Looking Beyond IoCs: Automatically Extracting Attack Patterns from External CTI

Md Tanvirul Alam[1], Dipkamal Bhusal[1], Youngja Park[2], Nidhi Rastogi[1]
[1]Rochester Institute of Technology, USA
{ma8235,db1702,nxrvse}@rit.edu
[2]IBM T. J. Watson Research Center, USA
young_park@us.ibm.com

arXiv:2211.01753v1 [cs.CR] 1 Nov 2022

***Abstract*—Public and commercial companies extensively share cyber threat intelligence (CTI) to prepare systems to defend against emerging cyberattacks. Most used intelligence thus far has been limited to tracking known threat indicators such as IP addresses and domain names as they are easier to extract using regular expressions. Due to the limited long-term usage and difficulty of performing a long-term analysis on indicators, we propose using significantly more robust threat intelligence signals called attack patterns. However, extracting attack patterns at scale is a challenging task. In this paper, we present LADDER, a knowledge extraction framework that can extract text-based attack patterns from CTI reports at scale. The model characterizes attack patterns by capturing phases of an attack in android and enterprise networks. It then systematically maps them to the MITRE ATT&CK pattern framework. We present several use cases to demonstrate the application of LADDER for SOC analysts in determining the presence of attack vectors belonging to emerging attacks in preparation for defenses in advance.**

1. Introduction

Defending against increasingly widespread cyberattacks demands careful unraveling of the attack steps. Knowledge of actions taken by an adversary for a successful attack equips security analysts to "lookout" for similar behavior and improve defense mechanisms against infiltration of networks and systems. MITRE terms this start-to-end strategy that adversaries utilize to compromise the confidentiality, integrity, and availability of assets of an organization as Tactics, Techniques, and Procedures (TTP) [53]. Figure 1 shows examples of TTPs. A proactive defense mechanism against the TTPs requires intercepting the adversary's advance before the exploit stage of the attack. Since APTs (Advanced Persistent Threats) have become very sophisticated, defense strategies must evolve from those based primarily on after-the-fact incident investigation and response to one that integrates cyber threat intelligence. Existing enterprise intrusion detection and prevention systems (IDPS) detect and produce alerts for suspicious events on a host; however, deriving high-level attack patterns of emerging campaigns is of significant interest. Organizations hesitate to share the "procedure" of the attack because they reveal detailed step-by-step descriptions of how an adversary would orchestrate and execute a technique. Therefore, sharing threat tactics and techniques is an excellent middle ground for understanding adversaries and cyber threats uncovered by an organization.

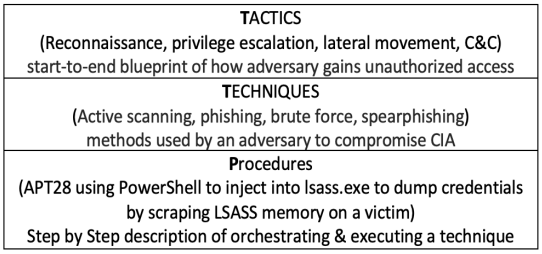


Figure 1. Examples of Tactics, Techniques, and Procedures (TTP) in the MIRE ATT&CK model. Details about the cyberattack increase with "tactics" providing the most abstraction and "procedures" providing the most detail.

After an attack is detected, security analysts rigorously analyze available evidence representing and assessing threats and patterns, which comprises techniques, tools, and procedures (TTPs) of the attacker(s). Called cyber threat intelligence (CTI), this information is disseminated through paid subscriptions or shared freely on blogs, bulletins, news, and reports (open-access CTI). However, there is no clear pathway toward utilizing this information concerning what the security analysts are managing, which is internal threat intelligence in the form of log alerts and anomalous events such as intrusion detection, security log analysis, Denial of Service (DoS) attack detection, and malware detection. When other organizations acquire this CTI, they *still* rely on security experts to comprehend and extract relevant intelligence from the CTI that *is relevant to them*.

To address this gap, we propose a framework, LADDER, for automatically extracting external attack patterns from CTI sources that describe malware and APT attacks, thereby improving threat detection at scale. The philosophy behind the name is akin to exploring and climbing the numerous rungs of attack steps to reach the last rung. Using the attack pattern extraction, classification, and summarization capabilities in LADDER, analysts can learn and analyze attack campaigns for existing or emerging threats from open-access CTI and use that to preempt an emerging attack on their organization. We demonstrate how to map the extracted attack patterns to internal log alerts through several use cases in Section

https://t.me/learningnets

6. LADDER also maps patterns in CTI to pre-defined techniques from the MITRE ATT&CK pattern knowledge base [6], when available in MITRE. Since the ATT&CK framework provides a mitigation and detection plan for every attack pattern technique, our mapping algorithm supports a mitigation plan for experienced and new SOC analysts. Additionally, we also observe patterns not defined by MITRE ATT&CK and show that our model can be used to detect patterns that are yet to be realized.

State-of-the-art for sharing CTI: When a new security threat emerges, current approaches cannot inform of potential attack patterns and corroborating evidence that has taken place over external enterprise networks. Existing services include security information and event management (SIEM) platforms such as Micro Focus ArcSight [14], CTI sharing platforms like IBM X-Force [12], and standards like TAXII, short for Trusted Automated eXchange of Intelligence Information [13]. These services do not take full advantage of the insights that CTI offers, empowering SOC analysts in indicating adversarial motives and their tactics, techniques, and procedures (TTPs). Research efforts so far focus on automating IoCs detection and extraction with the premise that tracking IoCs is equivalent to threat hunting [32]. In other words, recognizing entities that appear in the IoC descriptions in network logs is a sufficient indicator for a specific attack to infiltrate the enterprise successfully. Research has shown that there is little overlap in IoCs shared in CTI from different organizations, and IoCs may sometimes appear much later than when they become active [17]. Therefore, relying on IoCs for threat intelligence is not a long-term solution.

Challenges: Several challenges exist when extracting attack patterns from open-CTI at scale.

1) **Attack Pattern from CTI**: For each attack step that composes the attack, attackers may choose to execute various malicious actions. The attacker can exploit one or more vulnerabilities on the victim system, depending on the exploits available. Therefore, deriving high-level attack steps of an ongoing attack is a significant challenge.
2) **Access to large CTI**: A prevalent approach is to collect massive data for training classification models, including in the security domain. In the past decade, several security organizations have shared CTI with other security analysts; however, many use the paid model. Our empirical study shows that more data is not always needed for attack pattern determination.
3) **Correlating multiple CTI**: Large volume CTI is inevitably noisy and needs to establish: (a) correlation (multiple, sometimes conflicting sources of intelligence on the same attacker), (b) disambiguation (e.g., separating target organization from the attacked organization), and (c) structuring before becoming analyzable. Another challenge is establishing causal inferences based on missing information, context, and domain knowledge.
4) **Unstructured CTI**: The majority of CTI is written in natural language and published in an unstructured and semi-structured format. However, natural language models trained on a generic text corpus may not generalize to the lexical format and semantics used to describe attack information from CTI. For instance, phrase extraction models may label "bank" as an organization. A CTI may refer to "bank" as a banking app compromised by a banking Trojan.

Research Questions: For CTI to be effective, we have identified gaps that need to be addressed to improve their efficacy. CTI should be automated and integrated into current systems while ensuring that relevant evidence is captured in its entirety. However, conventional methods and recent research is geared toward extracting and exchanging IoCs, such as machine-readable malware signatures, email addresses, and domain names. Attack patterns provide more valuable insights when integrated with attack vector-specific IoCs [24]. Our first research question is *how to extract attack patterns from unstructured threat reports reliably?*. Our second research question is to *identify the plausibility of malware attack patterns given the existing evidence available about them*. In the context of a SOC analyst, it is crucial to not only determine existing telemetries from CTI and system logs but also predict attacker behavior, given limited information about its attack vectors. E.g., what kind of vulnerability or system exploit will a new malware use to gain access to sensitive data?

