results for the named entity extraction task are shown in Table 1. XLM-RoBERTa large model achieves the best performance for this task with an average F1 score of 78.98%. Class-specific results for this model are shown in Table 2. Classes with more samples in the training data generally produce better results. Operating System and Location classes yield better results than the other classes as they exhibit less variation in form. Performance reduces for Application and Organization as they have some overlap between them (e.g., Facebook and Twitter can be both Application and Organization). The most challenging class is the ThreatActor because it is usually an Organization or Person with malicious intent. So, this requires a thorough understanding of the context to detect them correctly. 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.

5.1.2 Attack Pattern Extraction. We use the same malware split in the NER task for attack pattern extraction. In a CTI report, usually, there are more sentences without an attack pattern than those that do. So, we randomly sample an equal number of negative sentences to balance the 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. To train the models, we use a mini-batch size of 32, and the optimal learning rate is chosen from [1e-5, 5e-5, 1e-6] using the validation set. We train the sentence classification and attack phrase extraction models for 20 and 30 epochs, respectively. We use Adam [25]as the optimizer for learning the model parameters.

Table 5: 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 6: 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

Table 3 shows the binary sentence classification task results. We achieve greater than 85% F1-score for all the models, with RoBERTa large performing best with an average F1-score of 89.62%. Table 4 shows the result for the attack phrase extraction task. Again, the RoBERTa-large model achieves the best result with an average F1-score of 90.64%. These results indicate that our algorithm can effectively identify the relevant sentences for attack patterns and accurately extract relevant parts. To learn the optimal parameter values w_t and τ required for mapping to MITRE ID, we manually map 80 randomly selected attack pattern descriptions annotated in our training corpus with their corresponding ATT&CK ID. The optimum values for the two parameters are 0.4 and 0.6, respectively.

5.2 Relation Extraction

When extracting relationships from threat reports, evaluating each pair of entities and inferring the relationship between them is essential. However, many pairs of entities may not have a meaningful relation in the context, even if they are valid according to the ontology. To account for such cases, we add the *NoRelation* class that predicts the absence of a relation between the entities under consideration. We randomly sample such plausible entities from the annotated CTI reports and use an 80:20 split for training and inference. Since the CTI reports may have a relation between entities that are far apart in the reports, we increase the sequence length to 512. Due to GPU memory constraints, we use a mini-batch size of 8 for the large and 16 for the small transformer models, respectively. We determine the optimal learning rate from [1e-5, 5e-5, 1e-6] and train the models for ten epochs using the AdamW optimizer.

Table 5 shows the model performance for the relation extraction task. The BERT-based model outperforms the other two models for this task. The BERT-large model shows the highest performance with a 92.62% average F1-score. Table 6 shows the results for specific classes. Attack patterns are part of a single relation type (*Malware uses AttackPattern*), therefore not included in the relation extraction. Consequently, we can infer the relation for the attack pattern once the malware under discussion is identified. While *discoveredIn* performs well for the annotated corpus because the reports were cleaned and mostly did not contain redundant time information on malware discovery. Among other classes, *uses* and *hasAlias* have the worst performance. Context is challenging with *uses* as it may get confused with *targets* as both contain the same type of head-tail entity pairs (e.g., Malware and Application. Relationship between two malware (*variantOf* and *hasAlias*) is challenging to detect since they may be expressed at a distant position in the report. It is important to note that these results are obtained from the manually cleaned test dataset, and the performance declines with more noisy texts.

5.3 Threat Intelligence Knowledge Graph

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

5.4 Inferring Entities

We create two test sets from the hand-annotated documents with varying triples for the prediction task. Entities in these triples belong to classes Malware, e.g., AttackPattern, Location, Application, Organization for testing. TestSet-1 consists of 25% of the annotated triples, and $TestSet_2$ consists of 40% of the triples. We experiment with three knowledge graphs for the prediction task; KG_1 has the remaining triples in hand-annotated CTI reports, and KG_2 consists of the triples generated using LADDER from the 12,000 documents. We train TuckER to predict the tail entities, employing 50 embedding dimensions with a mini-batch size of 64. The model is trained for 1000 iterations with an initial learning rate of 0.001.

Evaluation Criteria. To rank the performance of the prediction task, particularly in the context of knowledge graph prediction, we use evaluation criteria, including Mean Rank, Mean Reciprocal Rank (MRR), and Hits@*n*. These are calculated from the ranks of all true (actual) test triples that TuckER returns. MRR is the average 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. Higher scores are considered better.

Table 7: 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
KG_1	0.209	0.365	0.497	0.186	0.090	0.195	0.322	0.093
KG_2	0.221	0.353	0.516	0.211	0.215	0.359	0.501	0.203

Table 8: 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	KG_1	0.354	0.634	0.847	0.314	0.212	0.441	0.657	0.210
	KG_2	0.444	0.700	0.940	0.420	0.453	0.694	0.936	0.415
Location	KG_1	0.042	0.096	0.205	0.048	0.018	0.033	0.096	0.024
	KG_2	0.036	0.096	0.247	0.044	0.018	0.092	0.225	0.034
Application	KG_1	0.100	0.165	0.230	0.086	0.032	0.094	0.178	0.040
	KG_2	0.026	0.083	0.126	0.030	0.040	0.102	0.129	0.034

Results. Table 6.3 presents the results for the inference task. $TestSet_1$ performs similarly using the different training datasets for all Hits@(n) values. However, for $TestSet_2$, KG_1 performs worse than KG_2, with a difference of 16.4% Hits@10 between KG_1 and KG_2. This suggests that the additional triples obtained from a larger knowledge graph enable the model to make better predictions. KG_2 performs similarly on both the test datasets suggesting better generalizability. We show class-specific prediction results in Table 8 using KG1 and KG2 for three different tail entities - AttackPattern, Location and Application where the head entity is Malware. The most promising result is obtained for predicting AttackPattern, primarily from having 66 unique attack patterns. At the same time, the number of possible tail entities is much larger for relations involving other classes. This result implies that malware with similar properties may exhibit similar attack patterns. Similar to the aggregate result above, we observe no significant drop in performance for the two test datasets for KG_2. KG_1 sees a significant drop.

