CYTAG: Multi-Source Behavioral Aggregation of Natural Language Cyber Threat Intelligence

by

Olajide D. David

**A thesis in partial fulfillment
of the requirements for the degree of
Master of Science
(Cybersecurity and Information Assurance)
in the University of Michigan-Dearborn
2022**

Master's Thesis Committee:

**Assistant Professor Birhanu Eshete, Chair
Professor Hafiz Malik
Professor Bruce Maxim**

© Olajide D. David
2022

Dedication

I dedicate my thesis work to God almighty for His favor, grace, the wisdom, and strength granted to me to complete this work. A heart felt gratitude to my loving parents Mr. and Mrs. David, for making this possible with their unending support and encouragement throughout the course of my master's degree program. To my sisters who have always been by my side and are special. Also, to Tommy for motivating me.

I would also like to recognize Professor Di Ma for her words of encouragement and support, my program adviser Associate Professor Jinhua Guo and all the other teaching staff of the department of Cybersecurity and Information Assurance for the unrelenting efforts towards ensuring I had access to the best possible learning conditions and training.

I also dedicate this thesis to my friends Monica Oriaghe, Arnaud Shyaka, Kunle, and Rand who have supported me through this journey, this journey would not have been possible without them. To Mr. Michael Yockey, Mr. David Bankole, Dr Khan just to mention a few, you all have been amazing, supportive, and amiable. I want to say a very big thank you.

Acknowledgements

Special thanks go to my supervisor Professor Birhanu Eshete for his guidance, support, and unrelenting efforts towards making this thesis a success throughout the semester.

I would like to thank my wife Temi, who helped me during the tough moments of this thesis both technically and emotionally.

Table of Contents

List of Tables

List of Figures

Abstract

The current state-of-the-art in extracting machine-readable attack behavior graphs from natural language cyber threat intelligence (CTI) reports relies on one prominent CTI source such as a high-profile advanced persistent threat (APT) report. This thesis hypothesizes that multiple CTI sources offer complementary fragments of attack behavior due to factors such as variation in analysis details of an attack, polymorphic nature of malicious behavior manifestations, and experience and resources available to the analyst(s) who produce a CTI report. To test this hypothesis, this work proposes a systematic attack behavior graph aggregation approach, called CYTAG, that significantly enhances the fidelity of an attack graph given multiple CTI sources about a given attack such as an APT. CYTAG achieves this while preserving attack semantics and minimizing redundancy of nodes and edges in the aggregated attack graph. Evaluation of CYTAG on CTI reports covering multiple years and comparing its attack graph aggregation results with state-of-the-art attack behavior extraction approach suggests that CYTAG significantly improves the detection and forensics arsenal of cyber threat hunters with reasonable aggregation performance overhead.

Chapter 1: Introduction

In the world of cyber threat hunting and security operations, Cyber Threat Intelligence (CTI) is publicly released as technical reports by practitioners, white papers by security vendors, and technical security blogs by researchers. CTI reports play a crucial role as invaluable sources of actionable cyber threat information pertinent to attack vectors such as malware, botnets, ransomware, and sophisticated cyber-attacks such as Advanced Persistent Threats (APTs). In CTI reports, attack motives, goals, and technical progressions are narrated with a wealth of Indicators of compromise (IOCs) and causal links among attack activities.

Prior work: To make the best use of CTI reports for effective cyber threat hunting, attack detection, and forensic analysis, the cybersecurity community has developed methods focusing on various priorities. Among the notable directions taken are IOC representation and exchange [1]–[3], IOC extraction [[4], [5]], attack campaign characterization [6], and automated attack behavior graph extraction [7]–[9]. All these approaches aim to empower attack detection by reinforcing cyber defense countermeasures such as Anti-virus software and Intrusion Detection Systems (IDSs).

Motivation: The focus of this work is on automated extraction of attack behavior graphs from natural language CTI reports. The current state-of-the-art [8], [9] in automated extraction of machine-readable attack behavior graphs from CTI reports relies on analyzing one prominent CTI source such as a high-profile advanced persistent threat (APT) report —hence suffers from the single CTI source problem. In this work, we observe that multiple CTI sources often capture complementary fragments of threat behavior, and this observation is attributed to multiple factors.