Main Contributions: The main contributions are summarized below with a detailed technical description in Section 3:

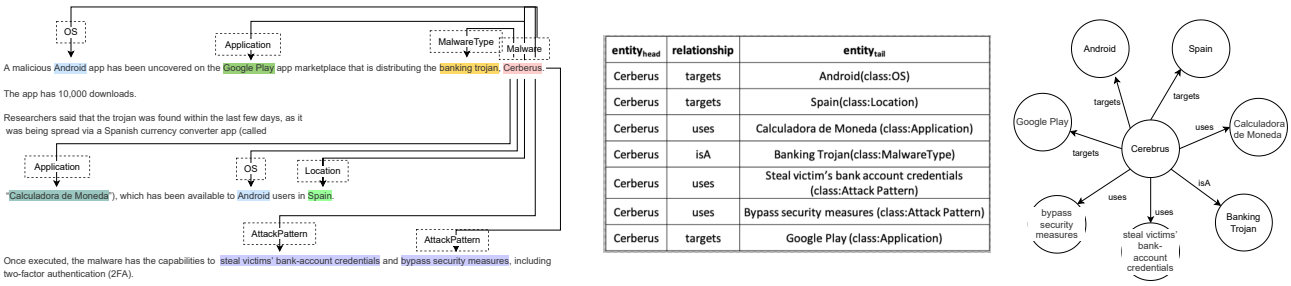
1) We propose a threat intelligence aggregation and analysis framework, LADDER, which extracts and aggregates attack patterns and other telemetries as evidence of attacks found in diverse, unstructured CTI. LADDER also classifies them to ATT&CK pattern techniques described by MITRE.
2) To demonstrate the effectiveness of LADDER and its security application for SOC analysts, we extract attack patterns with high accuracy and infer the next attack step from known information. The proposed knowledge graph-based framework predicts the subsequent attack patterns, including tactics and techniques, vulnerabilities, the system under attack, and locations for two malware– Anubis and FluBot.
3) We provide a new, open-access benchmark android malware dataset to train future CTI models. We share approximately 3,021 unique malware threat intelligence triples comprising a unique combination of nine entities (head and tail) and five relationships to connect the entities. This set of triples is hand-annotated on a corpus of 150 CTIs [1].

2. Motivating Example

To illustrate our approach, we describe the CTI of malware Cerberus – a trojan horse that targets Android mobile phone banking credentials. We use the same example throughout the paper for consistency. The Cerberus CTI describes how the malware works and how to recognize its tactics, techniques, and procedures (called attack patterns). The CTI also covers vulnerabilities of specific business technologies, such as email, domains and mobile devices. Consider the following snippet from the CTI on

[1] https://github.com/aiforsec22/IEEEEuroSP23

https://t.me/learningnets



$entity_{head}$	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.

Cerberus written by ThreatPost [2]:
"...A malicious Android app has been uncovered on the Google Play app marketplace that is distributing the banking Trojan, Cerberus. The app has 10,000 downloads. Researchers said that the trojan was found within the last few days, as it was being spread via a Spanish currency converter app (called "Calculadora de Moneda"), which has been available to Android users in Spain. Once executed, the malware has the capabilities to steal victims' bank-account credentials and bypass security measures, including two-factor authentication (2FA)..."

In Figure 2, we show the transition from an unstructured threat analysis report to a structured knowledge graph for "Cerberus". The entity extraction (e.g. application), triple generation ⟨malware, targets, application⟩, and knowledge graph construction (all triples combined) are detailed in Section 3.1. 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 class definitions for entities– Malware, Attack Pattern, Location, OS, Application (see Section 3.2), are mapped to existing threat intelligence ontology classes [19], [45] but modified for CTI and security logs, and also follow the STIX2.1 framework. Relationships between them are predefined in the same ontology.

We extract triples from open-access CTI sources to learn and train a threat intelligence model using past malware information, including attack patterns. Cerberus uses a total of 29 attack patterns which we capture from several CTI reports (accurate numbers can be tricky to compute) using the proposed the LADDER framework module that extracts attack patterns– TTPClassifier. Due to space constraints, a small part of the entire list is provided in Table 1. The complete list is in Appendix 8. The triple formed for attack patterns are of the form: $\langle \text{Cerberus}, uses, AttackPattern \rangle$. For example, $\langle \text{Cerberus}, uses, \text{"}Steal\ bank\ account\ credentials\text{"} \rangle$. TTPClassifier also maps the attack pattern written in natural language to the ATT&CK ID: T1636.

As discussed in Section 6, using TTPClassifier, an analyst receives a list of phrases that are attack patterns written in natural language, along with a mapping to ATT&CK IDs and corresponding descriptions. As discussed in Section 6: use cases, an analyst can map these attack patterns based on external telemetries. For example, an analyst can map these patterns to internal security logs using existing or newly created sigma rules [51]. In Section 6, we demonstrate that the analyst can query our knowledge graph using the attack pattern extracted from the CTI. The graph "infers" the attack patterns that the same malware may attempt but have not been reported or observed yet. Using this knowledge, an analyst can take preemptive measures and stop or deter adversaries from causing damage to the internal network. Knowing the MITRE ATT&CK also benefits the analyst since the mitigation approaches are available for each attack pattern, allowing even a less experienced security analyst to take timely measures.

3. Approach Overview - LADDER

The main idea of our proposed framework LADDER is that gathering past threat intelligence, like attack patterns, can train a threat intelligence model for (a) detecting semantically similar attack patterns occurring in threat analysis reports and (b) inferring attack tactic or technique for an emerging attack vector based on the available information. Our team of researchers analyzed many malware attack pattern steps and concluded that most of them can be mapped to the MITRE ATT&CK pattern framework. While each step's granular, low-level information is detailed, all attack patterns can be abstracted to a high-level definition. Additionally, LADDER captures additional information, such as locations, indicators of compromise (IoCs), and system information which can help identify the similarity between different malware and threat actors.

In this section, we provide a summary of the LADDER framework and modules. LADDER comprises four modules for using threat intelligence from CTI when they are structured and analyzed by a knowledge graph-based system. LADDER aggregates unstructured threat and attack information from diverse CTI sources and provides actionable analysis to security analysts (see Figure 3). Once trained and ready for use, a security analyst submits a corpus of unstructured threat information into LADDER. The trained system returns inferred intelligence like most machine learning models. However, instead of simply classifying the ingested data, LADDER returns classes along with matching instances. The predefined classes are - Malware, Malware Type, Application, Operating System, Organization, Person, Time, Threat Actor, Location, Attack Pattern. Nine of the ten classes return a list of classes and their respective instances found in the CTI. See Figure 3 for a high-level view of the framework

https://t.me/learningnets

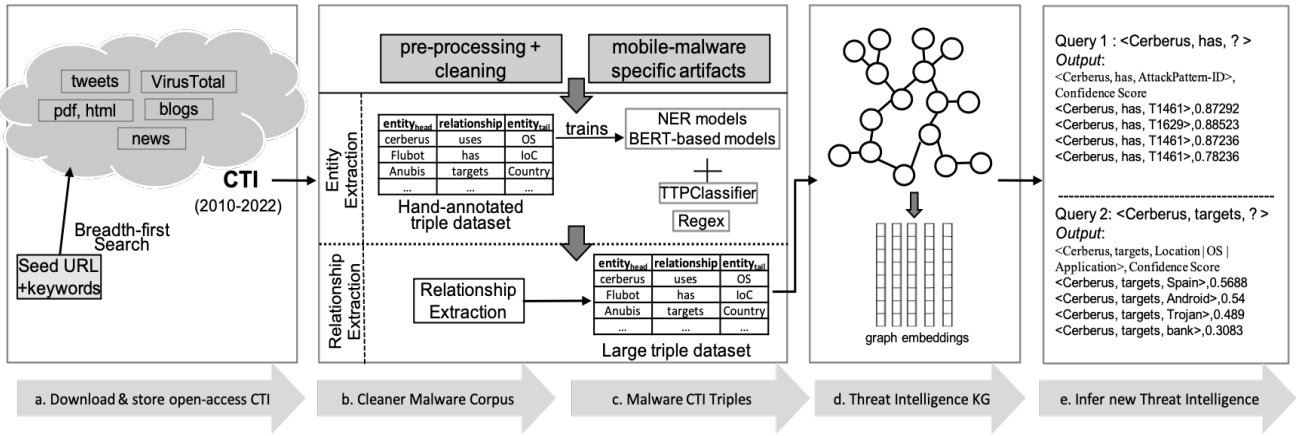


Figure 3. Proposed framework: LADDER and the five component modules. (a) Extracts CTI using crawlers, (b) Pre-processes and prepares for entity and relationship extraction, (c) generates data in the form of triples, (d) creates a knowledge graph by combining the triples and uses graph embedding methods 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.

and its various modules. We describe each of these in detail next.

