AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports

Zhenyuan Li[1], Jun Zeng[2], Yan Chen[3], and Zhenkai Liang[2]

[1] Zhejiang University, Hangzhou, China
[2] National University of Singapore, Singapore
[3] Northwestern University, Evanston, USA

Abstract. Cyber attacks are becoming more sophisticated and diverse, making detection increasingly challenging. To combat these attacks, security practitioners actively summarize and exchange their knowledge about attacks across organizations in the form of cyber threat intelligence (CTI) reports. However, as CTI reports written in natural language texts are not structured for automatic analysis, the report usage requires tedious manual efforts of threat intelligence recovery. Additionally, individual reports typically cover only a limited aspect of attack patterns (e.g., techniques) and thus are insufficient to provide a comprehensive view of attacks with multiple variants.

To take advantage of threat intelligence delivered by CTI reports, we propose AttacKG to automatically extract structured attack behavior graphs from CTI reports and identify the associated attack techniques. We then aggregate threat intelligence across reports to collect different aspects of techniques and enhance attack behavior graphs into technique knowledge graphs (TKGs).

In our evaluation against real-world CTI reports from diverse intelligence sources, AttacKG effectively identifies 28,262 attack techniques with 8,393 unique Indicators of Compromises. To further verify the accuracy of AttacKG in extracting threat intelligence, we run AttacKG on 16 manually labeled CTI reports. Experimental results show that AttacKG accurately identifies attack-relevant entities, dependencies, and techniques with F1-scores of 0.887, 0.896, and 0.789, which outperforms the state-of-the-art approaches (Extractor [37] and TTPDrill [25]). Moreover, our TKGs directly benefit downstream security practices built atop attack techniques, e.g., advanced persistent threat detection and cyber attack reconstruction.

1 Introduction

Advanced cyber attacks have been growing rapidly. The trend of attacks is to adopt increasingly sophisticated tactics and diverse techniques [13], such as multi-stage Advanced Persistent Threats (APTs), making detection more challenging than ever. To combat these attacks, security analysts actively exchange threat intelligence to enhance detection capabilities.

Among them, structured threat intelligence defined by open standards (e.g., OpenIoC [12], STIX [9], and CybOX [4]) are widely shared on open-source platforms (e.g., AlienVault OTX[1] and IBM X-Force[8]) and utilized in security operation centers. Such intelligence standards define cyber attacks as Indicators of Compromises (IoCs), which are artifacts in forensic intrusions, such as MD5 hashes of malware samples and IP/domains of command-and-control (C&C) servers. However, recent studies have shown that detection with disconnected IoCs is easy to bypass [29,33]. For example, attackers can frequently change domains used in attack campaigns to evade detection. In comparison, by taking IoC interactions into account, graph-based detection typically demonstrates better robustness [34] by identifying attack techniques aligned to adversarial goals. Specifically, attack techniques [24,14] are basic units that describe "how" attack actions are performed and are well used in security solutions (e.g., endpoint detection and response systems).

Attack techniques can be found in unstructured CTI reports written by security practitioners based on their observations of attack scenarios in the wild. In particular, a well-written report precisely describes attack behaviors through enumerating attack-relevant entities (e.g., *CVE-2017-11882*) and their dependencies (e.g., `stager` connecting to `C&C sever`). However, recovering attack behaviors from textual CTI requires non-trivial manual efforts. Intuitively, a system capable of automatically extracting attack technique knowledge from CTI reports can significantly benefit cyber defenses by reducing human efforts and accelerating attack responses. We identify two key challenges in the automation of knowledge extraction from CTI reports: (1) As CTI reports are written in an informal format, in natural languages, identifying structured attack behaviors needs to analyze semantics in unstructured CTI texts; (2) Attack knowledge is dispersed across multiple reports. Individual reports commonly focus on limited/incomplete attack cases, making it difficult to obtain a comprehensive view of attacks. And existing works on CTI report parsing [37,25,23,26,36] only focus on attack cases within a single report.

In this paper, we propose AttacKG , a novel approach to aggregate threat intelligence across CTI reports and construct a knowledge-enhanced attack graph that summarizes attack-technique-level workflows in CTI reports. Based on enhanced knowledge, we introduce a new concept called technique knowledge graph (TKG), which identifies causal techniques from attack graphs to describe complete attack chains in CTI reports. More specifically, we first adopt a pipeline to parse a CTI report and extract attack entities and their dependencies as an attack graph. Then, we initialize technique templates using attack graphs built upon technique procedure examples crawled from the MITRE ATT&CK knowledge base [14]. Next, we utilize a revised graph alignment algorithm to match attack graphs from CTI reports and technique templates. Towards this end, we can align and refine attack entities in both CTI reports and technique templates. While technique templates aggregate specific and potentially new intelligence from real-world attack scenarios, attack graphs can leverage such knowledge from templates to construct TKGs.

We implement AttacKG and evaluate it against 7,373 procedures of 179 techniques crawled from MITRE ATT&CK and 1,515 CTI reports collected from multiple intelligence sources [3,5]. Our experimental result demonstrates that AttacKG substantially outperforms existing CTI parsing solutions such as EXTRACTOR [37] and TTPDrill [25]: (1) With our CTI report parsing pipeline, AttacKG accurately constructs attack graphs from reports with F1-scores of 0.887 and 0.896 for entities and dependencies extraction, respectively; (2) Based on extracted attack graphs, AttacKG accurately identifies attack techniques with an F1-score of 0.789; (3) AttacKG successfully collect 28,262 techniques, and 8,393 unique IoCs from 1,515 CTI reports.

To the best of our knowledge, this is the first work to aggregate attack knowledge from multiple CTI reports at the technique level. In particular, our work makes the following contributions:

- We present a new pipeline for CTI report parsing with better efficiency and effectiveness in constructing attack graphs.
- We propose the design of technique templates to describe and collect technique knowledge, and a revised graph alignment algorithm to identify attack techniques with templates. By aligning templates with technique implementations described in attack graphs, we exchange the knowledge from both to refine each other and form technique knowledge graphs (TKGs).
- We implement AttacKG (open sourced[4]) and evaluate it with 1,515 real-world CTI reports. The results demonstrate that AttacKG accurately extracts attack graphs from reports and effectively aggregates technique-level threat intelligence from multiple unstructured CTI reports. We also show TKGs' benefits with two case studies.

2 Background and Related Work

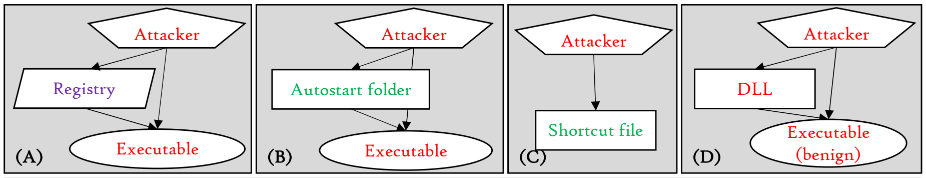
In this section, we first introduce the outbreaking attack mutations. Then, we introduce state-of-the-art threat intelligence extraction solutions. Finally, we present a real-world CTI report as a motivating example for intuitive illustration.

2.1 Cyber Attacks and Reports

Attackers actively create attack variants to evade detection. To systematize and summarize the behaviors in attack variants, MITRE proposed the ATT&CK Tactics-Techniques-Procedures (TTP) matrix based on real-world observations of cyber attacks. In the hierarchical TTPs matrix, tactics describe "why" an adversarial action is performed, which are typically fixed for an attack, while the selections and implementations of techniques that describe "how" to perform the adversarial action are more flexible.

As Figure 1 shows, the commonly used technique "*T1547-Boot or Logon Autostart Execution*" for tactic "*Persistent*" can be implemented in at least four different ways: (A) "*Registry Run Keys*", (B) "*Auto-start folder*", (C) "*Shortcut Modification*", and (D) "*DLL Side-loading*". The number of variants grows

[4] To facilitate follow-up research, we release the source code of AttacKG at https://github.com/li-zhenyuan/Knowledge-enhanced-Attack-Graph.