5.5 Comparison with state-of-the-art for TTP Classifier

We compare our attack pattern extraction with TTPDrill [19] and AttackKG [29] as they are the closest to LADDER, although there are differences that we discuss in the related work section. We use the open-source implementation of TTPDrill and AttackKG in our evaluation. Since TTPDrill and AttackKG provide models and patterns for the enterprise platform, we use the same for evaluation. We update the third step of our proposed TTPClassifier and match extracted phrase against the attack patterns listed on MITRE ATT&CK for enterprise [4]. Even though we trained our sentence classification and phrase extraction models for attack patterns on CTI gathered on mobile platforms, our evaluations also show high accuracy when testing CTI for other platforms since the semantic style for describing attack patterns is the same. This is because the description of the techniques follows a similar pattern in written texts.

[4] https://attack.mitre.org/techniques/enterprise/

Table 9: Comparison of attack pattern extraction with other methods (TP: true positives, FN: False Negatives, FP: False positives)

Method	TP	FN	FP	Precision	Recall	F1-score
MITRE	38	27	0	**1.00**	0.58	**0.74**
TTPDrill[19]	22	43	231	0.09	0.34	0.14
AttackKG[29]	12	53	85	0.12	0.18	0.15
TTPClassifier	41	24	22	0.65	**0.63**	0.64

When analyzing the attack patterns listed in the MITRE ATT&CK website for malware, *we often found that not all attack patterns reported in a CTI report are present. This indicates the difficulty of this task, even for human annotators.* In order to have a fair comparison, we create ground truth annotation from threat reports listed on the MITRE ATT&CK website for five different malware containing 9360 tokens. We show the result in Table 9. The attack patterns listed on the MITRE website have the overall best F1 score. Even though we did not find any false positives, 27 out of the 65 attack patterns we identified were not listed. This suggests that it is very likely that security analysts may miss some attack patterns when reading long, detailed CTI reports. Our proposed TTPClassifier achieves better recall than those listed on the MITRE website. TTPDrill and AttackKG achieve similar F1 scores, with TTPDrill showing better recall but having a lot of false positives. Since both approaches use template matching based approach generated from attack pattern description, they are ill-equipped to filter out irrelevant parts of large documents, which results in many false positives. Another issue with the template-matching-based approach is that it is challenging to identify a novel attack pattern when there is not enough example pattern available for an attack pattern. However, our machine-learning-based approach can mitigate this issue and identify new or emerging attack patterns.

5.6 Error Analysis

Since TTPClassifier is a multi-step algorithm, errors in an earlier stage may influence the result of the next stage. For example, if a sentence is misclassified as irrelevant, attack patterns present in that step will not be captured in the subsequent steps. However, a CTI report often describes the same attack pattern in multiple places. As a result, the attack pattern descriptions missed from one part of the report may be captured by another. This effect is further lessened when attack patterns are aggregated from multiple threat reports. Another case of an error that may be introduced in the first step is erroneously classifying an irrelevant sentence as positive. However, since the second step of the algorithm is independent of the first, some of them may be reduced if the model cannot identify any descriptive phrase of attack pattern in the sentence. False positives introduced in the second step may not match any existing attack pattern description and may be omitted in the final output. There may be errors introduced in the last step of the algorithm; however, since the framework can provide accompanying descriptions, analysts can verify them. We show statistics of the errors introduced after each step of the algorithm in Table 10 for the analyzed reports. The number of false positives is usually additive unless false positives introduced in one step are removed in a later stage. Also seen is that most of the false negatives are introduced in the first step of the algorithm, while most false positives are introduced in the last stage. Since the reports used in the experiment are all for different malware, the effect of false negatives will be reduced when results are aggregated from multiple reports for the same malware.

Table 10: The number of errors introduced after each algorithm step.

Step	FN	FP
1. Relevant Sentence Extraction	23	8
2. Attack Phrase Extraction	3	11
3. Mapping to MITRE ID	7	22

6 CASE STUDIES

6.1 Attack Pattern Extraction and Trend Analysis

The results in the preceding section indicate that the proposed algorithm can effectively extract attack patterns from CTI reports on different platforms. In this case study, we examine attack patterns extracted for a Windows malware, LitePower [1] in Table 11. We extract 18 attack patterns from this CTI report [55]. We list the five attack patterns shared between TTPClassifier and attack patterns listed for that malware on the MITRE website. TTPDrill failed to identify two example attack patterns: *T1518 Software Discovery*, and *T1082 System Information Discovery*. Interestingly, our algorithm extracted the relevant phrases for those patterns. The extracted phrase for T1518 was *conducts system reconnaissance to assess the AV software installed and the user privilege*, which got mapped to T1497– Virtualization/Sandbox Evasion. Upon further inspection, we noticed the phrase matched the sub-technique *System Checks*, which starts with the description *Adversaries may employ various system checks to detect.* This description was similar to the first half of the phrase *conducts system reconnaissance*, which resulted in the match. The extracted phrase for the second attack pattern was *volumeserialnumber List local disk drives.* The word *volumeserialnumber* was part of a Table in the CTI. This phrase did not match with any attack pattern with high enough similarity. The closest match was T1619– Cloud Storage Object Discovery. However, the description of this attack pattern contains a reference to another attack pattern *File and Directory Discovery*, which explains its relatively higher similarity.

A significant advantage of our proposed approach is that we can extract relevant phrases from CTI even when they are not mapped correctly or do not have a unique mapping with MITRE attack patterns. An example of the latter is the absence of an attack pattern in MITRE mobile platforms for *Masquerading*, which is included for enterprise platforms [41]. However, we have noticed several Android malware exhibiting this attack pattern. One such malware is Ginp. As described in a threat report published by ThreatFabric [63], this malware was *masquerading as a "Google Play Verificator" app.* Our approach can give an analyst the summarized version of the CTI report, including mentions of attack steps used by malware. We show **three example attack patterns** extracted by our algorithm

Table 11: Example Attack Patterns extracted from a threat report using TTPClassifier for LitePower malware

MITRE ID	Name	Description in Report	ATT&CK	TTPClassifier
T1059	Command and Scripting Interpreter	use a PowerShell script to execute commands	✓	✓
T1041	Exfiltration Over C2 Channel	send collected data, including screenshots, over its C2 channel	✓	✓
T1012	Query Registry	checks for the registry keys added for COM hijacking	✓	✓
T1113	Screen Capture	takes system screenshots and saves them to % AppData %	✓	✓
T1053	Scheduled Task/Job	creation of a legitimate scheduled task	✓	✓
T1518	Software Discovery	can identify installed AV software	✓	x
T1082	System Information Discovery	list local drives and enumerate the OS architecture	✓	x
T1564	Hide Artifacts	hide the main dropper spreadsheet	x	✓
T1112	Modify Registry	current user registry hive (HKCU)	x	✓
T1588	Obtain Capabilities	download and deploy further malware	x	✓

but not listed on MITRE ATT&CK. For example, the last attack pattern listed, T1588, describes that the malware can download and deploy further malware. These results suggest that the proposed algorithm can alleviate human labor when analyzing cyber threat reports.