3.1. Cyberthreat Intelligence (CTI)

CTI is usually available in the form of after-action reports [43], unstructured blogs, security bulletins [7], and technical reports. A standard approach to examining CTI reports involves (a) perusing the most prevalent techniques, trends, and threats observed by the author(s), (b) detecting, mitigating, and simulating specific threats and techniques, and (c) mapping ideas, guidance, and priorities to their security measures and general strategy. Analysts can track down and defend against known attacks using IoCs. More importantly, information like tactics, techniques, and procedures written in semantic format is assembled to characterize a threat attack for analysis and future predictions. We design a high-performance web crawler (details in appendix) that scrapes, collects, and stores over 90k unstructured threat report from 2010 to 2022. These threat reports explain the attack behavior of malware and APTs targeted at various platforms such as windows, Linux, Android, and iOS operating systems. Out of these over 12K reports had descriptions for mobile malware.

3.2. Concepts of threat intelligence

We extract threat intelligence concepts in the form of a triple, $\langle entity_{head} - relationship - entity_{tail} \rangle$. Also known as a semantic triple, it codifies a statement in a threat report about the malware in an RDF format, $\langle subject, predicate, object \rangle$ expressions. The triple entity is representative of one of ten classes from CTI reports– Malware, Malware Type, Application, Operating System, Organization, Person, Time, Threat Actor, Location, Attack Pattern. These triples aggregate in a graph called the threat intelligence knowledge graph, transforming the multi-modal open-access CTI into a structured format. Security analysts can use this graph to systematically *query* known threats and emerging threats and forecast trends and attack patterns using evidence in the form of existing knowledge in the graph. The concepts are detailed in the Appendix.

3.3. Generating Threat Intelligence Graph

Open-access CTI reports are passed into information extraction models pre-trained from triples collected from hand-annotated CTI reports. Hence, the extraction of relevant entities and asserting the relationship between entities is the key procedures to generate a graph. Traditionally information graphs were constructed using pre-existing

TABLE 1. **PARTIAL** LIST OF ATTACK PATTERNS FOR CERBERUS, EXTRACTED FROM CTI SOURCES, MAPPED TO MITRE ATT&CK FRAMEWORK, AND RANKED IN ORDER OF OCCURRENCE. THE FULL LIST IS IN APPENDIX.

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
T1577	Compromise Application Executable	Modify applications installed on a mobile device	6: Credential Access, 9: Collection
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
T1582	SMS Control	Delete, alter or send SMS messages.	12: Impact
T1616	Call Control	Make, forward or block phone calls	12: Impact, 9: Collection

https://t.me/learningnets

databases constructed by domain experts. However, for cybersecurity, we can utilize information extraction techniques and machine learning to collect facts from unstructured text sources and populate the graph.

3.4. Inferring Attack Patterns

The information extraction module also comprises an algorithm we propose– TTPClassifier that automates attack pattern extraction from CTI reports. In Figure 2(b), TTPClassifier extracts all known malware attack behaviors and guides us to a list of attack behaviors not seen before. The triple formed from this attack behavior is $\langle Cerberus, uses, conduct\ covert\ surveillance \rangle$.

Inferring attack patterns using LADDER works as follows: (a) **Training:** The knowledge graph comprising attack behaviors, classes, and relationships of all known, existing, and dormant malware are extracted from CTI sources using TTPClassifier and other information extraction models. Additionally, TTPClassifier also maps these behaviors to MITRE ATT&CK pattern IDs. (b) **Inferring:** When inferring new threat intelligence from an emerging threat, the security analyst queries the knowledge graph by inputting a triple, where the tail entity is missing. The LADDER framework infers all blank tail entities, including attack behaviors and all other classes, along with a confidence score.

In TTPClassifier, the attack behavior is mapped to an existing MITRE ATT&CK techniques. MITRE ATT&CK provides a comprehensive list of threat tactics and techniques based on real-world malware attacks. The list consists of a detailed description of attack techniques that different cybercriminals engage in. It also provides corresponding mitigation measures for each attack pattern technique. For example, a total of 66 attack techniques are listed for mobile malware. See Table 15 in the Appendix for an exhaustive list of attack IDs and corresponding descriptions.

4. System Design

4.1. Dataset Collection

To the best of our knowledge, no publicly available dataset comprises triples captured from the CTI of a cyber threat. Available open-access CTIs are not consistent or reliable in describing malware attacks. To ensure the highest quality triples for LADDER, we collected a carefully curated set of 150 threat reports. These reports are collected from 36 Android malware like *Cerberus, Rotexy, Judy, Gooligan, SpyNote RAT* mentioned in the MITRE website. Attack patterns from many of these reports exist, alibi paraphrased and mapped to the MITRE ATT&CK framework. The reports explain the malware's emergence, propagation, attack patterns, and IoCs in natural language and are written by security analysts from organizations such as McAfee, Symantec, and Kaspersky.

We manually annotated different threat concepts and their relationships as explained in Section 3.2 using the BRAT annotation tool [52]. Figure 4 (l) shows the distribution of the ten classes in these 150 documents. Some instances of concepts may belong to different classes

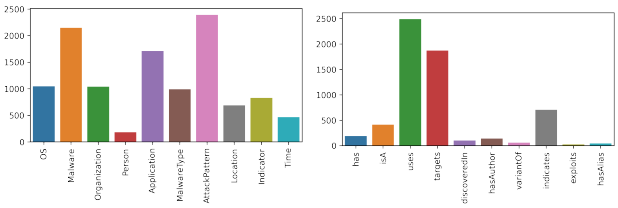


Figure 4. Entity (l) and relationships (r) distribution in the corpus

depending on the context they appear in. For example, *"Facebook", "Twitter"*, and *"Instagram"* may either belong to the class Organization if the text refers to those organizations directly or Application if the text describes their application for a platform. We annotate attack patterns as concepts, but they capture large phrases and even entire sentences compared to other types that comprise one or few words. As a result, an attack pattern may include other concepts within it. In such instances, we annotate the larger text as an attack pattern and the smaller concept mentioned inside the text as well. For example, the attack pattern *"break Android's application sandbox"* also contains the annotation for *Android* as *OS*.

See Figure 4 (r) for distributions of relationships. *Uses* has the highest count because it links attack patterns with the malware. *Indicates* is the relationship that connects indicators of compromise with the malware that they represent. *Targets* is a relationship between Malware and ThreatActor and other type of entities. *isA* draws a relationship between a broader category or family of the malware.