(A) To be started during the boot process of the infected machine, the malware creates the following registry key: *HKCU\Software\Classes\CLSID\{42aedc87-2188-41fd-b9a3-0c966feabec1}\InprocServer32* = *%APPDATA%\shdocvw.tlp*.
(B) Confucius has dropped malicious files into the startup folder *%AppData%\Microsoft\Windows\Start Menu\Programs\Startup* on a compromised host in order to maintain persistence.
(C) S-Type may create the file *%HOMEPATH%\Start Menu\Programs\Startup\Realtek {Unique Identifier}.lnk*, which points to the malicious *msdtc.exe* file already created in the %CommonFiles% directory.
(D) This results in the user seeing only the *Flash_Adobe_Install.exe* file to execute in order to install what they believe to be an update to Flash Player. When run, it will automatically load *goopdate.dll* due to search order hijacking.

Fig. 1: Technique template generated by AttacKG and corresponding real-world description for "*T1547-Boot or Logon Autostart Execution*" with four variants corresponding to fourteen MITRE sub-techniques categorized as (A) *Registry Run Keys*, (B) *Auto-start folder*, (C) *Shortcut Modification*, and (D) *DLL Side-loading*.

exponentially if detailed implementations, such as different selections of the registry key, are taken into consideration. Our observation is that while the ways of technique implementation are relatively limited, the implementation details are much more varied. Therefore, it is reasonable to believe that a system that can identify various attack techniques and collect the corresponding implementation details would significantly benefit downstream security tasks (e.g., intrusion detection and attack forensics).

The manually-crafted TTP matrix cannot cover various technique implementations. Such detailed implementation knowledge only comes from practice. Thus, security practitioners actively gather and share attack knowledge as threat intelligence. Such cyber threat intelligence (CTI) is typically managed and exchanged in the form of either structured and machine-digestible Indicators of Compromise (IoCs) or unstructured and natural language reports.

2.2 Threat Intelligence Extraction

Cyber threat intelligence (CTI) plays a vital role in security warfare to keep up with the rapidly evolving landscape of cyber attacks [35,21]. To facilitate CTI knowledge exchange and management, the security community has standardized open formats (e.g., OpenIoC [12], STIX [9], and CybOX [4]) to describe Indicators of Compromises (IoCs). Though structured and machine-readable, such intelligence lacks semantic information about how IoCs interact to form attack chains. While in this paper, we try to fulfill this semantic gap with attack-technique-level knowledge aggregated across CTI reports.

Poirot [33] utilizes manually extracted and generalized (attack) query graphs for intrusion detection in provenance graphs constructed from system logs, which validates the efficacy of threat intelligence in detection. However, manually extracting attack-relevant information from uninstructed texts is labor-intensive and error-prone, hindering CTI's applications in practice. Therefore, several ap-

Table 1: Comparison of threat intelligence extraction methods

	Automatic	Graph-structure	Technique-aware	Cross-reports
Poirot [33]	✗	✓	✗	✗
iACE [32]	✓	✗	✗	✗
Extractor [37] & ThreatRaptor [22]	✓	✓	✗	✗
TTPDrill [25] & rcATT[28], etc.	✓	✗	✓	✗
AttacKG	✓	✓	✓	✓

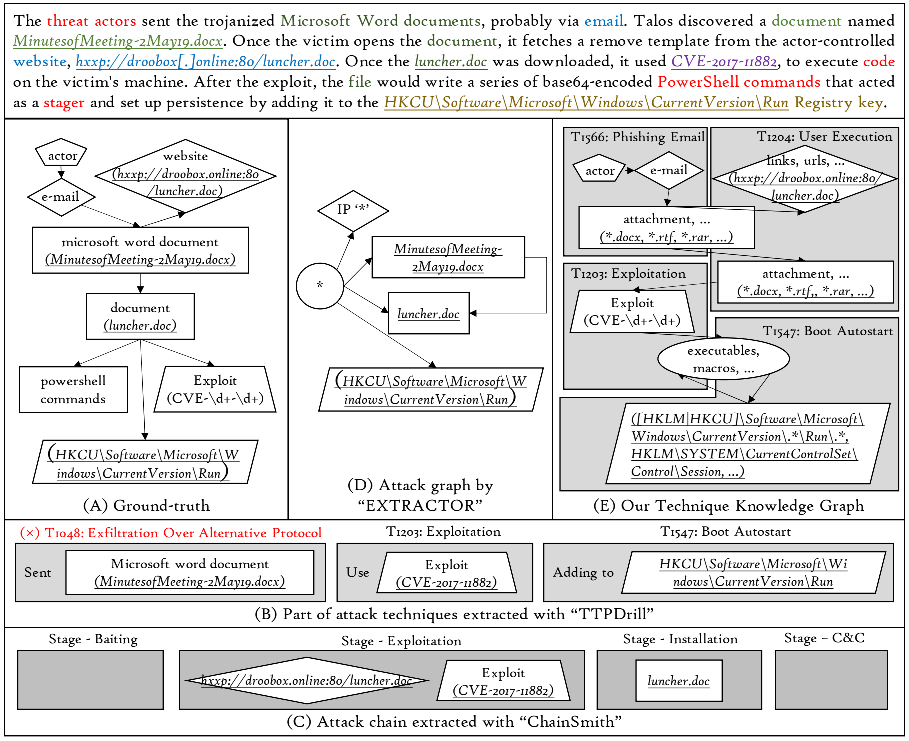
proaches have been proposed to analyze CTI reports automatically. As Table 1 shows, these works can be roughly divided into several categories. Specifically, iACE [32] presents a graph mining technique to collect IoCs available in tens of thousands of security articles. Extractor [37] and ThreatRaptor [22] customize NLP techniques to model attack behaviors in texts as attack graphs. TTPDrill [25], rcATT [28] and ChainSmith [39] derive threat actions from reports and map them to attack patterns (e.g., tactic and techniques in MITRE ATT&CK [14]) with pre-defined ontology or machine learning techniques. Similar to prior studies, the large body of AttacKG is to automate attack knowledge extraction from CTI. Nevertheless, AttacKG distinguishes itself from these works in the sense that it identifies TTPs and constructs technique knowledge graphs (TKGs) to summarize technique-level knowledge across CTI reports.

2.3 Motivating Example

Figure 2 presents a real-world APT attack campaign called Frankenstein [7]. The campaign name comes from the ability of its threat actors to piece together different independent techniques. As shown, this campaign consists of four attack techniques, namely, *T1566-Phishing E-mail*, *T1204-User Execution*, *T1203-Exploitation* and *T1547-Boot Autostart*. Each technique involves multiple entities and dependencies to accomplish one or more tactical attack objectives. It presents a typical multi-stage attack campaign that consists of multiple atomic techniques. To evade detection, this attack can be morphed easily by replacing any technique with an alternative one. Therefore, summarized knowledge of different attack techniques, which is robust and semantically rich, is beneficial to the detection and investigation of cyber attacks [30,33,25].