Among the main factors are variations in breadth and depth of analysis of an attack, variations in how malicious behaviors manifest depending on target environments and triggers, and experience and resources available to the analyst(s) who examine attacks and produce CTI reports. This work hypothesizes that, if carefully stitched together, the complementary attack semantics fragments from multiple CTI sources can potentially result in an attack behavior graph that is even richer in attack semantics and consequently improves attack detection and forensics capabilities in enterprise settings.

Approach Overview: To test the aforementioned hypothesis, this work proposes a systematic attack behavior graph aggregation approach, called CYTAG, that significantly enhances the fidelity of an attack graph given multiple CTI sources about a given attack such as an APT. CYTAG achieves this while preserving attack semantics and minimizing redundancy of nodes and edges in the aggregated attack graph. Given multiple CTI reports about the same attack (e.g., an APT), CYTAG operates in two phases: (*i*) attack behavior graph extraction and (*ii*) attack behavior graphs aggregation. In phase (*i*), CYTAG builds on the technique in Extractor [9]and improves it to extract attack behavior graphs from complex CTI reports. In phase (*ii*), attack behavior graphs extracted from multiple sources are systematically aggregated to enrich a single-source attack behavior graph by adding new nodes and edges. This is done while preserving the overall threat behavior narrative and eliminating repetitions that result from overlaps among CTI sources. The key intuition here is that through complementary attack behavior captured across multiple sources, CYTAG offers improved detection and forensics capabilities when the aggregated attack behavior graph is used.

Evaluation Overview: Evaluation of CYTAG on diverse CTI reports covering multiple years and comparing its attack graph aggregation results with state-of-the-art attack behavior extraction approach suggests that CYTAG significantly improves the detection and forensics arsenal of cyber threat hunters with reasonable aggregation performance overhead. We evaluated CYTAG by analyzing the node, edge, nodeedge relationship of the attack graph generated from the three other sources that were collected for each APT report. The APTs analyzed range from currently active to obsolete exploited threats. We applied CYTAG on 1K+ dataset of CTI reports which resulted in an improved attack graph extraction rate of an average of 21% more than Extractor [9](current state-of-the-art approach). Our results show that CYTAG is able to generate IoC-rich aggregated attack graphs from multiple CTI sources with high accuracy, demonstrating that our system produces promising directions towards improved arsenal for attack behavior graphs for cyber threat hunting.

Contributions: In the space of CTI-guided cyber threat hunting, this work makes the following contributions:

- Enhanced single-source attack graph extraction that improves current state-of-the-art (Extractor [9]) with enhancements to data collection and improvement of underlying threat dictionary used to power NLP-based graph extraction.
- A new multi-source attack graph aggregation approach that results in semantically-rich attack behavior graph.
- An evaluation that spans multiple years (2013 – 2019) of CTI reports and a demonstration of CYTAG's superior attack graph extraction capabilities.

Thesis Organization: This rest of this thesis report is organized as follows. In Chapter 2, we cover the background and related work. In Chapter 3, we present the approach details of CYTAG with the design on how CYTAG works in more details. In Chapter 4, we evaluate the performance of

CYTAG and analyze the effect of the graph aggregation techniques on CTI report datasets. Finally, we conclude with insights gained from the study and possible gaps in Chapter 5.

Chapter 2: Background and Related Work

This chapter first defines terminology and notations (Section 2.1), and then presents related work relevant to CYTAG (Section 2.2).

2.1 Background

To ease forthcoming discussions, we define important terminology and notations to be used throughout this thesis.

Indicator of Compromise (IOC). While different variations of the term exist [10], [11] we will consider the following as the working definition: *an artifact observed on a network or in an operating system that, with high confidence, indicates a computer intrusion/attack*. Typical IOCs include malware signatures, IP addresses/hostnames/URLs associated with malware, botnets, and command-and-control (C&C) servers, and hash values of malware binaries. Cyber threat hunters or security operation centers leverage preliminary IOCs to build more concrete indicators such as unusual inbound and outbound network traffic, unknown programs running within a host, unusual activities from privileged accounts (e.g., administrators), and abnormally large number of requests for the same file. Once IOCs are identified, they are typically used for early detection of attack attempts as part of anti-virus (AV) software and intrusion detection system (IDS).