4.2. Information Extraction from CTI for Threat Intelligence Graph

Crawling Open-access CTI sources: Our high-performance web crawler scraped and collected over 12k unstructured open-access CTI reports from public URLs. To ensure high-quality reports, we only scraped threat reports from security companies, websites of technology companies, and technology news reporting companies. The crawler performs a breadth-first search (BFS) given a seed URL. To ensure good quality reports, the seed URL always belongs to a security company or a technical CTI website. The crawler stores all URLs mentioned on the starting page. It then goes through all these URLs for text evaluation to determine if it is a relevant page. A relevant page has specific descriptions of malware (presence of malware-related keywords within the first n words of the article, where n¿100). If the page is relevant, the URL, along with its text, is saved. The starting page will thus lead to relevant pages, more relevant pages, and so on. Once the reports are downloaded, a text cleaner processes the content by removing HTML tags and images. We separately extracted the time information using a heuristic-based approach from the reports for longitudinal analysis. Many threat reports contain their publication date within the first few sentences of the report, typically after the title. We used the NER model provided in the Flair[2] NLP library and extracted the first *DATE* entity found in the

[2]https://github.com/flairNLP/flair

https://t.me/learningnets

top five sentences, if any. We then verified the date and extracted the year using python datefinder library[3]. We used threat reports published between 2010-2021. Our industry collaborator provided over 1 million unstructured CTI crawled from the Internet to increase the scale of the study. In addition, we also included 80k documents scraped from a corporate security lab. We detail the steps for web crawling in Algorithm 1 in Appendix.

Extracting and Labeling concepts (entities): Once the threat reports are pre-processed and cleaned, we extract different classes of entities using state-of-the-art natural language processing techniques. For classes including Malware, AttackPattern, Application, OS, Organization, Person, Time, Location, we fine-tune Transformer-based [59] pretrained language model with our hand-annotated dataset. We adopt this approach since such large-scale language models are highly effective in many downstream NLP tasks where the amount of labeled data is limited [22]. Extracting evolving security concepts like attack patterns and malware requires an understanding of the context, which the attention mechanism provides. We use the *tner* [57] library to fine-tune the models. Specifically, we use three transformer variants in our experiments: BERT [22], RoBERTa [33] and XLM-RoBERTa [20]. XLM-RoBERTa is a multilingual model, while the rest are pre-trained on English corpus. We take the hidden layer representation from the transformer model and add a classification layer that predicts the entity for each position. The classification layer consists of 10 neurons corresponding to the nine entity classes and the special no entity (*O*) token representing that a token does not belong to any entity classes. In contrast, some cybersecurity concepts like URL, IP address, email, and file name are extracted using pattern matching [44], [63]. We use a heuristic-based approach with pattern matching to extract different indicators, including URL, IP address, hash, and CVE ID. Some example regex patterns are shown in a Table in the Appendix.

4.3. Attack Pattern Extraction

Attack patterns differ significantly from other entities in terms of extraction and inference. They are not extracted using the approach mentioned in the previous section for other entities since attack patterns consist of a larger block of contiguous text and do not represent a single *named entity* but rather an action taken by a cyber-threat. Sometimes attack patterns may include other entity types in their description. For example, consider the sentence below:

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

Here, the attack pattern phrase *"capable of generating an instance of TeamViewer on mobile"* contains another entity *TeamViewer* of type Application within it.

We propose a novel approach for extracting attack patterns from threat reports to account for these and call it TTPClassifier. Our proposed approach consists of three sub-tasks: relevant sentence extraction, attack phrase extraction, and mapping attack patterns to MITRE ATT&CK. These are described next.

[3]https://github.com/akoumjian/datefinder

(a) Relevant sentence extraction. In the first step, we identify sentences in the threat report that contains a description of one or more attack patterns. We formulate this as a binary sentence classification task, i.e., the sentences containing one or more attack patterns are considered positive and the rest negative. During training, the positive sentences comprise all the annotated sentences containing attack patterns. We randomly sample the same number of negative instances from the remaining sentences to create a balanced dataset. We fine-tune pre-trained transformer models for the task. We add a linear layer consisting of two neurons for the classification task. The linear layer is preceded by an additional hidden layer for RoBERTa models.

(b) Attack phase extraction. The second step of the algorithm identifies the relevant parts of the sentence containing attack pattern descriptions for the sentences predicted as positive (i.e., containing at least one attack pattern). We predict the tag of each word using a sequence tagging model similar to those for extracting other entities for this task. We have two classes: whether or not a word is part of an attack pattern description. Although some sentences contain a single attack pattern description, others may contain more than one attack pattern. For example, consider the following sentence. *"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: 1) *"covertly send and steal SMS codes"* 2) *"open tailored overlays for various online banks"* 3) *"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. Using post-processing, we discard potentially invalid extractions, e.g., those that do not contain verbs.

(c) Mapping attack patterns. The last step in TTPClassifier is mapping each attack pattern to standardized MITRE ATT&CK techniques. MITRE has both techniques and sub-techniques, but we do not consider the sub-techniques in this study and map each extracted sequence to one of the techniques. As the number of classes is large and the annotation effort is significant to match an attack sequence to its MITRE ID, we adopt a semantic similarity-based approach for the mapping task. Using a pre-trained sentence transformer model [46], we compute the embeddings for extracted attack pattern phrase. We use embeddings from both the title and description of the MITRE ATT&CK ID described on the website. Sometimes CTI reports mention an attack pattern that resembles the title very closely. However, other times, we need the description to identify the matching technique. We compute a metric – weighted distance between the extracted phrase and a MITRE 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 embedding vector for the extracted attack phrase, , v^i_{title} and v^i_{desc} are the embedding vectors for the i^{th} MITRE ATT&CK respectively. cos represents the cosine distance between two vectors u, v computed as

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

https://t.me/learningnets

We iterate over all the different attack patterns present for a platform (there are 66 techniques for mobile platforms and 193 for 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.

Since threat reports are unstructured in nature, the same attack pattern may be described in different ways in different threat reports. If we use these as individual nodes in the knowledge graph, there will be much redundant information, and predicting attack patterns will be very challenging. Mapping to MITRE IDs allows us to have a fixed number of attack patterns in the knowledge graph.

Even though we train our model with threat reports on mobile platforms, we can also use the trained model to extract attack patterns for other platforms. The first two steps of our algorithm are platform-agnostic and can provide us with attack pattern descriptions as they appear in the text. We can change the mapping steps with the appropriate list of attack patterns present in MITRE to get attack patterns for other platforms. We demonstrate this in Section 6, where we extract 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 [19], [45]. 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 is based on the work proposed by [62] that incorporates entity information for relation classification. Specifically, given a text s with a pair of entities e_1 and e_2, we introduce four special tokens that capture the position information of the entities. For example, given the sentence:

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 as follows *[CLS]* $\langle e1 \rangle$ *Cerberus* $\langle /e1 \rangle$ *is capable of generating an instance of* $\langle e2 \rangle$ *TeamViewer* $\langle /e2 \rangle$ *on mobile.*

Here *[CLS]* is the special start of sequence token for the BERT model. We take the hidden layer representation for the start position of both entities and concatenate them to generate the final vector embedding. We pass the embedding vector through a couple of fully connected layers to predict the relation type between the pair of entities. Once the triples are generated from the large corpus, the threat intelligence knowledge graph is generated similarly to the hand-created triple dataset. The knowledge graph is now ready for querying.

4.5. Querying LADDER

With the threat intelligence graph 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. Formally, the threat intelligence graph is a directed multi-modal 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. 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 $e_{tail} \in \mathcal{E}$. Knowledge graph 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 **h** given $\langle$**?, r, t**$\rangle$ or **t** given $\langle$**h, r, ?**$\rangle$, where "**?**" indicates a missing entity.

For the entities and relationship e_{head}, e_{tail}, r we use the bold face **h, t, r** to indicate their low-dimensional embedding vectors, respectively. We use d_e and d_r to denote the embedding dimensions of entities and relations, respectively. The symbols n_e, n_r, and n_t denote the number of entities, relations, and triples of the KG. We use the f_ϕ notation for the scoring function of each triple $\langle$**h, r, t**$\rangle$. The scoring function measures the plausibility of a fact in $\mathcal{T}$, based on translational distance or semantic similarity [30].