From Figure 2, Subfigures (B) to (D) show the attack knowledge retrieved from the report sample by TTPDrill [25], ChainSmith [39], and EXTRACTOR [37], respectively, while Subfigure (A) represents the manually generated ground-truth. Subfigure (B) shows attack techniques identified by TTPDrill with manually-defined threat ontology. As shown, TTPDrill can only extract separate techniques from CTI reports without the whole picture. Besides, the ontology provided by TTPDrill contains only action-object pairs for technique identification, which is too vague and may lead to numerous false positives. As the example shows, sending a document is recognized as exfiltration in TTPDrill. However, the “trojanized” document is, in effect, sent by an attacker for exploitation. As shown in Subfigure (C), ChainSmith provides a semantic layer on top of IoCs that captures different roles of IoCs in a malicious campaign. However, they only give a coarse-grained four-stage classification with limited information.

https://blog.talosintelligence.com/2019/06/frankenstein-campaign.html



The threat actors sent the trojanized Microsoft Word documents, probably via email. Talos discovered a document named MinutesofMeeting-2May19.docx. Once the victim opens the document, it fetches a remove template from the actor-controlled website, hxxp://droobox[.]online:80/luncher.doc. Once the luncher.doc was downloaded, it used CVE-2017-11882, to execute code on the victim's machine. After the exploit, the file would write a series of base64-encoded PowerShell commands that acted as a stager and set up persistence by adding it to the HKCU\Software\Microsoft\Windows\CurrentVersion\Run Registry key.

actor
website
(hxxp://droobox.online:80/luncher.doc)
e-mail
microsoft word document (MinutesofMeeting-2May19.docx)
document (luncher.doc)
powershell commands
Exploit (CVE-\d+-\d+)
(HKCU\Software\Microsoft\Windows\CurrentVersion\Run)
(A) Ground-truth
IP '*'
*
MinutesofMeeting-2May19.docx
luncher.doc
(HKCU\Software\Microsoft\Windows\CurrentVersion\Run)
(D) Attack graph by "EXTRACTOR"
T1566: Phishing Email
T1204: User Execution
actor
e-mail
links, urls, ... (hxxp://droobox.online:80/luncher.doc)
attachment, ... (*.docx, *.rtf, *.rar, ...)
T1203: Exploitation
attachment, ... (*.docx, *.rtf,, *.rar, ...)
Exploit (CVE-\d+-\d+)
T1547: Boot Autostart
executables, macros, ...
([HKLM|HKCU]\Software\Microsoft\Windows\CurrentVersion\.*\Run\.*, HKLM\SYSTEM\CurrentControlSet\Control\Session, ...)
(E) Our Technique Knowledge Graph
(×) T1048: Exfiltration Over Alternative Protocol
Sent
Microsoft word document (MinutesofMeeting-2May19.docx)
T1203: Exploitation
Use
Exploit (CVE-2017-11882)
T1547: Boot Autostart
Adding to
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
(B) Part of attack techniques extracted with "TTPDrill"
Stage - Baiting
Stage - Exploitation
hxxp://droobox.online:80/luncher.doc
Exploit (CVE-2017-11882)
Stage - Installation
luncher.doc
Stage - C&C
(C) Attack chain extracted with "ChainSmith"

Fig. 2: A motivating example.

As Subfigure (D) shows, the attack graph generated by EXTRACTOR merges all non-IoC entities of the same type. It thus loses the structural information of attack behaviors, making it impossible to identify attack techniques accurately.

Subfigure (E) illustrates the ideal result we would like to extract in this paper. As long as we can locate attack techniques in attack graphs extracted from CTI reports, we are able to aggregate technique-level knowledge and enrich attack graphs with more comprehensive knowledge about the corresponding techniques. For example, we can find more possible vulnerabilities that can be used in *T1203-Exploitation for Execution* as a replacement for *CVE-2017-11882* appeared in this report. Moreover, the distinct threat intelligence can be collected and aggregated at the technique level across multiple CTI reports.

3 Approach

3.1 Overview of AttacKG

Figure 3 shows the architecture of AttacKG. At a high level, AttacKG has two subsystems: (1) an attack graph extraction pipeline for CTI reports parsing and attack graphs building, and (2) an attack technique identification subsystem for technique template generation and technique identification in attack graphs.

Extract Attack Graphs from CTI Reports. To accurately extract attack graphs from CTI reports, we design a parsing pipeline of five stages. As shown in

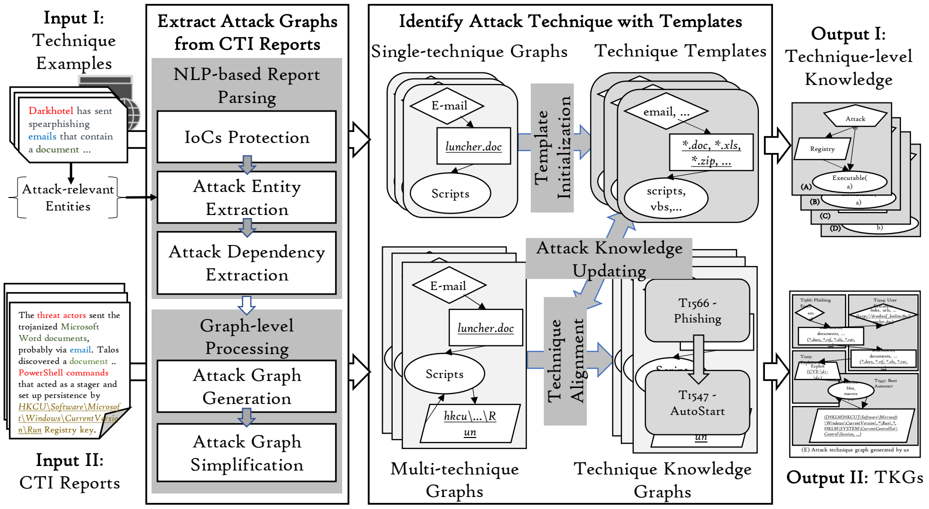

Input I:
Technique Examples
Darkhotel has sent spearphishing emails that contain a document ...
Attack-relevant Entities
Input II:
CTI Reports
The threat actors sent the trojanized Microsoft Word documents, probably via email. Talos discovered a document .. PowerShell commands that acted as a stager and set up persistence by HKCU\Software\Microsoft\Windows\CurrentVersion\Run Registry key.
Extract Attack Graphs from CTI Reports
NLP-based Report Parsing
IoCs Protection
Attack Entity Extraction
Attack Dependency Extraction
Graph-level Processing
Attack Graph Generation
Attack Graph Simplification
Identify Attack Technique with Templates
Single-technique Graphs
Technique Templates
E-mail
luncher.doc
Scripts
Template Initialization
email, ...
*.doc, *.xls, *.zip, ...
scripts, vbs,...
Attack Knowledge Updating
E-mail
luncher.doc
Scripts
hkcu\...\Run
Technique Alignment
T1566 - Phishing
T1547 - AutoStart
Multi-technique Graphs
Technique Knowledge Graphs
Output I:
Technique-level Knowledge
Attack
Registry
Executable(a)
(A)
(B)
(C)
(D)
Output II: TKGs
T1566: Phishing
T1204: User
links, urls, ...
documents, ...
(*.docx, *.rtf, *.xls, *.rar, ...)
T1203:
Exploit
(CVE-\d+-\d+)
bles, macros
T1547: Boot Autostart
([HKLM|HKCU]\Software\Microsoft\Windows\CurrentVersion\.*\Run\.*, HKLM\SYSTEM\CurrentControlSet\Control\Session, ...)
(E) Attack technique graph generated by us

Fig. 3: Overview of AttacKG architecture.

Figure 3, this pipeline has two inputs: (1) technique procedure examples crawled from MITRE ATT&CK describing individual techniques, and (2) CTI reports describing multi-technique attack campaigns. These two inputs are isomorphic, and the corresponding outputs of the pipeline are single-technique graphs for attack techniques and multi-technique graphs for attack campaigns.

Identify Attack Technique with Templates. As discussed in Section 2, individual CTI reports typically have a limited aspect of attack patterns without a global view. In this paper, we aim to bridge this gap by aggregating threat intelligence across CTI reports with technique templates. For this purpose, we propose technique templates to aggregate technique-level intelligence and a revised graph alignment algorithm to identify techniques in attack graphs.

As Figure 3 shows, technique templates are initialized with single-technique attack graphs extracted from technique examples crawled from MITRE. Then, we adopt a revised attack graph alignment algorithm to identify attack techniques in multi-technique graphs extracted from CTI reports with the pre-initialized templates. By aligning multi-technique attack graphs and technique templates, we can enhance the attack graphs with attack knowledge in templates into a technique knowledge graph (TKG) and update the technique templates with rich intelligence from CTI reports.