Advanced Persistent Threat (APT). It is a stealthy threat actor, typically a nation state or state-sponsored group, which gains unauthorized access to a computer network and remains undetected for an extended period. Alternatively, the term may also refer to non-state-sponsored groups conducting large-scale targeted intrusions for specific goals [12]. A typical APT is characterized by the so called “APT kill-chain” which consists of an initial penetration/compromise (e.g., via a drive-by-download or a spear-phishing attack) reconnaissance, C&C communication, privilege escalation, lateral movement through a network, ex-filtration of confidential information, and clean-up of attack footprint [13].

Cyber Threat Intelligence (CTI). It is defined as “evidence-based knowledge, including context, mechanisms, indicators, implications, and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject’s response to that menace or hazard [14]. CTI is instrumental in enabling an enterprise build insight into the ever-changing cyber threat landscape to proactively identify and mitigate cyber [15]. With IOCs as integral parts, CTI often comes in several forms and a natural language CTI is the subject of this thesis. From now on, when we say, “CTI reports”, we mean CTI written a natural language such as English, and it contains a technical narrative of how an attack (such as an APT) unfolds.

Attack Behavior Graph (ABG). For the purpose of this work, what we refer to as a ABG is a directed acyclic graph (DAG) representation where the nodes capture entities such as programs/processes, files, network sockets etc. Edges represent actions/events such as execution, write, read, connect, send etc. An illustration of a ABG is shown in Figure 2-1. The edges are meant to capture information flow/causality among nodes depending on the direction.

Shape Notations: An oval represents a process/program (e.g., bash, sh, /bin/rm). A box refers to a file (e.g., /tmp/netrecon.log). A diamond represents network socket/IP address (e.g., 0-128.61.240.66 _9999). Edges are directed from active node such as a process to passive node such as a file/network socket, and edge direction indicates information flow (e.g., write, sendto) or causality (e.g., clone, execve).

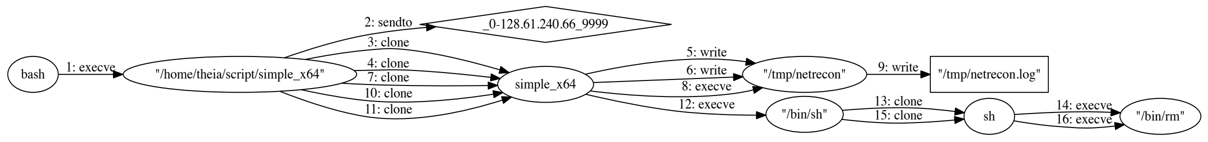


Figure 2-1: An illustration of an attack behavior graph structure used in this work

2.2 Related Work

Existing literature on analysis of CTI reports falls into two broad categories: *extraction and exchange of IOCs* and *attack behavior/narrative extraction*. In this chapter, we review related work in these two categories.

2.2.1 IOC Extraction and Exchange

IOC representation via the OpenIOC standard [1] is among the notable CTI exchange formats. Based on OpenIOC, several methods for automatically capture of IOCs from unstructured text and technical publications were developed. These include STIX [16] and MISP [3], with the flexibility of converting one to another. This is typically done by establishing a relation between the IOC token and other content in the CTI report, such as the terms “downloads”, “attachment” in **Error! Reference source not found.** adapted from [5].

2.2.2 Attack Behavior/Narrative Extraction

iACE [5]: This approach is an NLP-based semantic enrichment of basic IOC tokens leveraging named entity recognition (NER) and relationship extraction (RE). By employing a novel application of graph similarity comparison to anchor a phrase with probable IOC tokens and context terms, the approach quickly validates the validity of these elements using their relationships. This basic technique has been shown to be extremely effective, far exceeding the industry’s top-of-the-line IOC analyzer [4] and Named Entity Resolution (NER) tool [17] in terms of precision and coverage. Through analysis of over 71,000 security blogs, a substantial number of IOCs (over 900K) were extracted, and they include attack domains, hash values signatures, and IP addresses of malicious actors. While iACE is among the early methods to automatically enrich IOCs from CTI reports, it is limited to a single source (only security blogs) and its semantic enrichment is limited to sentence (misses intra- and inter-CTI report semantic link).