Vector Embeddings: All triples have three vectors, $\mathbf{e_1}, \mathbf{e_2} \in \mathbb{R}^{d_e}$, $r \in \mathbb{R}^{d_r}$. Triples are represented by relatively low (e.g., 30-200) dimensional vector space embeddings [16], [60], which preserves information about the structure and key features. The embeddings for all the triples in the KG involve factorization of co-occurrence-based tensors [16].

Using the link prediction approach, LADDER infers the "missing" $entity_{tail}$. Given a knowledge generated using the modules described in LADDER, link prediction infers the missing triple by learning a scoring function. We use TuckER [16] for inferring the entity concepts in our threat intelligence graph since it outperforms traditional link prediction models like RESCAL, DistMult, ComplEX, ConvE, and HypER. The comparison is out of scope for this paper. TuckER is a linear model based on tucker decomposition [56] of entity embedding matrix and relational embedding matrix in a knowledge graph. It can predict entities in all relationship types (symmetric, asymmetric, transitive) and thus is considered fully expressive. In addition to the inference, it also evaluates the accuracy and recommends a ranking of inferred entities.

5. Experiments and Results

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

5.1. Information Extraction

We use the PyTorch [41] deep learning framework to implement the models for the information extraction and subsequent tasks. We fine-tune the pretrained transformer models available in Huggingface's transformers library [61]. We use a sequence length of 128 for training the NER models. We train the models for 20 epochs using 32

https://t.me/learningnets

TABLE 2. 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 3. 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

samples per mini-batch. We use a learning rate of 1e-5 for the base models and 1e-6 for the large models and optimize the models using the AdamW optimizer [34].

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

5.1.1. Entity Extraction. We show the results for the entity extraction task using different transformer models in Table 2. XLM-RoBERTa large model performs the best for this task with an average F1 score of 78.98%. We show the class-specific results for this model in Table 3. We generally obtain better results for classes with more samples in the training dataset. Operating System and Location yield better results than the other classes as they do not have much variation in forms. Application and Organization have some overlap between them (e.g., Facebook and Twitter can be both Application and Organization) which reduces the performance. The most challenging class appears to be the ThreatActor. This is because ThreatActor is usually an Organization or Person with malicious intent. So, this requires a thorough understanding of the context to detect them properly. Having a limited input length means this may not be possible to infer from a single sentence. There are also not enough samples for this class in the training data, further reducing the performance.

This section compares our work and prior work on cybersecurity named entity recognition. The work in [18] 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 automatically generated using expert rules and were not always accurate. The authors provide three annotated dataset created from different sources – NVD, MS-Bulletin [37] and Metasploit [36]. They reported the annotation F1-score by randomly sampling and hand-annotating a subset of the dataset. The

TABLE 4. 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 5. 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

F1-score for the three data sources were 87.5%, 77.8%, and 69.1%, respectively. Since the ground truth annotation was inaccurate, the results obtained in those datasets do not indicate how models trained on them would work in practice. A more recent dataset was published in [27] containing four major classes of interest – URL, Hash, IP address, and Malware. This dataset was manually annotated; however, it lacks the variety of entities we included 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 [47] and achieved an F1-score of 79.8%. Although the topics covered in this dataset differ from our dataset, this result is comparable to the average F1-score on our dataset – 78.98%. Another work in [43] extracted entities from malware after action reports (AAR) with an average F1-score of 77% for 11 different classes, including Campaign, Software, Tool, IP Address, etc.

5.1.2. Attack Pattern Extraction. We use the same malware split as in the NER task for attack pattern extraction. There are usually more sentences in a report that do not contain an attack pattern description than those that do. So, we randomly sample the same amount of negative sentences as the number of sentences that include attack patterns to create a balanced dataset for the sentence classification task. We fine-tune the transformer models for sentence classification and attack phrase extraction subtasks. We use a sequence length of 256 to train these models. We use a mini-batch size of 32, and the optimal learning rate from [1e-5, 5e-5, 1e-6], optimized using the validation set. We train the sentence classification models for 20 epochs and the attack phrase extraction models for 30 epochs. The models are trained using the Adam optimizer [28].

We show the result for the binary sentence classification task in Table 4. We get greater than 85% F1-score for all the models with RoBERTa large performing best with an average F1-score of 89.62%. The results for the attack phrase extraction task are in Table in 5. Here also, we obtain the best result using the RoBERTa-large model, with an average F1-score of 90.64%. These results suggest that our algorithm can detect the relevant sentences for attack patterns from the text and extract the part of the

https://t.me/learningnets

TABLE 6. 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 7. 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

sentence that mentions the attack patterns with a high degree of accuracy.

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

5.2. Relation Extraction

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

We show the aggregate result for relation extraction in Table 6. The BERT-based model outperforms the other two models for this task. The best result is obtained using the BERT-large model with a 92.62% average F1-score. We show the results for specific classes in Table 7. We do not include attack patterns for the relation extraction task as they are part of a single relation type (*Malware uses AttackPattern*). So, we can infer the relation for attack pattern once we identify the malware under discussion. Although *discoveredIn* performed well for the annotated corpus, it was primarily because the reports were cleaned and usually did not contain redundant time information of malware discovery. Among other classes, *uses* and *hasAlias* performs much worse. Context is challenging with $uses$ as it is usually confused with *targets* since both contain the same type of head-tail entity pairs (e.g., Malware and Application. Relationship between two malware (*variantOf* and *hasAlias*) is difficult to detect since they may be expressed at a distant position in report. We note that these results are obtained on the manually cleaned test dataset, and the performance drops on more noisy texts.

To the best of our knowledge, there is no open-source dataset for cybersecurity relation extraction. An early work in [25] used semi-supervised learning with a bootstrapping algorithm for extracting the relation between security entities. They achieved an average F1-score of 82% on a dataset containing 8 different relations. The work in [42] used a word embedding model for relation extraction in cybersecurity texts. They considered six different types of relations – hasProduct, hasVulnerability, uses, indicates, mitigates, related-to. They achieved an average F1-score of 92%, which is comparable to ours. Another work in [23] 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 [47], i.e., where the set of relations is not predetermined, achieved an average F1-score of 59.4%.

5.3. Threat Intelligence Knowledge Graph

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

We also include triples from Virus Total (VT) to create a larger knowledge graph to evaluate link prediction performance in the presence of more IoCs. VT provides academic API to access intelligence on a database of several million malware samples. We use the VT API V3.0 [10] to extract IoCs (contacted domains and IP addresses) and permission requests for each malware hash collected from a larger corpus of OpenCTI reports. We map each permission to attack pattern and MITRE ID using TTPClassifier and evaluate attack pattern prediction from querying the larger knowledge graph.

5.4. Inferring entities with TuckER

We create two different test sets from the hand-annotated documents with a varying number of triples for the link prediction task. We considered triples involving Malware e.g., AttackPattern, Location, Application,

https://t.me/learningnets

TABLE 8. INFERENCE (LINK PREDICTION) RESULTS FOR DIFFERENT TRAINING AND TEST DATASETS

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

TABLE 9. CLASS-SPECIFIC INFERENCE (LINK PREDICTION) RESULTS.

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

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

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

We show the results for the inference (link prediction) task in Table 8. For TestSet 1, we have similar performance using the different training datasets for all Hits@(n) values. However, for TestSet 2, KG1 performs much worse than KG2. We have a difference of 16.4% @Hits 10 between KG1 and KG2. This suggests that the additional triples obtained from a larger knowledge graph enable the model to make better predictions. The performance for KG2 is similar on the two test datasets suggesting better generalization ability. Adding additional triple from VirusTotal in KG3 decreases performance slightly compared to KG2, suggesting that the inclusion of additional IoCs may not help with the inference (link prediction) task. This is most likely because IoCs are usually unique across different malware.