Finally, we obtain two outputs: (1) technique templates that collect and aggregate attack knowledge across CTI reports at the technique level; (2) TKGs that summarize complete attack chains in CTI reports. It is worth mentioning that AttacKG can tolerate a few false-positives/false-negatives in templates or attack graphs as long as techniques implementations appear multiple times in different reports and most of them are parsed correctly.

3.2 CTI Reports Parser

In this section, we introduce our primary approach for extracting attack graphs from CTI reports. Well-written CTI reports include detailed technical descriptions of how attack-related entities interact to accomplish adversarial objectives in attack campaigns. Despite the rich information, it is challenging to accurately extract attack behaviors from CTI reports written in natural language. Specifically, we identify four key challenges:
[C1] Domain-specific terms identification. CTI reports often contain numerous security-related terms, such as IoCs and attack family names, that include special characters and confuse most off-the-shelf NLP models.
[C2] Attack entity and dependency extraction. Unlike provenance graphs that record attack actions with full details, CTI reports are written in a more summarized manner, providing an overview of the attack workflow. Thus, the attack graphs extracted from CTI reports usually illustrate coarse-grained and incomplete dependencies among entities.
[C3] Co-reference resolution. Co-reference is very common in natural language. We identify two types of co-reference in CTI reports. Explicit co-references use pronouns like "*it*" and "*this*" or definite article "*the*," while implicit co-references use synonyms to refer to entities that appear in the preceding text.
[C4] Attack graph construction and simplification. Attack scenarios described in natural language are redundant and fractured, which NLP technology cannot address. Therefore we need to construct and simplify graphs with the assistance of domain knowledge.

To overcome the above challenges, we design a new CTI report parsing pipeline based on the existing ones [22,6], with better performance in handling co-reference and constructing attack graphs. Notice that most CTI reports are shared in the forms of PDF and HTML, which we further translate into a uniform text format with open-source tools (e.g.,*pdfpulmer* and *html2text*).

IoC Recognition and Protection with Regex [C1]. CTI reports contain numerous domain-specific terms, such as `CVE-2017-21880` and `/etc/passwd`, which include special characters and thus confuse general NLP models. In order to avoid the influence of these terms while preserving the information on attack behaviors, we identify them with a refined version of open-source IoC recognizer [10] by extending the regex set and replacing them with commonly used words according to their entity types. We also record the location of replaced words for subsequent resumption of the IoCs. Afterward, we are able to adopt standard models [17] for first-stage CTI report parsing.

Attack Entity and Dependency Extraction [C2, C3]. In addition to IoC entities, non-IoC entities also play important roles in attack technique expression. For better extraction, we classify entities into six types. Among them, **Actor** and **Executable** represent subjects of attack behaviors, while **File**, **Network Connection**, and **Registry** denote common system-level objects. Additionally, we identify several "other" types of entities that frequently appear in certain techniques but are difficult to directly map to system objects. As a result, we classify them into a separate category named **Others**.

Then, we adopt a learning-based Named Entity Recognition (NER) model to recognize entities in CTI reports. The model is pre-trained on a large general corpus[5] and re-trained with technique examples randomly sampled from MITRE that cover all techniques and entity types. To improve the accuracy of entity identification, we further use a customized rule-based entity recognizer[6] to identify well-defined and common entities. Furthermore, we leverage an open-source co-reference resolver, co-referee[7], for explicit co-reference's resolution. All pronouns for an attack-relevant entity are recorded in a linked table, and the corresponding nodes will be merged when constructing the attack graph.

For intra-sentence dependency extraction, we first construct a dependency tree for each sentence with a learning-based nature language parsing model [17]. Then, we enumerate all pairs of attack-relevant entities (including their pronouns) and estimate the distance between them with the distance of their Lowest Common Ancestor (LCA) and the distance of their position in the sentence. Each entity will establish dependencies with its nearest entity unless only one entity exists in the sentence.

Attack Graph Generation and Simplification [C3, C4]. Given extracted attack entities and dependencies, we can initialize a graph, called *attack graph* (G_a), where nodes represent attack-relevant entities and edges represent their dependencies. So far, we have only considered the dependencies within sentences. Cross-sentence dependencies will be established through both explicit and implicit co-reference nodes. In particular, by merging co-reference nodes, we not only remove redundant nodes but also combine sentence-level sub-graphs into a whole attack graph. Explicit co-reference can be recognized by general NLP models, as discussed in Section 3.2, while implicit co-references need to be identified based on entities' type and character-level overlaps. To do so, we use node-level alignment scores, as discussed in Section 3.3, to determine whether two nodes should be treated as co-references nodes.

Finally, we generate a concise and clear attack graph describing all attack behaviors that appear in a CTI report. Our evaluation of a wide range of CTI reports demonstrates that the attack graph extraction pipeline is both accurate and effective (Section 4.2). And a few false positives/negatives have limited impact on the subsequent fuzzy alignment-based technique identification.

3.3 Technique Templates and Graph Alignment Algorithm

In order to identify techniques from attack graphs while preserving their implementation details mentioned in CTI reports, we first need a universal description of attack techniques. Towards this end, we propose the design of graph-structured technique templates to represent individual techniques. Then, inspired by the graph alignment algorithm in Poirot [33] for attack behavior identification in

[5] https://github.com/explosion/spacy-models/releases/tag/en_core_web_sm-3.1.0
[6] https://spacy.io/api/entityruler
[7] https://github.com/msg-systems/coreferee

provenance graphs, we design a revised graph alignment algorithm for technique identification in attack graphs with templates. Finally, we introduce how to initialize and update technique templates with alignment results.

Design of Technique Templates To describe attack behaviors (represented as graphs) within techniques while aggregating threat intelligence, we model technique templates also as graphs (G_t) with statistics of occurrences of entities (nodes) and dependencies (edges) in different CTI reports. In the graph, nodes represent aggregated entity knowledge, and edges represent possible dependencies among them.

Moreover, we calculate the confidence of entities and dependencies by their occurrences in different reports. In this way, as long as techniques appear multiple times in different reports and most of them are parsed correctly, the impact of possible false positives and/or false negatives introduced by AttacKG's misidentifications and adversarial or low-quality CTI reports can be tolerated.

Graph Alignment for Technique Identification and Technique Knowledge Graph Construction. As discussed above, both attack graphs we extracted in Section 3.2 and technique templates we generated in Section 3.3 may contain false positives and/or false negatives. Therefore, we cannot use the exact match to align them. As an alternative, we propose a graph alignment algorithm between technique template G_t and attack graph G_a based on fuzzy matching. Specifically, as shown in Table 2, we define two kinds of alignments, i.e., node alignment between two nodes in two separate graphs and graph alignment that measures the overall similarity between a technique template and a specific subgraph in an attack graph.

Table 2: Notations in Graph Alignment

Notation	Description
$i : k$	Node alignment between node i and k from separate graphs
$i \dashrightarrow j$	A dependency (path) from node i to node j
$G_t :: G_a$	Graph alignment between Template G_t and Attack Graph G_a
$\Gamma(i : k)$	Alignment score between node i to node k
$\Gamma(G_t :: G_a)$	Alignment score between Template G_t and Attack Graph G_a
$\Gamma_N(G_t :: G_a)$	Node-level alignment score between Template G_t and Graph G_a
$\Gamma_E(G_t :: G_a)$	Dependency-level alignment score between Template G_t and Graph G_a

Node alignment. We first enumerate every node k in an attack graph G_a to find its alignment candidates for every node i in a technique template G_t by calculating the alignment score for nodes $\Gamma(i : k)$. The alignment score between two nodes is computed by Equations (1) and (2):

$$\Gamma(i : k) = \begin{cases} \gamma + (1-\gamma) \cdot Sim(i, k) & i_{type} = k_{type} \\ 0 & i_{type} \neq k_{type} \end{cases}, \tag{1}$$

$$Sim(i, k) = Max(sim(i_{IoC}, k_{IoC}), sim(i_{NLP}, k_{NLP})). \tag{2}$$

Intuitively, if node i and node k have different types, then the alignment score will be zero. Otherwise, they will get a type-matched score γ. Then the similarity of nodes' attributes ($Sim(i,k)$) can be determined by calculating character level [11] similarity ($sim(i,k)$) of IoC terms and natural language descriptions, which is enumerated in templates. If the alignment score reaches a pre-defined threshold, we will record the node alignment candidate in G_a to a list of the corresponding template node.