Large Scale Malware Behavior Analysis: The proposed TTPClassifier is used to extract attack patterns for 433 malware instances found in 12K threat reports to perform attack pattern trend analysis. We only count unique attack patterns for the same malware if they are mentioned in multiple reports and obtain 3159 attack patterns between 2015 and 2021. We map the attack patterns to MITRE attack technique IDs and plot the distribution of attack IDs against time, see Figure 5(r) for three different trends. Refer to Figure 5(l) for a plot on all other attack techniques. The trend analysis shown in Figure 5(l) and Figure 5(r) are based on the CTI sources we have analyzed and are not representative of attack patterns deployed by malware in the wild.

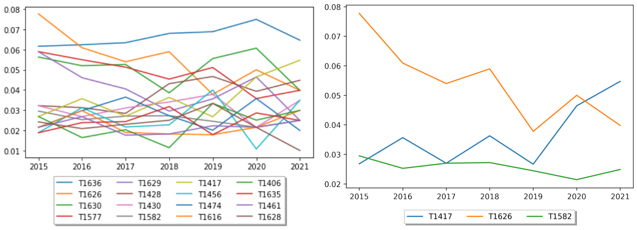
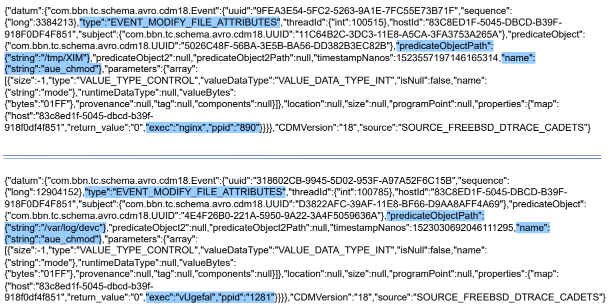


Figure 5: Distributions of all (l) and three (r) attack techniques vs. time. The X-axis represents the year when an attack technique was observed in the CTI, and Y-axis represents the normalized count of that attack pattern (ratio of the count of an attack technique observed in that year to the total number of attack techniques observed in that year).

In Figure 5(r), we observe an upward trend of T1417 (Input Capture) with peak usage in 2021. This attack technique encompasses any methods an adversary uses to steal the application credentials of users. Adversaries can use keylogging or GUI capture methods to steal user input. For instance, Anubis malware has a keylogger that works in every application installed on the device [17]. Over the years, a potential reason for the increase in this attack technique is the increased use of mobile-based digital solutions like banking, finance, and shopping. We expect this attack technique to become more prevalent in the coming years. We observe a downward trend of T1626 (abuse elevation control mechanism). T1626 encompasses methods an adversary uses to grant themselves high-level permissions in a device. For example, Red Alert 2.0 malware can request device administrator permissions [10]. One reason for the reduction in this attack technique could be that users have become more cautious of applications that ask for device administration requests. Also, Android OS 7+ has introduced changes that make abuse of administrator privilege more difficult. As more devices update to the latest OS, adversaries will find it more difficult to manipulate users to gain elevated access to devices and use them for malicious purposes. T1582 (SMS Control) exhibits an almost flat trend over the years, with a slight dip and rise in multiple years. This attack ID represents the SMS control technique where an adversary deletes, alters, or sends SMS messages without user permission. For example, Anubis malware can send, receive, and delete SMS messages from a user's device [17]. This trend suggests that SMS phishing remains as widespread on the mobile platform as before.

6.2 Threat Hunting

Automating the process of attack pattern extraction can assist in threat hunting and protection against APT campaigns. Correlating attack patterns extracted from CTI with kernel logs can pinpoint an attacker's activities. However, the same APT campaign may manifest differently in different settings due to differences in OS, targeted applications, and threat variations. As a result, relying on IoCs for precise threat hunting is unreliable since attackers can modify them to evade detection [27]. We illustrate this with sample logs from the DARPA Transparent Computing Engagement 3 dataset in Figure 6. The logs collected in the figure display an attack on a FreeBSD server that exploits an Nginx backdoor vulnerability. These logs exemplify the attack pattern T1222– File and Directory Permissions Modification which was used as a precursor to creating a new elevated process (attack pattern T1548). The accompanying ground truth CTI description refers to the *ability to create a new elevated process*, with the interactions as $F1 > elevate/tmp/XIM$. As we can see, there are variations in the logs due to the process ID, process name, and the file that is being modified. Consequently, instead of relying solely on matching the exact IoC, which may differ from what is described in a CTI report for a particular setting, it is more reliable to use high-level abstract information like attack patterns in conjunction with IoCs to identify attacks.



```
{"datum":{"com.bbn.tc.schema.avro.cdm18.Event":{"uuid":"9FEA3E54-5FC2-5263-9A1E-7FC55E73B71F","sequence":
{"long":3384213},"type":"EVENT_MODIFY_FILE_ATTRIBUTES","threadId":{"int":100515},"hostId":"83C8ED1F-5045-DBCD-B39F-
918F0DF4F851","subject":{"com.bbn.tc.schema.avro.cdm18.UUID":"11C64B2C-3DC3-11E8-A5CA-3FA3753A265A"},"predicateObject":
{"com.bbn.tc.schema.avro.cdm18.UUID":"5026C48F-56BA-3E5B-BA56-DD382B3EC82B"},"predicateObjectPath":
{"string":"/tmp/XIM"},"predicateObject2":null,"predicateObject2Path":null,"timestampNanos":1523557197146165314,"name":
{"string":"aue_chmod"},"parameters":{"array":
[{"size":-1,"type":"VALUE_TYPE_CONTROL","valueDataType":"VALUE_DATA_TYPE_INT","isNull":false,"name":
{"string":"mode"},"runtimeDataType":null,"valueBytes":
{"bytes":"01FF"},"provenance":null,"tag":null,"components":null}]},"location":null,"size":null,"programPoint":null,"properties":{"map":
{"host":"83c8ed1f-5045-dbcd-b39f-
918f0df4f851","return_value":"0","exec":"nginx","ppid":"890"}}}},"CDMVersion":"18","source":"SOURCE_FREEBSD_DTRACE_CADETS"}
```

```
{"datum":{"com.bbn.tc.schema.avro.cdm18.Event":{"uuid":"318602CB-9945-5D02-953F-A97A52F6C15B","sequence":
{"long":12904152},"type":"EVENT_MODIFY_FILE_ATTRIBUTES","threadId":{"int":100785},"hostId":"83C8ED1F-5045-DBCD-B39F-
918F0DF4F851","subject":{"com.bbn.tc.schema.avro.cdm18.UUID":"D3822AFC-39AF-11E8-BF66-D9AA8AFF4A69"},"predicateObject":
{"com.bbn.tc.schema.avro.cdm18.UUID":"4E4F26B0-221A-5950-9A22-3A4F5059636A"},"predicateObjectPath":
{"string":"/var/log/devc"},"predicateObject2":null,"predicateObject2Path":null,"timestampNanos":1523030692046111295,"name":
{"string":"aue_chmod"},"parameters":{"array":
[{"size":-1,"type":"VALUE_TYPE_CONTROL","valueDataType":"VALUE_DATA_TYPE_INT","isNull":false,"name":
{"string":"mode"},"runtimeDataType":null,"valueBytes":
{"bytes":"01FF"},"provenance":null,"tag":null,"components":null}]},"location":null,"size":null,"programPoint":null,"properties":{"map":
{"host":"83c8ed1f-5045-dbcd-b39f-
918f0df4f851","return_value":"0","exec":"vUgefal","ppid":"1281"}}}},"CDMVersion":"18","source":"SOURCE_FREEBSD_DTRACE_CADETS"}
```