We show class-specific prediction result in Table 9 using KG1 and KG2 for three different tail entities - AttackPattern, Location and Application where the head entity is Malware. Here the most promising result is obtained for predicting AttackPattern. The primary reason is that we have 66 unique attack patterns, but the number of possible tail entities is much larger for relations involving other classes. This result suggests that malware with similar properties may exhibit similar attack patterns. Similar to the aggregate result above, we see no significant drop in performance for the two test datasets for KG2 as opposed to KG1, where there is a significant drop.

TABLE 10. 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 [24]	22	43	231	0.09	0.34	0.14
AttackKG [31]	12	53	85	0.12	0.18	0.15
TTPClassifier	41	24	22	0.65	**0.63**	0.64

6. Case Studies

1. Attack Pattern Extraction and Trend Analysis: One of our research questions is how reliably we can extract attack patterns from unstructured CTI reports.

We compare our attack pattern extraction with TTPDrill [24] and AttackKG [31] as they are the closest to the proposed work, alibi there are differences that we discuss in the related work section. We use the open-source implementation of TTPDrill and AttackKG in our evaluation. When analyzing the attack patterns listed in the MITRE ATT&CK website for malware, *we found that often not all attack patterns reported in a CTI are present. This indicates the difficulty of this task, even for human annotators*. Since TTPDrill and AttackKG provide models and patterns for the enterprise platform, we use the same for evaluation. Even though we trained our sentence classification and phrase extraction models for attack patterns on CTI gathered on mobile platforms, unsurprisingly our evaluations show high accuracy when testing CTI for other platforms as well 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. We create ground truth annotation from five reports listed on the MITRE ATT&CK website for five different malware. We show the result in Table 6. 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 large 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 or when there is not enough example pattern available for an attack pattern.

We show a case study involving one of the five malware– LitePower[4] malware in Table 11. We extract 18 different attack patterns from the CTI report of this malware [48]. 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

[4]https://attack.mitre.org/software/S0680/

https://t.me/learningnets

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	✓

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 was matched with the sub-technique *System Checks*, which starts with the description *Adversaries may employ various system checks to detect.* This description was highly 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 relatively higher similarity with it. One major 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 that there is yet no attack pattern in MITRE mobile platforms for $Masquerading$ which is included for enterprise platforms [39]. 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 [55], this malware was *masquerading as a "Google Play Verificator" app*. Our approach can give an analyst the summarized version of a threat report with a mention of attack steps used by malware. We show **three example attack patterns** extracted by our algorithm 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.

Large Scale Malware Behavior Analysis: We use our TTPClassifier to extract attack patterns for 433 malware in 12K threat reports (these were found to have relevant threat intelligence after filtering from over 1 million threat reports) to perform attack pattern trend analysis. We only counted unique attack patterns for the same malware if they are mentioned in multiple reports and obtained 3159 attack patterns in the time period of 2015-2021. We map the attack patterns to MITRE attack technique IDs and plot the distribution of attack IDs against time. We show the plot for three different trends in Figure 5(r). See Figure 5(l) for a plot of 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 not a representation of attack patterns deployed by malware in the wild.

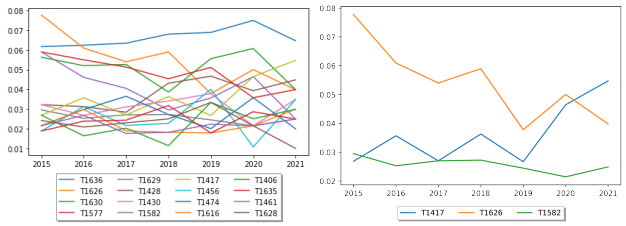


Figure 5. Distributions of all (l) and three (r) attack techniques vs. time. The X-axis represents the year when an attack technique was observed in the CTI, and Y-axis represents the normalized count of that attack pattern (ratio of the count of an attack technique observed in that year to the total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

In Figure 5(r), we can observe an upward trend of T1417 (Input Capture) with peak usage in 2021. This attack technique encompasses any methods that an adversary uses to steal the application credentials of users. Adversaries can use keylogging or GUI capture methods to steal user input. For example, Anubis malware has a keylogger that works in every application installed on the device [11]. One possible reason for the increase in the use of this attack technique is the increased use of mobile-based digital solutions like banking, finance, and shopping over the years. We expect this attack technique to be used more often in the future.

We observe a downward trend of T1626 (abuse elevation control mechanism). T1626 encompasses methods used by an adversary to grant itself high-level permissions in a device. For example, Red Alert 2.0 malware can request device administrator permissions [9]. One reason for the reduction in this attack technique could be that users are now more cautious of applications that ask for device administration requests. Secondly, Android OS 7+ has introduced changes that make abuse of administrator privilege more difficult. As more devices get updated with the latest OS, it will be more difficult for adversaries to manipulate users to gain elevated access to devices and use them for malicious purposes.

T1582 (SMS Control) has 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 [11]. This trend suggests that SMS phishing is as widespread on the mobile platform as before.

2. Threat Hunting: Automating the process of attack pattern extraction can aid in threat hunting and protection against APT campaigns. Correlating attack patterns

https://t.me/learningnets

Figure 6. Two actual logs for the attack pattern T1222– File and Directory Permissions Modification. Texts in blue highlight represent the relevant part of the logs for the attack pattern.

extracted from CTI with kernel logs can pinpoint an attacker's activities. However, the same APT campaign may manifest in different forms in different settings due to differences in OS, targeted applications, variations of the threat, and so on. As a result, relying on IoCs for exact threat hunting is unreliable since attackers can modify them to evade detection [29]. We illustrate this with some sample logs from the DARPA Transparent Computing Engagement 3 dataset in Figure 6. These logs are collected from an attack on a FreeBSD server by exploiting an Nginx backdoor vulnerability. These logs represent the attack pattern T1222– File and Directory Permissions Modification which was used as a precursor for creating a new elevated process (attack pattern T1548). The accompanying description provided in the ground truth report was *ability to create a new elevated process* and the interactions were of the form $F1 > elevate/tmp/XIM$. As we can see, there are variations in the logs due to the process ID, process name, and the file that is being modified. So, instead of matching the exact IoC, which may be different from that described in a threat report for a particular setting, it is more reliable to use high-level abstract information like attack patterns in conjunction with IoCs for identifying attacks. Extracting attack patterns from kernel logs is beyond the scope of this study. There is some ongoing open-source effort for creating rules for MITRE attack pattern extraction from logs– for example, the Sigma rules. However, these rules do not yet have sufficient coverage for 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 different enterprise attack patterns in MITRE, there are sigma rules for 60 of them for Linux resulting in a coverage of 31%. As a result, many attack patterns extracted from CTI may go undetected in the log. Regardless, improvement in attack pattern extraction from logs can allow analysts an efficient way of identifying specific APT attacks in kernel logs and is one of our future research goals.

3. Attack Pattern Prediction: Information extracted from a single CTI often does not contain complete information about a specific malware. Consolidating data from multiple reports allows us to bridge that gap and provides a more holistic view of any cyber attack. However, it is still possible that some specific malware characteristics are not captured in the reports analyzed, resulting in missing information in the knowledge graph. We can find such missing information using the link prediction task, 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 not been captured yet. We can use the link prediction task as a proxy to predict such attack patterns, which malware may use in the near future.

TABLE 12. ATTACKPATTERN PREDICTION FOR ANUBIS AND FLUBOT MALWARE. SORTED BY CONFIDENCE SCORE(GREEN: OBSERVED PATTERNS,RED : ATTACK PATTERN NOT OBSERVED.)