Graph alignment. Afterward, we iterate through all candidate nodes to calculate the overall alignment scores $\mathit{\Gamma}(G_t :: G_a)$ in two parts: node-level alignment scores $\mathit{\Gamma}_N(G_t :: G_a)$ and edge-level alignment scores $\mathit{\Gamma}_E(G_t :: G_a)$. Specifically, the alignment score between a technique template and an attack graph can be computed by Equations (3), (4), and (5):

$$\mathit{\Gamma}_N(G_t :: G_a) = \sum_{i \in G_t, k \in G_a} (\mathit{\Gamma}(i:k) \cdot i_{occur}) \Big/ \sum_{i \in G_t} (i_{occur}), \tag{3}$$

$$\begin{aligned}\mathit{\Gamma}_E(G_t :: G_a) = &\sum_{\substack{i \dashrightarrow j \in G_t \\ k \dashrightarrow l \in G_a}} \left(\frac{(\mathit{\Gamma}(i:k) \cdot \mathit{\Gamma}(j:l)}{C_{min}(k \dashrightarrow l)} \cdot (i \dashrightarrow j)_{occur} \right) \\ &\Big/ \sum_{i \dashrightarrow j \in G_a} ((i \dashrightarrow j)_{occur}),\end{aligned} \tag{4}$$

$$\mathit{\Gamma}(G_t :: G_a) = \frac{1}{2} \cdot (\mathit{\Gamma}_N(G_t :: G_a) + \mathit{\Gamma}_E(G_t :: G_a)). \tag{5}$$

As shown, the node-level alignment score ($\mathit{\Gamma}_N(G_t :: G_a)$) is a weighted sum of the alignment score ($\mathit{\Gamma}(i:k)$) of each node. The weights are proportional to the number of node occurrences (i_{occur}) recorded in the template. In this way, we enhance the impact of important entities and dependencies that commonly appear in different CTI reports. Meanwhile, the edge-level alignment score ($\mathit{\Gamma}_E(G_t :: G_a)$) depends on three factors: alignment scores of the nodes at both ends of the dependency ($\mathit{\Gamma}(i:k)$ and $\mathit{\Gamma}(j:l)$); the minimal hop between both ends of the dependency ($C_{min}(k \dashrightarrow l)$) in the attack graph; the number of node occurrences ($(i \dashrightarrow j)_{occur}$) recorded in the template. If two nodes are not connected, the dependency between them ($C_{min}(k \dashrightarrow l)$) will be set to infinity. Finally, the outputs of all five equations above are normalized to interval $[0,1]$. We note that although the graph traversal is used, the computational overhead of the whole algorithm is acceptable due to the limited size of both attack graph and technique templates.

After obtaining alignment scores for candidate permutations of each technique, we compare them with a pre-defined threshold and finally select aligned subgraphs of techniques. It is noteworthy that one attack graph node can be aligned to multiple techniques, and one technique can be found multiple time in an attack graph as long as each aligned subgraph have an alignment score above the threshold.

TKG construction. With the graph alignment results, we can attach attack knowledge described in technique templates, including alternative entities and techniques, to the corresponding positions in an attack graph. Then, we can obtain the technique knowledge graph (TKG) that introduces the whole attack chain in a CTI report with enhanced knowledge.

Initialization and Updating of Technique Templates Both the initialization and updating of technique templates rely on the graph alignment results.

Updating. By aligning, the node in an attack graph can be mapped to the node in the identified technique template. Then we update the IoC and natural language description sets in the template node with new terms from the aligned attack graph node. This allows us to aggregate threat intelligence across CTI reports at the technique level.

Initialization. The initialization of technique templates starts with a random single-technique attack graph extracted from MITRE technique examples. Then we align the initial template with other single-technique attack graphs of the same technique. The information in aligned attack graph nodes will be merged into the corresponding template node. And the unaligned nodes will be added to templates as new nodes, which is different from the updating process.

Finally, we generate the technique template that aggregates threat intelligence and covers multiple technique variants across multiple reports, as the example shown in Figure 1 and Table 3.

Table 3: Attack entities in the template of the technique `T1547-Boot or Logon Autostart Execution`.

Template entities	**NLP descriptions**	**IoC terms**
Executable	scripts, macros, ...	*.exe, *.ps1, ...
Register	register keys, register, ...	HKLM\...\windows\currentversion\winlogon\*, HKLM\...\active setup\installed components*, HKLM\Software\*\Run, ...
Autostart Folder	startup folder, path, ...	%HOMEPATH%\Start Menu\Programs\Startup\, ~/.config/autostart/*, ...
Shortcut File	shortcut, ...	*.lnk, ...
Side-loading DLL	winlogon helper DLL, SSP DLL, ...	sspisrv.dll, ...

4 Evaluation

In this section, we focus on evaluating AttacKG's accuracy of attack graph extraction and technique identification as a CTI report parser and its effectiveness in technique-level intelligence aggregation as a CTI knowledge collector. In particular, our evaluation aims at answering the following research questions (RQs): (**RQ1**) How accurate is AttacKG in extracting attack graphs (attack-related entities and dependencies) from CTI reports? (**RQ2**) How accurate is AttacKG in identifying attack techniques in CTI reports? (**RQ3**) How effective is AttacKG in aggregating technique-level intelligence from massive CTI reports? Finally, we also would like to evaluate how TKGs benefit downstream security tasks.

4.1 Evaluation Setup

To evaluate AttacKG, we crawled 1,515 real-world CTI reports mentioned in MITRE ATT&CK references whose sources range from Cisco Talos Intelligence Group [3], Microsoft Security Intelligence Center [16], etc. Moreover, we crawled 7,373 technique procedure examples out of 179 techniques from the MITRE ATT&CK knowledge-base [14] to formulate our technique templates.

To answer RQ1 and RQ2, we manually label the ground-truth of entities, dependencies, and techniques in 16 of the collected reports: (1) *DARPA TC Reports*: We select five attack reports released by the DARPA TC program's fifth engagement that cover different OS platforms (i.e., Linux, Windows, and FreeBSD), vulnerabilities (e.g., Firefox backdoor), and exploits (e.g., Firefox BITS Micro). (2) *Real-world APT Campaign Reports*: To explore the performance of AttacKG in practice, we select another eleven public CTI reports that describe APT campaigns from three well-known threat groups, i.e., Frankenstein [7], OceanLotus (APT32) [20], and Cobalt Group [15].

4.2 Evaluation Results

RQ1: How accurate is AttacKG in extracting attack graphs from CTI reports? A typical attack technique consists of multiple threat actions presented as a set of connected entities in an attack graph. In particular, the accurate extraction of attack graphs is an essential starting point toward automated identification of attack techniques from CTI reports. To evaluate the accuracy of AttacKG in extracting attack graphs, we adopt the aforementioned 16 well-labeled CTI reports. We manually identify attack-related entities in the reports and correlate entities based on our domain knowledge of the attack workflow. It is noteworthy that in addition to natural language descriptions, DARPA TC reports also provide the graph representation of attacks, which serves as additional documentation to complement our manual labels.

Given ground-truth entities and dependencies in the reports, we are able to compare AttacKG with the state-of-the-art open-source[8] CTI report parser, EXTRACTOR [37], in terms of the precision, recall, and F1-score. For a fair comparison, we enable all optimizations in EXTRACTOR (e.g., Ellipsis Subject Resolution) when constructing attack graphs upon textual attack descriptions. As discussed in Section 3, an entity may correspond to multiple co-references across a CTI report. Since our goal is to identify unique entities (e.g., IoCs), we merge co-reference entities in the attack graph and integrate their dependencies with the remaining entities.