Figure 6: 2 actual logs for attack pattern T1222– File and Directory Permissions Modification. Blue highlights relevant part(zoom for best view).

Challenges. Extracting attack patterns from kernel logs is beyond the scope of this study. There are a few ongoing open-source efforts for developing rules for MITRE attack pattern extraction from logs– for example, the Sigma rules.[5] However, these rules do not yet adequately cover different attack patterns. Some rules are described in terms of specific tools or software, and when different software with the same functionality is used, they cannot be captured with the existing rules. Rules also differ based on the operating systems. For example, Out of 193 enterprise attack patterns in MITRE, there are sigma rules for 60 for Linux, resulting in a coverage of 31%. As a result, many attack patterns identified through CTI may remain undetected in the log. Regardless, improvement in attack pattern extraction from logs can provide analysts with an efficient way of identifying specific APT attacks within kernel logs. This is one of our key objectives for future research.

6.3 Attack Pattern Prediction

Information extracted from a single CTI often lacks comprehensive information about a specific malware. Consolidating data from multiple reports allows us to bridge that gap and gain a more holistic perspective of any cyber attack. However, it is still possible that some specific malware characteristics may not be captured in the analyzed reports, resulting in missing information or gaps in the knowledge graph. We can find such missing information using knowledge graph link prediction, where the goal is to predict a missing tail entity given the head entity and the relation type. Another way to look into this is that as malware evolves, it can attempt new intrusion approaches resulting in attack patterns that have yet to be captured. We can use the link prediction task as a proxy to predict such attack patterns, which the malware may use in the near future.

We conduct a study of *AttackPattern* prediction for two malware: Anubis and Flubot. Anubis is an Android banking trojan that was active from 2017 until late 2021 and employed a wide range of attack techniques [33]. Flubot is another banking malware targeting Android users since the first quarter of 2021 and has been active ever since [34]. We identified 33 unique attack patterns for Anubis and 23 for Flubot from the reports collected during these periods. We used triples from the KG_2 (see Table 6.3) graph for the prediction task. We removed *AttackPattern* triples for making a prediction (i.e. when predicting for Anubis, we remove triples of the form *Anubis uses T1636*). Next, we used TuckER to predict the tail entities. For instance, we queried for the *uses* relation, e.g., $\langle Anubis, uses, ?\rangle$. Table 12 shows the top 20 attack patterns predicted for both malware and confidence scores. As we include more predictions, it invariably leads to less confident outputs and results in more false positives. However, we see promising results for the top predictions. We get nine correct predictions for Anubis and seven correct predictions for Flubot out of the top 10. When considering the top 15, the number of correct predictions is 12 for both malware types. These findings suggest a strong correlation between different malware behavior. We can leverage a knowledge graph of this correlation for predicting unknown behavior, viz., inferring future attack patterns from available information. We can also use this to fill up missing information in existing knowledge graphs.

Table 12: AttackPattern prediction for Anubis & Flubot sorted by confidence (Green: Observed,Red: Not Observed Patterns.)

Anubis		Flubot	
Prediction	**Confidence**	**Prediction**	**Confidence**
🟩 T1636	0.522	🟩 T1636	0.743
🟩 T1626	0.410	🟥 T1630	0.661
🟩 T1630	0.406	🟩 T1577	0.652
🟩 T1577	0.385	🟩 T1626	0.646
🟩 T1629	0.354	🟥 T1428	0.592
🟩 T1428	0.351	🟩 T1629	0.580
🟩 T1430	0.274	🟥 T1430	0.508
🟩 T1417	0.258	🟩 T1417	0.455
🟩 T1582	0.251	🟩 T1474	0.444
🟥 T1406	0.244	🟩 T1628	0.423
🟩 T1456	0.229	🟩 T1406	0.419
🟩 T1474	0.226	🟩 T1582	0.406
🟩 T1616	0.226	🟩 T1635	0.386
🟥 T1628	0.224	🟩 T1456	0.386
🟥 T1461	0.208	🟩 T1616	0.380
🟩 T1635	0.205	🟥 T1625	0.371
🟥 T1625	0.195	🟥 T1461	0.359
🟥 T1404	0.169	🟩 T1634	0.335
🟩 T1639	0.163	🟥 T1639	0.323

6.4 Identifying Similar Malware and APT Groups

We can use the information aggregated in the knowledge graph to identify similarities between entities, such as malware and threat actors. When a new malware emerges, it is of interest to security analysts to recognize similar malware, as this can provide valuable insight into the malware behavior. Similarly, if a previous APT group launches a new campaign, it may not be apparent initially from the limited information. However, if we can identify past

[5] https://github.com/SigmaHQ/sigma

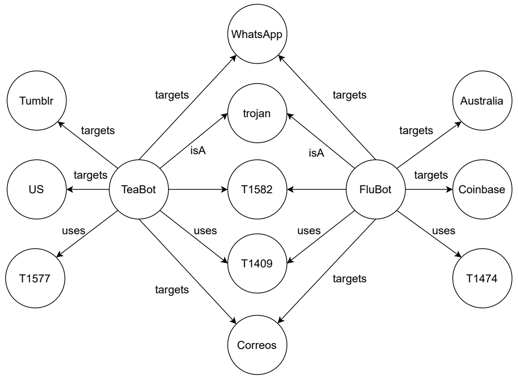


Figure 7: Subgraph on similarity between FluBot & TeaBot

APT groups with similar characteristics, it can help implement preventive measures against such an attack.