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

We conduct a study of *AttackPattern* prediction for two malware - Anubis and Flubot. Anubis is an Android banking trojan active from 2017 until late 2021, featuring a wide range of attack techniques [1]. Flubot is another banking malware that has been targetting Android users since the first quarter of 2021 and has been active ever since [3]. We identified 33 unique attack patterns for Anubis and 23 for Flubot from the reports collected during these periods. We use triples from the KG2 (see Table **??**) graph for the prediction task. We remove *AttackPattern* triples for making prediction (i.e., when predicting for Anubis we remove triples of the form *Anubis uses T1636*) and use TuckER to query the tail entity for the *uses* relation e.g., $\langle Anubis, uses, ? \rangle$.

We show the top 20 attack patterns predicted for the two malware with confidence scores in Table 12. As we include more predictions, it invariably includes less confident outputs resulting in more false positives. However, we see promising results for the top predictions. We get 9 correct predictions for Anubis and 7 correct predictions for Flubot out of the top 10. When we extend to the top 15, the number of correct predictions is 12 for both malware. These results suggest a strong correlation exists between different malware behavior. We can use a knowledge graph to leverage this for predicting unknown behavior, viz., infer future attack patterns from the available information. We can also use this for filling up missing information in existing knowledge graphs.

4. Identifying Similar Malware and APT Groups: We can use the information aggregated in the knowledge graph to find similarities between entities such as malware

https://t.me/learningnets

and threat actors. When a new malware emerges, it is of interest to security analysts to identify similar malware, as it can provide valuable insight into the malware behavior. Similarly, if some previous APT group launches a new campaign, it may not be apparent initially from the limited information. If we can identify past APT groups with similar characteristics, it can help take preventive measures against such an attack.

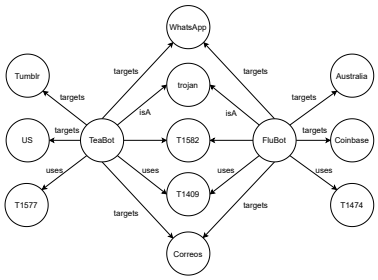


Figure 7. Subgraph showing similarity between FluBot and TeaBot malware (not all nodes are included)

We use the subgraph centered on an entity to find the similarity between malware and threat actors in this study. We construct a set consisting of all the nodes adjacent to malware consisting of entity and relation information. We use the Jaccard index [4] of these constructed set 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, TeaBot was identified as the most similar malware to FluBot using the approach. Both are Android banking trojans that were active globally in 2021 [8]. We show some shared nodes extracted from the knowledge graph between these 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 (MITRE 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 different malware authored by the same threat actor for similarity calculation. We again use the Jaccard index for computing the similarity between threat actors. 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 [5]. This suggests that APT15 have been reported under these names in different threat reports. The other APT group, Boyusec, shares some similarities with APT15, including developing Android surveillance ware, targetting messaging apps like Telegram, and common locations like Uyghur.

7. Related work

LADDER is closely related to three areas that support threat intelligence, especially attack pattern inference using open-access CTI, information extraction from unstructured and semi-structured corpus, and prediction. **CTI:** Prior research has centered around building rules or models for searching and analyzing IoCs. Most shared intelligence has been limited to tracking known threat indicators such as IP addresses, domain names, and file hashes [32], [38], [50]. [15], [17], [49] use internal enterprise logs to capture threat intelligence, and produce attack patterns, which is not scalable. Therefore none of these can be directly compared to our framework and inference-generating threat intelligence knowledge graphs. The limitation of this approach is that there is little to no overlap in the shared information, and no long-term analysis is possible using this intelligence. Also, none of this research captures detailed attack patterns in a standardized format nor utilizes CTI reports for analysis. In [54], Thein et al. propose a neural network approach for classifying sentences of a document into five phases of the cyber kill chain, which are broad categories for all the phases of a cyberattack. Similarly, [24] combines dependency parser and heuristics to extract threat actions from a document and map them to kill chain phases. Luo et al. [35] formulate the event extraction as a sequence tagging task and uses a bidirectional LSTM network to solve it. **Discovering and Extracting Threat concepts:** [18] implement a maximum entropy model for automatically labeling cybersecurity concepts. However, more recently, named entity recognition (NER) models have been built to adopt a hybrid approach combining multiple methods for NER [63]. Kim et al. [27] built a NER system using a deep bidirectional LSTM-CRF network trained on a combination of features. The work in [21] provides a comparative study of different neural networks, including CNN, LSTM, BERT, and CRF, for cybersecurity NER. OpenCTI reports come from diverse sources and therefore ambiguity and duplication in threat concepts are research worthy problems. Word embeddings have been used for entity disambiguation in [58] by clustering entities based on their similarity in the embedding space. **Threat Intelligence Knowledge Graph:** Knowledge graph construction in the cybersecurity domain is limited compared to other domains such as biomedical studies. The study in [40] implements a collaborative framework with the help of semantically rich knowledge representation for 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 [43], which contains an analysis of various cyber-attacks and is prepared from the investigation of attacks. This system consists of a custom NER and an entity fusion technique to merge concepts extracted from multiple reports. SEPSES [26] is a cybersecurity KG populated from multiple heterogeneous cybersecurity data sources and frequently updated. An Extraction, Transformation, Loading (ETL) periodically checks and updates the KG as new security information becomes available.

https://t.me/learningnets

8. Conclusion

We propose LADDER, a framework to infer attack patterns along with other threat intelligence for emerging threats. Security analysts can use this framework to preemptively learn about potential ways an emerging threat, described in open-access CTI, can impact their internal enterprise network. We describe the challenges encountered in extraction threat information from CTI and models that work well for cyberthreat dataset. We infer attack pattern steps, which along with other classes - Location, Application, OS provide strong evidence to a security analyst when a new attack emerges.

References

[1] Anubis apk. https://malpedia.caad.fkie.fraunhofer.de/details/apk.anubis.

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

[3] Flubot apk. https://malpedia.caad.fkie.fraunhofer.de/details/apk.flubot.

[4] Jaccard index. https://en.wikipedia.org/wiki/Jaccard_index.

[5] Ke3chang. https://attack.mitre.org/groups/G0004/.

[6] Mitre. https://attack.mitre.org/.

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

[8] New flubot and teabot global malware campaigns discovered. https://www.bitdefender.com/blog/labs/new-flubot-and-teabot-global-malware-campaigns-discovered.

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

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

[11] Anubis targets android. https://cofense.com/infostealer-keylogger-ransomware-one-anubis-targets-250-android-applications/, 2021.

[12] Ibm x-force security services. https://www.ibm.com/security/xforce, 2022.

[13] Introduction to taxii. https://oasis-open.github.io/cti-documentation/taxii/intro.html, 2022.

[14] Security information and event management tool: Cyberres. https://www.microfocus.com/en-us/cyberres/secops/arcsight-esm, 2022.

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

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

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

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

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

[20] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

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

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186. Association for Computational Linguistics, 2019.

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

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

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

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

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

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

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

[30] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[31] Zhenyuan Li, Jun Zeng, Yan Chen, and Zhenkai Liang. Attackg: Constructing technique knowledge graph from cyber threat intelligence reports. In *European Symposium on Research in Computer Security*, pages 589–609. Springer, 2022.

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

[33] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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

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

[36] Metaspoilt. Metaspoilt.

[37] Microsoft. Ms-bulletin.

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

https://t.me/learningnets

[39] MITRE. Mitre-masquerading.

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

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

[42] Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 879–886, 2019.

[43] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating cybersecurity knowledge graphs from malware after action reports. *IEEE Access*, 8:211691–211703, 2020.

[44] Aritran Piplai, Sudip Mittal, Anupam Joshi, Tim Finin, James Holt, and Richard Zak. Creating cybersecurity knowledge graphs from malware after action reports. *IEEE Access*, 8:211691–211703, 2020.

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

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

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

[48] SecureList. Wirte's campaign in the middle east 'living off the land' since at least 2019.

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

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

[51] SigmaHQ. Sigmahq/sigma: Generic signature format for siem systems.

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

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

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

[55] ThreatFabric. Ginp - a malware patchwork borrowing from anubis.