Table 4 summarizes the results of AttacKG and EXTRACTOR in capturing entities and dependencies from the selected 16 CTI reports(Rows 2-7). As can be seen, despite slightly lower precision caused by a higher false-positive rate, AttacKG yields better accuracy overall (with an average F1-score improvement of 0.12) than EXTRACTOR due to a much lower false-negative rate. This is expected as EXTRACTOR aggregates all non-IoC entities of the same type

[8] https://github.com/ksatvat/EXTRACTOR

Table 4: Accuracy of attack graph extraction and technique identification in 16 CTI reports. (Columns 2-9 present the count of manually-generated ground-truth and $-false\_negative(+false\_positive)$ in extracting attack-related entities, dependencies, and techniques. Columns 10-12 present the overall Precision, Recall, and F1-score.)

CTI reports	Entities			Dependencies			Techniques		
	Manual	Extractor	AttacKG	Manual	Extractor	AttacKG	Manual	TTPDrill	AttacKG
TC_Firefox DNS Drakon APT	10	-4 (+4)	-0 (+1)	9	-4 (+3)	-2 (+1)	8	-2 (+10)	-0 (+3)
TC_Firefox Drakon Copykatz	6	-2 (+0)	-1 (+0)	5	-2 (+0)	-2 (+0)	4	-1 (+13)	-1 (+0)
TC_Firefox BITS Micro APT	11	-6 (+0)	-1 (+4)	10	-7 (+0)	-0 (+0)	5	-1 (+14)	-2 (+2)
TC_SSH BinFmt-Elevate	6	-4 (+0)	-1 (+0)	5	-4 (+0)	-0 (+0)	5	-2 (+14)	-2 (+2)
TC_Nginx Drakon APT	15	-2 (+0)	-2 (+0)	15	-0 (+0)	-2 (+0)	6	-2 (+22)	-0 (+2)
Frankenstein Campaign	14	-3 (+1)	-0 (+2)	16	-5 (+1)	-0 (+2)	9	-1 (+18)	-1 (+1)
OceanLotus(APT32) Campaign	7	-0 (+2)	-0 (+2)	7	-0 (+1)	-1 (+0)	5	-1 (+12)	-2 (+0)
Cobalt Campaign	17	-6 (+0)	-1 (+5)	17	-4 (+0)	-1 (+2)	8	-2 (+21)	-1 (+1)
DeputyDog Campaign	13	-1 (+2)	-0 (+2)	14	-1 (+1)	-2 (+0)	10	-1 (+35)	-0 (+6)
HawkEye Campaign	16	-2 (+3)	-3 (+4)	17	-5 (+3)	-3 (+2)	11	-2 (+64)	-1 (+3)
DustySky Campaign	12	-2 (+1)	-0 (+3)	12	-2 (+1)	-0 (+3)	5	-0 (+32)	-0 (+1)
TrickLoad Spyware Campaign	17	-3 (+1)	-0 (+0)	16	-4 (+0)	-0 (+1)	4	-0 (+18)	-2 (+0)
Emotet Campaign	8	-4 (+0)	-1 (+1)	7	-4 (+0)	-2 (+1)	7	-2 (+16)	-3 (+0)
Uroburos Campaign	12	-1 (+2)	-2 (+3)	13	-3 (+0)	-2 (+0)	7	-0 (+23)	-1 (+2)
APT41 Campaign	13	-1 (+5)	-1 (+0)	12	-0 (+1)	-1 (+2)	6	-2 (+26)	-1 (+1)
Espionage Campaign	11	-2 (+6)	-3 (+1)	10	-3 (+2)	-3 (+1)	4	-0 (+19)	-0 (+1)
Overall Presicion	**1.000**	**0.843**	**0.860**	**1.000**	**0.913**	**0.906**	**1.000**	**0.196**	**0.771**
Overall Recall	**1.000**	**0.771**	**0.915**	**1.000**	**0.741**	**0.886**	**1.000**	**0.837**	**0.808**
Overall F-1 Score	**1.000**	**0.806**	**0.887**	**1.000**	**0.818**	**0.896**	**1.000**	**0.318**	**0.789**

(e.g., process) into one entity, as shown in Figure 2. In other words, no matter how many false-positive entities EXTRACTOR produces, they are treated as one false extraction as long as they belong to the same type. It is noteworthy that such aggregation design inevitably losses structural information of attack graphs and makes follow-up technique identification almost impossible. Hence, we only compare AttacKG with EXTRACTOR in extracting attack graphs rather than identifying attack techniques.

RQ2: How accurate is AttacKG in identifying attack techniques in CTI reports? To answer RQ2, we use AttacKG to identify attack techniques in the 16 CTI reports and compare it with the state-of-the-art technique identifier, TTPDrill [25]. The core idea of TTPDrill is to extract threat actions from CTI reports and attribute such actions to techniques based on threat-action ontology. Specifically, it manually defines 392 threat actions for 187 attack techniques in the original paper, while such ontology knowledge base has been extended to cover 3,092 threat actions for 246 attack techniques in its latest open-source implementation[9]. Also noteworthy is that all attack techniques used by TTPDrill are derived from an old version of the MITRE ATT&CK matrix. To allow for a consistent comparison, we map every technique in TTPDrill to the latest version technique via the hyperlinks provided by MITRE. For example, *T1086-PowerShell* in TTPDrill is updated to *T1059/001-Command and Scripting Interpreter: PowerShell*.

We evaluate AttacKG and TTPDrill on the 16 CTI reports annotated with the ground-truth techniques adopted in the attacks. The technique identification results are summarized in the last three rows in Table 4. We can observe that while both AttacKG and TTPDrill achieve reasonably low false-negative rates,

[9] https://github.com/mpurba1/TTPDrill-0.3

TTPDrill is prone to high volumes of false-positive techniques (15.5 false positives per a report on average), which is nearly three times as many as the true positives. As a result, while the recall of AttacKG is only slightly higher than TTPDrill by 0.1, AttacKG significantly outperforms TTPDrill in terms of the precision and F1-score by 0.575 and 0.462, respectively. This result makes sense as TTPDrill treats threat actions extracted from CTI reports as action-object pairs. Accordingly, techniques that share partial threat actions tend to look similar in TTPDrill. In contrast, AttacKG aligns techniques to attack graphs, considering the full contexts of threat actions.

Moreover, it is worth mentioning that we use fuzzy matching based on alignment scores for technique identification; thus, our approach can correctly identify attack techniques even with FPs/FNs in technique templates and/or extracted attack graphs. Our observation is that the overall accuracy is highest when the graph alignment score's threshold is 0.85, The details of the threshold selection can be found in Appendix 4.4. To verify the importance of each component in AttacKG towards technique identification, we perform an ablation study by considering four variants of AttacKG, as discussed in Appendix 4.4.

RQ3: How effective is AttacKG at aggregating technique-level intelligence from massive reports? To answer RQ3, we explore the effectiveness of AttacKG in extracting threat intelligence (e.g., techniques and IoCs entities) on 1,515 CTI reports collected from different intelligence sources. Table 5 lists the ten most common techniques that appeared in the 1,515 reports and the number of their corresponding unique IoCs, which mostly overlap with manually generated top TTP lists by PICUS [18] and redcanary [19].

Table 5: Effectiveness of Threat Intelligence Extraction from 1,515 CTI Reports.

Top 6 Techniques	Occurrences in reports	#Unique IoCs				
		Executable	Network	File	Registry	Vulner.
T1071 - Command & Control	1113	12	452	371	-	12
T1059 - Scripting Interpreter	1089	6	394	284	100	9
T1083 - File/Directory Discovery	1060	-	-	249	-	-
T1170 - Indicator Removal	990	6	-	255	74	7
T1105 - Ingress Tool Transfer	990	-	389	261	-	-
T1003 - OS Credential Dumping	961	-	-	220	-	-
All Techniques Summary	**28262**	**495**	**2813**	**4634**	**384**	**67**

Each report, on average, contains 18.7 techniques and 5.5 unique IoCs, and different techniques likely involve different IoCs. Note that most CTI reports do not provide unified and formatted intelligence to validate our extracted results, which is also our motivation behind this work. Therefore, we randomly select several technique templates with intelligence aggregated from reports for manual investigation. Specifically, we observe that templates successfully collect unique IoCs for different technique implementations across CTI reports. As the example in Figure 1 and Table 3 shows, we identify multiple unique IoC terms playing similar roles in different reports. Such template-aggregated intelligence can directly enrich our understanding of attack techniques. Furthermore, the TKG built on templates can help understand the entire attack and possible variants for more robust detection and investigation, as shown in Section 2.3.