In this study, we use the subgraph centered on an entity to find the similarity between malware and threat actors. We construct a set comprising all the adjacent nodes to malware consisting of entity and relation information. We use the Jaccard index [70] of these sets to measure the similarity between a pair of malware. We iterate over all the malware in our larger knowledge graph to identify the most similar one to a given malware. For example, using this approach, TeaBot was identified as the most similar malware to FluBot. Both are Android banking trojans that were active globally in 2021 [7]. We show some of the shared nodes between the two malware in Figure 7. As we can see, both are Android trojans that target banking apps (Correos). Both have the attack pattern– "SMS control mechanism" (ATT&CK ID-T1582) by which they take control of the user's SMS app to interrupt incoming messages and send messages to spread the malware. They also collect stored data from the user's application (MITRE ID-T1409) and target some social media apps like WhatsApp. Some nodes are not shared between the two malware; for example, although FluBot targeted Australia, TeaBot did not. TeaBot targeted the Tumblr app, which FluBot did not.

When identifying similarities between threat actors, we also consider the neighboring nodes of the malware authored by those actors (characterized by the hasAuthor relationship). This aggregates information from all the malware authored by the same threat actor for similarity calculation. Again, we use the Jaccard index to compute the similarity between the connected nodes. For example, when looking for the most similar threat actors to APT15 from the knowledge graph, we found GREF, Boyusec, and Ke3chang. Among these, GREF and Ke3chang have been listed as being associated with APT15 on the MITRE website [40]. This suggests that APT15 has been reported under different names in different CTI reports. The other APT group, Boyusec, shares some similarities with APT15, including developing Android surveillance ware, targeting the same messaging apps like Telegram and locations like Uyghur.

7 RELATED WORK

Our research closely relates to different areas that support threat intelligence: attack pattern extraction on CTI, information extraction from corpus written in natural language, and cyber threat intelligence knowledge graph.

TTP Extraction from CTI: Prior research has primarily centered around building rules or models for searching and analyzing IoCs. 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. Most shared intelligence has been limited to tracking known threat indicators such as IP addresses, domain names, and file hashes [30, 38, 57]. Some studies [3, 8, 56] use internal enterprise logs to capture threat intelligence and generate attack patterns, but this approach is not scalable. Therefore, none of these can be directly compared to our framework for inference-generating threat intelligence knowledge graphs. In [62], Thein et al. propose a neural network approach for classifying sentences of a document into five phases of the cyber kill chain, which represent broad categories for all the phases of a cyberattack. Similarly, TTP-Drill [19] combines dependency parser and heuristics to extract threat actions from a document and maps them to kill chain phases. Luo et al. [32] formulate the event extraction as a sequence tagging task and use a bidirectional LSTM network to learn it. AttackKG [29] designs technique templates in a graph structure for each attack pattern and maps them to individual MITRE ATT&CK IDs using a graph alignment algorithm. Our work stands apart from previous research because we develop a novel machine-learning-based algorithm for TTP extraction from unstructured CTI reports. Since our approach does not rely on predefined static patterns, it can discover new attack pattern descriptions from threat reports that are yet to be included in the MITRE standard, as shown in Section 6.1. Our approach also results in a lower number of false positives when compared to relevant works.

Named Entity Extraction: The work in [9] provides an open-source dataset for cybersecurity named entity extraction. The dataset contains labels for five categories: Version, Application, OS, Vendor, and Relevant. However, the labels were generated automatically using expert rules and were not always accurate. The authors provide three annotated datasets created from different sources – NVD, MS-Bulletin [37], and Metasploit [36]. The annotation F1-score for the three data sources were 87.5%, 77.8%, and 69.1%, respectively. A more recent dataset published in [24] contains four classes of interest – URL, Hash, IP address, and Malware. This dataset was manually annotated; however, it lacks the variety of entities in our ontology. The original work used CNN architecture with LSTM and CRF layers and reported an F1-score of 75.1 %. More recent work on this dataset used transformer-based architecture [53] and achieved an F1-score of 79.8%. Another study in [49] extracted entities from malware after action reports (AAR) with an average F1-score of 77% for 11 different classes. Recent named entity recognition (NER) models have been built to adopt a hybrid approach combining multiple methods for NER [72]. Kim et al. [24] developed a NER system using a deep bidirectional LSTM-CRF network trained on a combination of features. The work in [15] compares neural networks for cybersecurity NER, including CNN, LSTM, BERT, and CRF.

Relation Extraction: To the best of our knowledge, there is no open-source dataset for cybersecurity relation extraction. Early work in [21] 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 eight different relation types. The work in [48] 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, and achieved an average F1-score of 92%. Another study in [18] 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 [53], i.e., where the set of relations is not predetermined, achieved an average F1-score of 59.4%.

Threat Intelligence Knowledge Graph: Knowledge graph construction in cybersecurity is limited compared to other domains, such as biomedical studies [44]. The study in [43] 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 [49], which contains an analysis of various cyber-attacks and is prepared from investigating attacks. This system consists of a custom NER and an entity fusion technique to merge concepts extracted from multiple reports. SEPSES [23] 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. EXTRACTOR [54] uses a similar multi-step approach to ours in the process of creating an attack graph from threat reports for threat hunting from kernel logs. However, EXTRACTOR does not include an attack pattern extraction component in the pipeline. EXTRACTOR mainly includes IoCs as nodes in the knowledge graph, which differs from our goal of capturing tactical threat intelligence for malware behavior analysis.

8 CONCLUSION

We propose LADDER, a framework designed to infer attack patterns along with other threat intelligence for existing and emerging threats. We discuss the challenges in extracting threat information (IoCs and attack patterns) from CTI and identify transformer models that work well on cyber threat datasets. LADDER enables security analysts to proactively gain insights into potential ways an emerging threat, described in a CTI report, can impact their internal enterprise network. LADDER also infers attack patterns, which, combined with other classes, Location, Application, OS, provides strong evidence for a security analyst when confronting an emerging threat. For future work, we plan to enhance the capabilities of the LADDER framework, integrate a temporal analysis component to track the evolution of attack patterns and techniques over time, evaluate and benchmark the framework's performance against large CTI corpus, and improve integration with system logs providing analysts with a unified interface for threat intelligence.