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

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

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

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

[60] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

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

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

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

https://t.me/learningnets

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 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, iOS fall in this category. For example: *Cerberus* (malware) targets *Android* (OS).

6. Organization includes a company, public or private, targeted by a threat attack. Some entities like Facebook, Google, or Twitter can be identified as both 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 analysed 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 cities, targeted by a threat attack. Example: *Cerberus* (malware) targeted banking users in *Italy*(Location).

11. Indicators of Compromise (IOC) include anything that indicates the presence of a malware and its attack pattern. These primarily include email, hashes, ip addresses, file name, domain name, network, port, protocol, URL. Example: *corona-apps.apk* (IOC) indicates the distribution of Cerberus malware.

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

1) isA: This relationship classifies a specific malware to a broad 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 Attack-Pattern. Example: Cerberus *uses* overlay attacks.
4) hasAuthor: This relationship connects a malware to a threat actor who was responsible for developing the malware. Example: Cerberus *hasAuthor* Kilobyte in 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 a malware and 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 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.

Regular expressions used to extract IoCs

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

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.

MITRE ATT&CK Patterns for android devices

See Table 15 for a complete list of MITRE attack techniques.

https://t.me/learningnets

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 mechanism.	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

Web crawler

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
```

https://t.me/learningnets

TABLE 15. MOBILE ATTACK TECHNIQUES IN MITRE ATT&CK FRAMEWORK

MITRE ID	Name	Description of adversary behavior	Kill-chain Phase
T1626	Abuse Elevation Control Mechanism	Gain higher-level permissions by taking advantage of built-in control mechanism.	4: Privilege Escalation
T1517	Access Notifications	Collect notifications sent by OS or applications in a mobile device.	6: Credential Access 9: Collection
T1640	Account Access Removal	Remove access of users to accounts by deleting, locking or manipulating access.	12: Impact
T1638	Adversary-in-the-Middle	Position itself between two or more networked devices.	9: Collection
T1437	Application Layer Protocol	Communicate using application layer protocol to avoid detection.	10: Command & Control
T1532	Archive Collected Data	Compress or encrypt data before exfiltration.	9: Collection
T1429	Audio Capture	Capture audio of a mobile device.	9: Collection
T1398	Boot or Logon Initialization Scripts	Execute scripts at boot or logon initialization to establish persistence.	3: Persistence
T1616	Call Control	Make, forward or block phone calls.	12: Impact 9: Collection
T1414	Clipboard Data	Abuse clipboard manager to access information copied to clipboard.	6: Credential Access 9: Collection
T1623	Command and Scripting Interpreter	Abuse command and script interpreter to execute binaries or scripts.	2: Execution
T1577	Compromise Application Executable	Modify applications installed on a mobile device.	3: Persistence
T1645	Compromise Client Software Binary	Modify system software binaries to establish access to device.	3: Persistence
T1634	Credentials from Password Store	Use password storage locations to steal user credentials.	6: Credential Access
T1471	Data Encrypted for Impact	Encrypt files on a device to prevent user access.	12: Impact
T1533	Data from Local System	Search local file system to find files of interest.	9: Collection
T1641	Data Manipulation	Insert, delete, or modify data to manipulate external outcomes or hide activity.	12: Impact
T1407	Download New Code at Runtime	Download and execute code not included in the original package.	5: Defense Evasion
T1456	Drive By Compromise	Gain access to a system by user's compromised activity like visiting a website.	1: Initial Access
T1637	Dynamic Resolution	Establish connection to C&C infrastructure dynamically.	10: Command & Control
T1521	Encrypted Channel	Employ encryption to hide C&C traffic.	10: Command & Control
T1642	Endpoint Denial of Service	Perform DoS attacks to block the availability of service.	12: Impact
T1624	Event Triggered Execution	Establish persistence using mechanisms that trigger execution based on events.	3: Persistence
T1627	Execution Guardrails	Constrain execution or actions based on adversary supplied conditions.	5: Defense Evasion
T1639	Exfiltration Over Alternative Protocol	Steal data by ex-filtrating it over different protocol than the existing C&C.	11: Exfiltration
T1646	Exfiltration Over C2 Channel	Steal data by exfiltrating it over existing C&C channel.	11: Exfiltration
T1404	Exploitation for Privilege Escalation	Exploit vulnerabilities for elevating privileges.	3: Persistence
T1428	Exploitation of Remote Services	Exploit remote services of servers, workstations or other resources to gain unauthorized access.	8: Lateral Movement
T1420	File and Directory Discovery	Search files in device for desired information.	7: Discovery
T1541	Foreground Persistence	Maintain continuous sensor access in a device.	3: Persistence 5: Defense Evasion
T1643	Generate Traffic from Victim	Generate outgoing traffic from devices.	12: Impact
T1628	Hide Artifacts	Hide adversary artifacts to evade detection.	5: Defense Evasion
T1625	Hijack Execution Flow	Hijack application execution and execute their own payloads.	3: Persistence
T1617	Hooking	Utilize hooking to hide the presence of adversary artifacts.	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
T1544	Ingress Tool Transfer	Transfer tools and files from external system onto a device.	10: Command & Control
T1417	Input Capture	Capture user input to obtain credentials or information.	6: Credential Access 9: Collection
T1516	Input Injection	Mimic user interaction abusing Android's accessibility APIs.	5: Defense Evasion 12: Impact
T1430	Location tracking	Track a device's physical location.	7: Discovery 9: Collection
T1461	Lockscreen Bypass	Bypass device lockscreen.	1: Initial Access
T1575	Native API	Write native functions using Android's Native Kit to achieve binary execution.	2: Execution 5: Defense Evasion
T1464	Network Denial of Service	Perform DoS attack to degrade or block the targeted resources.	12: Impact
T1423	Network Service Scanning	Receive a listing of services running on remote hosts.	7: Discovery
T1509	Non-Standard Port	Generate network traffic using a protocol and port paring that are typically not associated.	5: Defense Evasion
T1406	Obfuscated Files or Information	Encrypt, encode or obfuscate the contents of payload or file.	5: Defense Evasion
T1644	Out of Band Data	Communicate using out of band data streams.	10: Command & Control
T1424	Process Discovery	Get information on running processes on a device.	7: Discovery
T1631	Process Injection	Inject code into processes to evade process based defense.	3: Persistence 5: Defense Evasion
T1636	Protected user data	Collect data from permission-backed data stores on a device.	9: Collection
T1604	Proxy Through Victim	Use a device as a proxy server to the internet.	5: Defense Evasion
T1458	Replication Through Removable Media	Copy malware to devices connected vis USB for lateral movement.	1: Initial Access 8: Lateral Movement
T1603	Scheduled task/job	Abuse task scheduling functionality to execute malicious code.	2: Execution
T1513	Screen capture	Capture screen to collect additional information of a device.	9: Collection
T1582	SMS Control	Delete, alter or send SMS messages.	12: Impact
T1418	Software Discovery	Get a list of all applications installed on a device.	7: Discovery
T1635	Steal Application Access Token	Steal user tokens to acquire credentials to access systems and resources.	6: Credential Access
T1409	Stored Application Data	Access and collect application data on a device.	9: Collection
T1632	Subvert Trust Controls	Undermine security controls.	5: Defense Evasion
T1474	Supply Chain Compromise	Manipulate products delivery mechanisms prior to receipt by a consumer.	1: Initial Access
T1426	System Information Discovery	Get detail information about a device.	7: Discovery
T1422	System Network Configuration Discovery	Get details about network configuration and settings.	7: Discovery
T1421	System Network Connections Discovery	Get details about network connections to and from device.	7: Discovery
T1512	Video Capture	Capture video recordings using device camera.	9: Collection
T1633	Sandbox Evasion/Virtualization	Employ means to detect and avoid analysis environments.	5: Defense Evasion
T1481	Web Service	Use an existing legitimate web service for transferring data to and from a device.	10: Command & Control

https://t.me/learningnets