4.3 Case Study

This subsection discusses how TKGs can be adopted in real-world security tasks with case studies. Specifically, TKGs adopt the collated knowledge to enrich reports, thus helping to understand and reconstruct the attacks involved. In addition, TKGs with aggregated technique-level intelligence can enhance the detection of attack variants.

TKG for attack reconstruction. In order for security practitioners and researchers to have an in-depth analysis, they have to bridge the knowledge gap between the real attacks and CTI reports. This gap can be addressed by having a first-hand practical environment that thoroughly describes how attack steps are performed in the CTI reports. AttacKG provides structured knowledge about an attack scenario, making it easier to reproduce cyber attacks in a testbed environment, benefiting analysts [38] with high fidelity and live reconstructed environment with in-depth details. We have demonstrated how AttacKG supports attack reconstruction in [31].

Taking the Frankenstein campaign as an example, with the TKG extracted from the corresponding report, we can quickly identify nine techniques for six tactics involved in the campaign, including *T1566-Phishing* for tactic Initial Access, *T1547-Boot Autostart* for tactic Persistence, *T1203-Exploitation for Execution* for tactic Execution, etc. Then, we can infer the environment needed to reconstruct the attack based on the techniques and entities involved in the attack. Specifically, *autostart* with registry implies that the attack is running in Windows. The use of vulnerability (*CVE-2017-11882*) for execution indicates the requirement of specific versions of Microsoft Office. After setup the environment, we can reproduce the campaign with open-source attack technique implementation, such as Atomi-Red-Teams [2]. All in all, AttacKG provides necessary information as the first step in the reconstruction process.

TKGs for attack variants detection. As discussed in Section 2.1, frequent and widely used attack variants are posing challenges for detection. Take a simple *T1204-User Execution* and *T1547-Boot Autostart* two-stage attack excerpted from Frankenstein Campaign, for example. Subfigures (A) and (B) in Figure 4 demonstrate the attack and its variants with three nodes mutated. Specifically, the file server URL in *T1204* was changed, and the implementation of *T1547* was switched from (A) *Registry Run Keys* to (D) *DLL Side-loading.* It is noteworthy that such changes will not affect the functionality of the attack.

Then, three representative intelligence-based detection schemes based on different granularity are selected for comparison, namely, 1) iACE [32] that automatically extracts node-level intelligence (subfigure (C)), 2) Poirot [33] that adopt manually-extracted attack graphs for threat hunting (subfigure (D)), and

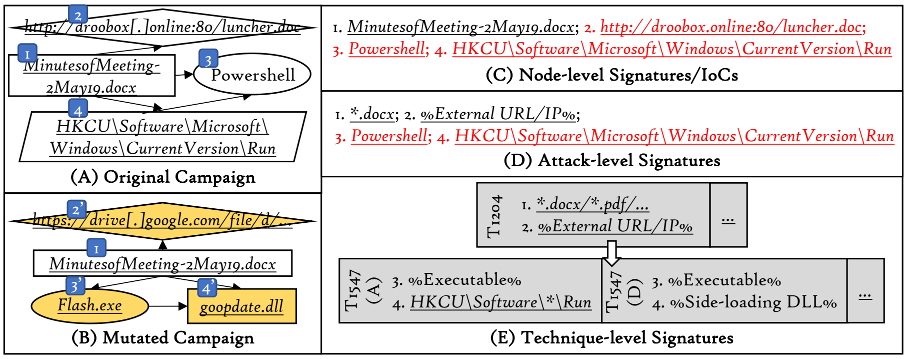

2
http://droobox[.]online:80/luncher.doc
1
MinutesofMeeting-2May19.docx
3
Powershell
4
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
(A) Original Campaign
2'
https://drive[.]google.com/file/d/...
1
MinutesofMeeting-2May19.docx
3'
Flash.exe
4'
goopdate.dll
(B) Mutated Campaign
1. MinutesofMeeting-2May19.docx; 2. http://droobox.online:80/luncher.doc;
3. Powershell; 4. HKCU\Software\Microsoft\Windows\CurrentVersion\Run
(C) Node-level Signatures/IoCs
1. *.docx; 2. %External URL/IP%;
3. Powershell; 4. HKCU\Software\Microsoft\Windows\CurrentVersion\Run
(D) Attack-level Signatures
T1204
1. *.docx/*.pdf/...
2. %External URL/IP%
...
T1547 (A)
3. %Executable%
4. HKCU\Software\*\Run
T1547 (D)
3. %Executable%
4. %Side-loading DLL%
...
(E) Technique-level Signatures

Fig. 4: TKG for attack variants detection. (Red outlined mismatch items.)

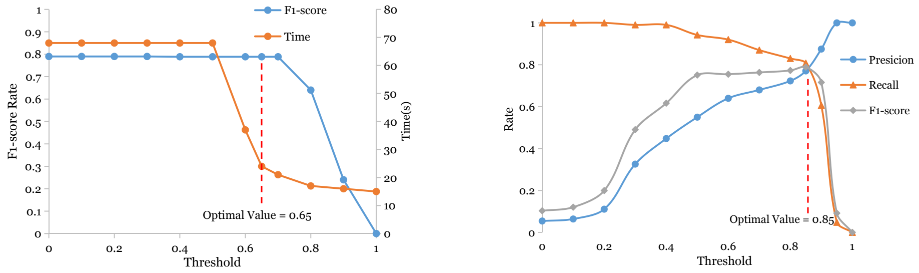
3) our approach that aggregates technique-level intelligence from multiple reports (subfigure (E)). As the matching results show, to avoid introducing numerous false positives, node intelligence-based detection requires exact matching, which can be easily bypassed by obfuscation. By considering structure information, attack-level matching allows the generalization of node information to improve detection generality. However, attackers can still easily evade detection by changing techniques used in the campaign.

Nevertheless, the technique-level intelligence we provide enables detectors to detect different attack techniques independently. Moreover, the pooled technique knowledge from multiple reports can effectively improve the detection of various variants. And the aggregated intelligence can be automatically merged with approaches like Eiger [27] for better generality.

4.4 Minor Evaluation Points

Selecting the Threshold Value The selection of the threshold value for node/graph alignment scores affects the accuracy and efficiency of AttacKG. Specifically, too low a threshold for graph alignment score could result in premature matching (false positives), while too high could lead to missing reasonable matches (false negatives). For node alignment score, too low a threshold could leave unnecessary alignment candidates and cost longer report analysis time, while too high could lead to false negatives. Thus, there are trade-offs in choosing optimal threshold values. To determine optimal threshold values, we measure the F-score and report analysis time using varying threshold values, as shown in Figure 5, and select optimal threshold values (0.65 for node alignment, 0.85 for graph alignment) that make each index better at the same time.

Ablation Study of AttacKG. In particular, we first remove part of the attributes in entities: the IoC information and natural language text termed $AttacKG_{w\backslash o\ IoC\ information}$ and $AttacKG_{w\backslash o\ natural\ language\ text}$, respectively. Note that unlike the EXTRACTOR's practice of merging entities, which may result in information loss, we only remove partial entity attributes without sacrificing the structural information of attack graphs. Moreover, we obtain another variant by filtering out dependencies in attack graphs termed $AttacKG_{w\backslash o\ dependencies}$.



(a) node alignment score

(b) graph alignment score

Fig. 5: Threshold selection

That is, we predict attack techniques only based on entity sets. Finally, we disable the graph simplification component termed $AttacKG_{w\backslash o\ graph\ simplification}$.