REFERENCES

[1] 2023. LitePower. https://attack.mitre.org/software/S0680/

[2] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 North American Chapter of the Association for Computational Linguistics (NAACL-Demonstrations)*. 54–59.

[3] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. 2021. ATLAS: A Sequence-based Learning Approach for Attack Investigation. In *30th USENIX Security Symposium (USENIX Security 21)*. 3005–3022.

[4] Andy Applebaum, Doug Miller, Blake Strom, Chris Korban, and Ross Wolf. 2016. Intelligent, automated red team emulation. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*. 363–373.

[5] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. (2019), 5185–5194.

[6] Sean Barnum. 2012. Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX). *Mitre Corporation* 11 (2012), 1–22.

[7] Bitdefender. 2022. New FluBot and TeaBot Global Malware Campaigns Discovered. https://www.bitdefender.com/blog/labs/new-flubot-and-teabot-global-malware-campaigns-discovered.

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

[9] Robert A. Bridges, Corinne L. Jones, Michael D. Iannacone, and John R. Goodall. 2013. Automatic Labeling for Entity Extraction in Cyber Security. *CoRR* abs/1308.4941 (2013). arXiv:1308.4941 http://arxiv.org/abs/1308.4941

[10] Jagadeesh Chandraiah. 2018. Red Alert 2.0: Android Trojan. https://news.sophos.com/en-us/2018/07/23/red-alert-2-0-android-trojan-targets-security-seekers/.

[11] Ryan Christian, Sharmishtha Dutta, Youngja Park, and Nidhi Rastogi. 2021. POSTER: An Ontology-driven Knowledge Graph for Android Malware. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2435–2437.

[12] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

[13] Julie Connolly, Mark Davidson, and Charles Schmidt. 2014. The trusted automated exchange of indicator information (taxii). *The MITRE Corporation* (2014), 1–20.

[14] Henry Dalziel. 2014. *How to define and build an effective cyber threat intelligence capability*. Syngress.

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

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[17] Marcel Feller. 2020. Anubis Targets Android. https://cofense.com/infostealer-keylogger-ransomware-one-anubis-targets-250-android-applications/.

[18] Yongyan Guo, Zhengyu Liu, Cheng Huang, Jiayong Liu, Wangyuan Jing, Ziwang Wang, and Yanghao Wang. 2021. CyberRel: Joint Entity and Relation Extraction for Cybersecurity Concepts. In *International Conference on Information and Communications Security*. Springer, 447–463.

[19] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017. TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources. In *Proceedings of the 33rd annual computer security applications conference*. 103–115.

[20] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

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

[22] Kaspersky. 2022. Kaspersky daily. https://usa.kaspersky.com/blog/

[23] Elmar Kiesling, Andreas Ekelhart, Kabul Kurniawan, and Fajar Juang Ekaputra. 2019. The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference,*

Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11779). Springer, 198–214. https://doi.org/10.1007/978-3-030-30796-7_13

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

[25] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. (2015). http://arxiv.org/abs/1412.6980

[26] Alec Koumjian. 2023. datefinder. https://github.com/akoumjian/datefinder.

[27] Max Landauer, Florian Skopik, Markus Wurzenberger, Wolfgang Hotwagner, and Andreas Rauber. 2019. A Framework for Cyber Threat Intelligence Extraction from Raw Log Data. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 3200–3209.

[28] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2022. A Survey on Deep Learning for Named Entity Recognition. *IEEE Trans. Knowl. Data Eng.* 34, 1 (2022), 50–70. https://doi.org/10.1109/TKDE.2020.2981314

[29] Zhenyuan Li, Jun Zeng, Yan Chen, and Zhenkai Liang. 2022. AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports. In *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 13554)*. Springer, 589–609. https://doi.org/10.1007/978-3-031-17140-6_29

[30] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem A. Beyah. 2016. 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, Vienna, Austria, October 24-28, 2016*. ACM, 755–766. https://doi.org/10.1145/2976749.2978315

[31] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[32] Ning Luo, Xiangyu Du, Yitong He, Jun Jiang, Xuren Wang, Zhengwei Jiang, and Kai Zhang. 2021. 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)*. IEEE, 422–427.

[33] Malopedia. 2022. Anubis APK. https://malpedia.caad.fkie.fraunhofer.de/details/apk.anubis.

[34] Malopedia. 2022. Flubot APK. https://malpedia.caad.fkie.fraunhofer.de/details/apk.flubot.

[35] McAfee. 2022. McAfee Blogs. https://www.mcafee.com/blogs

[36] Metaspoilt. 2023. Metaspoilt. https://www.metasploit.com/

[37] Microsoft. 2016. MS-Bulletin. https://docs.microsoft.com/en-us/previous-versions/dn602597(v=msdn.10)?redirectedfrom=MSDN

[38] Sadegh M. Milajerdi, Birhanu Eshete, Rigel Gjomemo, and V. N. Venkatakrishnan. 2019. 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, CCS 2019, London, UK, November 11-15, 2019*. ACM, 1813–1830. https://doi.org/10.1145/3319535.3363217

[39] Doug Miller, Ron Alford, Andy Applebaum, Henry Foster, Caleb Little, and Blake Strom. 2018. *Automated adversary emulation: A case for planning and acting with unknowns*. Technical Report. MITRE CORP MCLEAN VA MCLEAN.

[40] Mitre. 2022. Ke3chang. https://attack.mitre.org/groups/G0004/.

[41] MITRE. 2022. Masquerading. https://attack.mitre.org/versions/v11/techniques/T1036/

[42] MITRE. 2022. MITRE ATT&CK. https://attack.mitre.org/

[43] Sandeep Nair Narayanan, Ashwinkumar Ganesan, Karuna Pande Joshi, Tim Oates, Anupam Joshi, and Timothy W. Finin. 2018. Early Detection of Cybersecurity Threats Using Collaborative Cognition. *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)* (2018), 354–363.

[44] David N Nicholson and Casey S Greene. 2020. Constructing knowledge graphs and their biomedical applications. *Computational and structural biotechnology journal* 18 (2020), 1414–1428.

[45] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33. https://doi.org/10.1109/JPROC.2015.2483592

[46] NIST. 2022. Threat intelligence. https://csrc.nist.gov/glossary/term/threat_intelligence

[47] Lindsey O'Donnell. 2020. Cerberus unleashed. https://threatpost.com/cerberus-banking-trojan-unleashed-google-play/157218/.