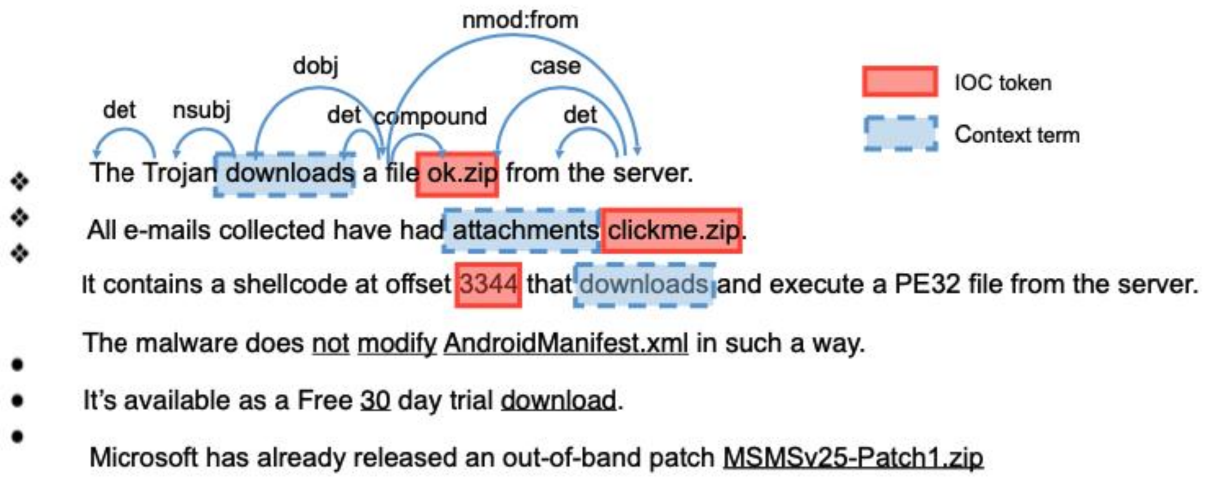

nmod:from
dobj
case
det
nsubj
det
compound
det
IOC token
Context term
The Trojan downloads a file ok.zip from the server.
All e-mails collected have had attachments clickme.zip.
It contains a shellcode at offset 3344 that downloads and execute a PE32 file from the server.
The malware does not modify AndroidManifest.xml in such a way.
It's available as a Free 30 day trial download.
Microsoft has already released an out-of-band patch MSMSv25-Patch1.zip

Figure 2-2: An example of relation between the IOC token and other content [5]

TTPDrill [7]: This work demonstrates the use of threat action ontology [7] and makes use of reports from technical CTI websites (e.g., Symantec Security Center). These reports are filtered based on some features such as number of words as articles containing Tactics, techniques, and procedures (TTPs) which usually have longer words as they help give a description of the attack, their threats, actions, and targets compared to other articles such as news on the web page and ads that are considerably shorter [18]. Another feature is securityAction-word density [19] i.e., extracting all verbs and calculate the percentage of appearance of verbs in an article compared to the total number of words in it and securityTarget-word density, i.e., extracting all security nouns, calculating the percentage of nouns in an article compared to the total number of words in the article. Each report contains links that are downloaded, some containing irrelevant information to

CTI such as ads, contact-us, etc. A document classifier, Support Vector Machine (SVM), is used to filter out unwanted content. There is a text sanitization process on the scraped texts by comparing the Document Object Model (DOM) trees to nodes that correlate to web elements such as images and HyperText Markup Language (HTML) tags to remove them.

ChainSmith [6]: This work presents extraction of IOCs from technical security articles by detecting based on multiple articles and not specific IoCs and categorizing them based on the malware delivery model as CTI reports contain noise, i.e., information not related to attack. The dataset used were collected from Symantec's WINE platform [20] because of the coverage on large number of hosts and similarity with other security vendors. The sources were selected on articles likely to contain detailed information about the attack and from diverse sources (blogs, news websites, etc.). ChainSmith makes use of python NLTK [21] and Stanford coreNLP [22]and uses word2vec [23] to parse words semantically. To get functional similarity instead of topical similarity on words, ChainSmith utilizes dependency-based word embedding. To improve the classification of IOCs by removing irrelevant words, a list of rules with regular expressions to identifying candidate IOCs are used. With known repetitions found in sentences when describing the same IOC, ChainSmith trains binary classifiers to identify topic probabilities and defines features to identify sentence topics that result in a reduced false positive rate.

GapFinder [24]: In this work, the proposed approach checks for semantic inconsistencies in CTI reporting and analyzes structured relations extracted from CTI sources with over 474K articles. It identifies technical inconsistencies that arise in the description of open-source malware by inferring similar terms for different words in unstructured reports. GapFinder derives structured relation-types and entity-types of the text. In the relation-type a malware dictionary is created to find sentences that contain malware names using Part of Speech (PoS) tagging and dependency parsing and constructs a malware graph based on the common node which refers to the malware name.