Table 6: Ablation study of different components used in technique identification.

Components	Precision	Recall	F1-Score
w/ all component	0.782	**0.860**	**0.819**
w/o IoC information	**0.833**	0.600	0.698
w/o natural language text	0.690	0.800	0.741
w/o dependencies	0.667	0.480	0.558
w/o graph simplification	0.696	0.780	0.736

As different component combinations may affect the distribution of alignment scores, we adjust and choose identification thresholds separately for AttacKG variants in light of the optimal F1-scores. The results are summarized in Table 6. We find that removing any component would degrade AttacKG 's performance, which well justifies our design choice. Especially, $AttacKG_{w\backslash o\ dependencies}$ consistently performs the worst across all evaluation metrics. It verifies the substantial influence of graph structures in technique identification.

Efficiency of AttacKG Settings. We experimentally compared AttacKG's efficiency with TTPDrill and Extractor on the 16 CTI report samples mentioned in Section 4.1 on a PC with AMD Ryzen 7-4800H Processor 2.9 GHz, 8 Cores, and 16 Gigabytes of memory, running Windows 11 64-bit Professional. The size of the reports used as samples ranges from 61 words to 1029 words, with an average of 278.2 words.

Results. Extractor is the most complex system that consists of multiple NLP models and thus has the highest runtime overhead, taking 239.70 seconds on average to parse a report. Compared to Extractor, AttacKG adopts a simpler CTI report parsing pipeline. On average, it takes 8.9 seconds and 15.1 seconds for graph extraction and technique identification, respectively, totaling 24.0 seconds. TTPDrill, on the other hand, uses the simplest model without constructing attack graphs and thus is the fastest, taking only 5.9 seconds on average per report, but at the cost of a high false-positive rate.

5 Conclusion

In this paper, we propose an automated solution for retrieving structured threat intelligence from CTI reports. We use the notion of technique templates to summarize attack-technique-level threat intelligence across CTI reports. Then, we leverage attack knowledge in the templates to enhance attack graphs extracted from CTI reports and generate the technical knowledge graph (TKG). We implement our prototype system, AttacKG, and evaluate it with 1,515 real-world CTI reports. Our evaluation results show that AttacKG can extract attack graphs from CTI reports accurately and aggregate technique-level threat intelligence from massive reports effectively.

Acknowledgement. This material is based upon work supported in part by National Science Foundation with the Award Number (FAIN) 2148177 and the National Research Foundation, Prime Ministers Office, Singapore under its National Cybersecurity R&D Program (Award No. NRF-NCL-P2-0001) and administered by the National Cybersecurity R&D Directorate. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

1. AlienVault OTX, https://otx.alienvault.com
2. Atomic Red Team, https://github.com/redcanaryco/atomic-red-team
3. Cisco Talos Intelligence Group - Comprehensive Threat Intelligence, https://blog.talosintelligence.com/
4. CybOX - Cyber Observable Expression, https://cyboxproject.github.io/
5. DARPA Transparent Computing, https://www.darpa.mil/program/transparent-computing
6. Extractor, https://github.com/ksatvat/EXTRACTOR
7. Frankenstein Campaign, https://blog.talosintelligence.com/2019/06/frankenstein-campaign.html
8. IBM X-Force, https://exchange.xforce.ibmcloud.com/
9. Introduction to STIX, https://oasis-open.github.io/cti-documentation/stix/intro.html
10. ioc parser, https://github.com/armbues/ioc_parser
11. Levenshtein distance, https://en.wikipedia.org/wiki/Levenshtein_distance
12. mandiant/OpenIOC_1.1, https://github.com/mandiant/OpenIOC{_}1.1
13. Microsoft says SolarWinds hackers stole source code for 3 products, https://arstechnica.com/information-technology/2021/02/microsoft-says-solarwinds-hackers-stole-source-code-for-3-products
14. MITRE ATT&CK®, https://attack.mitre.org/
15. Multiple Cobalt Personality Disorder, https://blog.talosintelligence.com/2018/07/multiple-cobalt-personality-disorder.html
16. Security intelligence — Microsoft Security Blog, https://www.microsoft.com/security/blog/security-intelligence/
17. spaCy, https://spacy.io/
18. The Top Ten MITRE ATT&CK Techniques, https://www.picussecurity.com/resource/the-top-ten-mitre-attck-techniques
19. Top MITRE ATT&CK Techniques, https://redcanary.com/threat-detection-report/techniques/

20. OceanLotus: Extending Cyber Espionage Operations Through Fake Websites (2021), https://www.volexity.com/blog/2020/11/06/oceanlotus-extending-cyber-espionage-operations-through-fake-websites/
21. Gao, P., Liu, X., Choi, E., et al.: A system for automated open-source threat intelligence gathering and management. In: SIGMOD (2021)
22. Gao, P., Shao, F., Liu, X., et al.: Enabling Efficient Cyber Threat Hunting With Cyber Threat Intelligence. ICDE (2021)
23. Ghazi, Y., Anwar, Z., Mumtaz, R., Saleem, S., Tahir, A.: A supervised machine learning based approach for automatically extracting high-level threat intelligence from unstructured sources. In: 2018 International Conference on Frontiers of Information Technology (FIT). pp. 129–134. IEEE (2018)
24. Hossain, M.N., Sheikhi, S., Sekar, R.: Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In: IEEE S&P (2020)
25. Husari, G., Al-Shaer, E., Ahmed, M., Chu, B., Niu, X.: TTPDrill: Automatic and accurate extraction of threat actions from unstructured text of CTI Sources. In: ACM International Conference Proceeding Series. vol. Part F1325 (2017)
26. Husari, G., Niu, X., Chu, B., Al-Shaer, E.: Using entropy and mutual information to extract threat actions from cyber threat intelligence. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 1–6. IEEE (2018)
27. Kurogome, Y., Otsuki, Y., et al.: Eiger: Automated IOC generation for accurate and interpretable endpoint malware detection. In: ACM ACSAC (2019)
28. Legoy, V., Caselli, M., Seifert, C., Peter, A.: Automated retrieval of att&ck tactics and techniques for cyber threat reports. arXiv preprint arXiv:2004.14322 (2020)
29. Li, G., Dunn, M., Pearce, P., et al.: Reading the Tea Leaves: A Comparative Analysis of Threat Intelligence. In: Usenix Security Symposium (2019)
30. Li, Z., Chen, Q.A., Yang, R., Chen, Y.: Threat Detection and Investigation with System-level Provenance Graphs: A Survey. Computer & Security **106** (2021)
31. Li, Z., Soltani, A., Yusof, A., et al.: Poster: Towards automated and large-scale cyber attack reconstruction with apt reports. In: NDSS'22 Poster Session
32. Liao, X., Yuan, K., Wang, X., et al.: Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. In: CCS (2016)
33. Milajerdi, S.M., Gjomemo, R., Eshete, B., et al.: Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In: CCS (nov 2019)
34. Momeni Milajerdi, S., Gjomemo, R., et al.: HOLMES: Real-time APT detection through correlation of suspicious information flows. In: IEEE S&P (2019)
35. Mu, D., Cuevas, A., Yang, L., et al.: Understanding the reproducibility of crowd-reported security vulnerabilities. In: Usenix Security Symposium (2018)
36. Ramnani, R.R., Shivaram, K., Sengupta, S.: Semi-automated information extraction from unstructured threat advisories. In: Proceedings of the 10th Innovations in Software Engineering Conference. pp. 181–187 (2017)
37. Satvat, K., Gjomemo, R., Venkatakrishnan, V.: Extractor: Extracting attack behavior from threat reports. In: IEEE EuroS&P (2021)
38. Uetz, R., Hemminghaus, C., Hackländer, L., et al.: Reproducible and adaptable log data generation for sound cybersecurity experiments. In: Annual Computer Security Applications Conference. pp. 690–705 (2021)
39. Zhu, Z., Dumitras, T.: ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports. In: IEEE European Symposium on Security and Privacy (2018)