[48] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. 2019. RelExt: Relation Extraction using Deep Learning approaches for Cybersecurity Knowledge Graph Improvement. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 879–886.

[49] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. 2020. Creating Cybersecurity Knowledge Graphs From Malware After Action Reports. *IEEE Access* 8 (2020), 211691–211703.

[50] Nidhi Rastogi, Sharmishtha Dutta, Alex Gittens, Mohammed J Zaki, and Charu Aggarwal. 2022. TINKER: A framework for Open source Cyberthreat Intelligence. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 1569–1574.

[51] Nidhi Rastogi, Sharmishtha Dutta, Mohammed J Zaki, Alex Gittens, and Charu Aggarwal. 2020. MALOnt: An Ontology for Malware Threat Intelligence. In *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 28–44.

[52] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 3980–3990. https://doi.org/10.18653/v1/D19-1410

[53] Injy Sarhan and Marco René Spruit. 2021. Open-CyKG: An Open Cyber Threat Intelligence Knowledge Graph. *Knowledge Based System* 233 (2021), 107524.

[54] Kiavash Satvat, Rigel Gjomemo, and VN Venkatakrishnan. 2021. EXTRACTOR: Extracting attack behavior from threat reports. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 598–615.

[55] SecureList. 2021. WIRTE's campaign in the Middle East 'living off the land' since at least 2019. https://securelist.com/wirtes-campaign-in-the-middle-east-living-off-the-land-since-at-least-2019/105044/

[56] Yun Shen and Gianluca Stringhini. 2019. ATTACK2VEC: Leveraging Temporal Word Embeddings to Understand the Evolution of Cyberattacks. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*. USENIX Association, 905–921. https://www.usenix.org/conference/usenixsecurity19/presentation/shen

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

[58] Sudeep Singh and Naveen Selvan. 2023. The Unintentional Leak: A glimpse into the attack vectors of APT37. https://www.zscaler.com/blogs/security-research/unintentional-leak-glimpse-attack-vectors-apt37

[59] Pontus Stenetorp, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou, and Junichi Tsujii. 2012. 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)*.

[60] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. 2018. Mitre ATT&CK: Design and philosophy. In *Technical report*. The MITRE Corporation.

[61] Symantec. 2022. Symantec Enterprise Blogs/Threat Intelligence. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence

[62] Thin Tharaphe Thein, Yuki Ezawa, Shunta Nakagawa, Keisuke Furumoto, Yoshiaki Shiraishi, Masami Mohri, Yasuhiro Takano, and Masakatu Morii. 2020. Paragraph-based Estimation of Cyber Kill Chain Phase from Threat Intelligence Reports. *Journal of Information Processing* 28 (2020), 1025–1029.

[63] ThreatFabric. 2019. Ginp - A malware patchwork borrowing from Anubis. https://www.threatfabric.com/blogs/ginp_a_malware_patchwork_borrowing_from_anubis.html

[64] Wiem Tounsi and Helmi Rais. 2018. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & security* 72 (2018), 212–233.

[65] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.

[66] Andrea Tundis, Samuel Ruppert, and Max Mühlhäuser. 2022. A Feature-driven Method for Automating the Assessment of OSINT Cyber Threat Sources. *Computers & Security* 113 (2022), 102576.

[67] Asahi Ushio and José Camacho-Collados. 2021. T-NER: An All-Round Python Library for Transformer-based Named Entity Recognition. (2021), 53–62. https://doi.org/10.18653/v1/2021.eacl-demos.7

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. *Advances in neural information processing systems* 30 (2017).

[69] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[70] Wikipedia. 2023. Jaccard Index. https://en.wikipedia.org/wiki/Jaccard_index.

[71] Shanchan Wu and Yifan He. 2019. Enriching Pre-trained Language Model with Entity Information for Relation Classification. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 2361–2364.

[72] Feng Yi, Bo Jiang, Lu Wang, and Jianjun Wu. 2020. Cybersecurity Named Entity Recognition Using Multi-Modal Ensemble Learning. *IEEE Access* 8 (2020), 63214–63224.

APPENDIX

Threat Intelligence Concepts

1. Malware is a central concept of the threat intelligence framework. It is malware reported and analyzed in an open-access CTI, like Cerberus. All other concepts are related to it through a set of relationships. For example, Cerberus (class: Malware) is discovered in July, 2019 (class: Time). It "tampers with device functionality and steals banking credentials" (class: Attack Pattern). It targets banking users in Spain (class: Location).

2. Attack Pattern captures the procedures with which malware performs an attack. For example, an adversary may encrypt files stored on the device to prevent the user from accessing them, with the intent of only unlocking access to the files after a ransom is paid. An open-access CTI report consists of many attack patterns explaining the behavior and procedures of an attack. Example: Cerberus *steals victim's bank-account credentials* (attack pattern).

3. Malware Type includes the broad family of malware based on their attack pattern or delivery method. These include, but are not limited to, banking malware, ransomware, adware, spyware, bot, or trojan. Example: *Cerberus* (malware) is a *banking trojan* (MalwareType).

4. Application includes any software product targeted by malware. Applications can be social media applications or specific businesses like banking apps, e-wallets, games, and utility applications. For example: *Cerberus* (malware) targets a *banking apps* (Application).

5. Operating system captures the type of OS and kernel targeted by an instance of the malware. Most common OS like Windows, Linux, Mac, Android, and iOS fall in this category. For example: *Cerberus* (malware) targets *Android* (OS).

6. Organization includes a public or private company targeted by a threat attack. Some entities like Facebook, Google, or Twitter can be identified as applications and organizations. In such cases, we follow the context in which the concept appears in the text. For example: *Cerberus* (malware) targets *Google* (Application) users. *Google* (Organization) has updated its security in the play-store to remove harmful applications.

7. Person is an individual who discovered or analyzed the threat attack. Generally, individuals working in a security company are captured by this class.

8. Time conveys when a particular event related to malware occurred. For example: In *June 2019* (Time), ThreatFabric *Organization* found a new *Android* (OS) malware, dubbed *Cerberus* (malware).

9. Threat Actor is a person or organization acting with malicious intent either with the development or distribution of malware. Such names are specifically mentioned in the report. For example: *Lazarus* is a group associated with *North Korea* (Location) known for *ransomware*(Malware Type) and attacking banks.

10. Location captures the geographical region, country, or city targeted by a threat attack. Example: *Cerberus* (malware) targeted banking users in *Italy*(Location).

11. Indicators of Compromise (IOC) include anything indicating malware's presence and attack pattern. These include email, hashes, IP addresses, file names, domain names, networks, ports, protocols, and URLs. Example: *corona-apps.apk* (IOC) indicates the distribution of Cerberus malware.

In addition to these concepts, we establish relationships between different instances using different relationships:

(1) isA: This relationship classifies a specific malware to a broader family. Example: Cerberus *isA* banking Trojan.
(2) targets: This relationship is between malware and its target, like Malware and Location, Organization, or Application. Example: Cerberus *targets* banking users in Spain.
(3) uses: This relationship is between malware and any entity that it uses to perform an attack, like between Malware and Application, or Malware and AttackPattern. Example: Cerberus *uses* overlay attacks.
(4) hasAuthor: This relationship connects malware to a threat actor who was responsible for developing the malware. Example: Cerberus *hasAuthor* Kilobyte in the dark web.
(5) hasAlias: A malware or threat actor can be identified with a different name. hasAlias captures this relation. This is the only transitive relation, i.e., the head and tail entities are interchangeable. Example: "Malware Marcher *hasAlias* ExoBot" is equivalent to "ExoBot *hasAlias* Marcher".
(6) indicates: This relationship connects an *indicator* to the malware that it represents. Example: Package *com.uxlgtsvfdc.zipvwntdy indicates* Cerberus.
(7) discoveredIn: This relationship connects malware and the time it was discovered in. Example: Cerberus was *discoveredIn* July 2019.
(8) exploits: This relationship connects a Malware and vulnerability (CVE-IDs) it exploited to perform the attack. Example: Cerberus *exploits* XSS vulnerabilities.
(9) variantOf: This relationship connects a Malware to another Malware if one is a variant of another. Example: ERMAC is a *variantOf* Cerberus.
(10) has: This is a broad relationship to capture any connection not explained by other relationships, like between Malware and any other entities.

Web crawler

See Algorithm 1.

Algorithm 1: High-Performance Web Crawler

```
Input: seed_url
Output: web_text, relevant_urls
  scraped_urls ⟵ s
  check for keywords in scraped_urls
  if keyword in url then
    url_queue.add(url)
    web_text.add(text(url))
  end if
  for N in generations do
    for url in url_queue do
      Thread ⟵ scrape(url)
      scraped_urls ⟵ Thread
      check for keywords in scraped_urls
      if keyword in url and url not in url_queue then
        url_queue.add(url)
        web_text.add(text(url))
        relevant_urls.add(url)
      end if
    end for
  end for
```

Regular expressions used to extract IoCs

<table>
<tr><th>Entity Types</th><th>Regular Expression</th></tr>
<tr><td>FilePath</td><td>r’[a-zA-Z]:\\([0-9a-zA-Z]+)’, r’(\/[^\s\n]+)+’</td></tr>
<tr><td>Email</td><td>r’[a-z][_a-z0-9-.]+@[a-z0-9-]+[a-z]+’</td></tr>
<tr><td>SHA256</td><td>r’[a-f0-9]{64}|[A-F0-9]{64}’</td></tr>
<tr><td>SHA1</td><td>r’[a-f0-9]{40}|[A-F0-9]{40}’</td></tr>
<tr><td>CVE</td><td>r’CVE—[0-9]{4}—[0-9]{4,6}’</td></tr>
<tr><td>IPv4</td><td>r’^((25[0-5]|(2[0-4]|1\d|[1-9]|)\d)(\.(?!) |)){4}$’</td></tr>
</table>

Table 13: Regular expressions used to extract IoCs.

Attack Patterns for trojan horse, Cerberus

See Table 14 for a complete list of MITRE attack techniques.

Table 14: Attack Patterns for Cerberus, extracted from CTI sources, mapped to MITRE ATT&CK Framework, and ranked in order of occurrence.

MITRE ID	Name	Description of adversary behavior	Kill-chain Phase
T1461	Lockscreen Bypass	Bypass device lock-screen	1: Initial Access
T1404	Exploitation for Privilege Escalation	Exploit vulnerabilities for elevating privileges	3: Persistence
T1626	Abuse Elevation Control Mechanism	Gain higher-level permissions by taking advantage of built-in control mechanisms.	4: Privilege Escalation
T1406	Obsfucated Files or Information	Encrypt, encode or obfuscate the contents of payload or file	5: Defense Evasion
T1407	Download New Code at Runtime	Download and execute code not included in the original package.	5: Defense Evasion
T1628	Hide Artifacts	Hide adversary artifacts to evade detection	5: Defense Evasion
T1629	Impair Defense	Hinder or disable defensive mechanisms of a device	5: Defense Evasion
T1630	Indicator Removal on Host	Delete, hide, or alter generate adversary artifacts on a device.	5: Defense Evasion
T1516	Input Injection	Mimic user interaction abusing Android’s accessibility APIs	5: Defense Evasion, 12: Impact
T1635	Adversary-in-the-Middle	Position itself between two or more networked devices.	6: Credential Access
T1417	Input Capture	Capture user input to obtain credentials or information	6: Credential Access, 9: Collection
T1517	Access Notifications	Collect notifications sent by OS or applications in a mobile device	6: Credential Access, 9: Collection
T1577	Compromise Application Executable	Modify applications installed on a mobile device	6: Credential Access, 9: Collection
T1430	Location tracking	Track a device’s physical location	7: Discovery, 9: Collection
T1428	Exploitation of Remote Services	Exploit remote services of servers, workstations or other resources to gain unauthorized access.	8: Lateral Movement
T1429	Audio Capture	Capture audio of a mobile device	9: Collection
T1512	Video Capture	Video or image files may be written to disk and exfiltrated later.	9: Collection
T1513	Screen capture	Capture screen to collect additional information about a device.	9: Collection
T1636	Protected user data	Collect data from permission-backed data stores on a device	9: Collection
T1481	Web Service	Use an existing, legitimate web service for transferring data to and from a device	10: Command & Control
T1639	Exfiltration Over Alternative Protocol	Steal data by exfiltrating it over different protocol than the existing C&C.	11: Exfiltration
T1471	Data Encrypted for Impact	Encrypt files on a device to prevent user access	12: Impact
T1582	SMS Control	Delete, alter or send SMS messages.	12: Impact
T1616	Call Control	Make, forward or block phone calls	12: Impact, 9: Collection