Extractor [9]: This approach is aimed to generate provenance graphs automatically from CTI reports, evaluate various threat reports and real-world attacks by extracting graphs that match those produced manually by security specialists. It solves one of the challenges of extracting IOCs from complex sentences seen in iACE[5] . Various NLP toolkits (e.g., Spacy [25], deep BiLSTM) are used on the unstructured texts of the CTI reports to ensure the output contains information relating to the attacks. Several analyses were done on the text of the CTI reports from text transformation to shorten long sentences to shorter forms which contain an action per sentence to homogenization (replacing multiple textual representations of the same concept to a common textual representation). The success of this paper was the several rounds of text transformations that enables to convert a highly complex sentence into a simpler form while still having the behavior of the attack i.e., the action and its context. *This thesis builds on Extractor with multi-source aggregation of attack behavior graphs as the novel contribution. In section 4.3.2, we compare CYTAG's aggregated graph results against Extractor's single-source attack behavior graph results*.

AttacKG [8]: This study proposes automatic extraction of structured attack behavior graphs from several CTI reports while identifying the adopted attack techniques. The key idea is due to the gap that exists in individual reports that give a limited description of an attack pattern. It scrapes CTI reports using NLP-based report parsing [26] to extract attack-relevant entities. In doing so, it preserves the information of the entities by replacing them with commonly used words to exclude special characters based on the entity type identified with an open-source IOC recognizer [4]. Similar to GapFinder [24], entities are separated into IOC and non-IOC entities with non-IOC classified into 7 types. With this classification a NER model is adopted to allow for better extraction and recognition of entities in CTI reports. To extract dependencies, a dependency tree is constructed using a learning-based natural language processing which parses each sentence as each entity establishes dependencies with an entity closest to it. Given the attack entities and dependencies, an attack graph is generated that describes the attack behaviors that appear in a CTI report using the MITRE template [27].

Chapter 3: Approach

This chapter first presents an overview of CYTAG in Section 3.1 and then details of CYTAG in Sections 3.2 - 3.5.

3.1 CYTAG Overview

The main goal in CYTAG is to enrich an attack behavior graph via systematic aggregation of individual attack graphs generated from individual CTI reports. As motivated Chapter 1, the key intuition behind CYTAG is that multiple CTI sources offer complementary IOCs that, if systematically aggregated, are crucial to result in a richer attack behavior graph that not only covers more ground but also enables capturing more attack signals when the aggregated graph is used as a basis for intrusion detection or forensic analysis. Towards this goal, given multiple CTI reports about the same attack (e.g., an APT), CYTAG operates in two phases: (*i*) attack behavior graph extraction and (*ii*) attack behavior graphs aggregation.

In phase (*i*), CYTAG learns the relationship between IOCs and the threat actions within a sentence using NLP techniques. To capture the characteristics of IOCs in CTI reports, of-the-shelf NLP techniques are not suitable for cybersecurity related domains. To address this limitation, in CYTAG, we instead rely on analysis of sentences from a dictionary of cybersecurity related words and adopt the attack behavior graph extraction technique in Extractor [9] to extract attack graphs from complex sentences containing IOCs.

In phase (*ii*), attack behavior graphs extracted from multiple sources are systematically aggregated to enrich a single-source attack behavior graph by adding new nodes and edges. This is done while preserving the overall attack behavior narrative and eliminating repetitions that result from overlaps among CTI sources.

Figure 3-1 shows an overview of CYTAG with elaborations on phases (*i*) and (*ii*). As part of phase (*i*), we perform data collection to acquire multiple CTI reports for each widely disclosed attack. In CTI, due to the unstructured manner of collecting sources, finding a centralized repository that consists of multiple CTI reports over the years is not easy. To fill this data acquisition gap, we begin with the CTI reports maintained in the OSINT Framework [28] and the APTnotes corpus [29] —a repository that contains CTI reports of prominent APTs spanning well over a decade (2008 – 2022). To augment the APTNotes dataset in a manner that enables us to collect multiple CTI reports for each APT, we build a web scrapper and filter. To sanitize each CTI report from non-IOC content, we apply DOM tree analysis to remove content that may lead to false positives (e.g., IP address/host name of the CTI source). The next step involves the preprocessing of each sentence to determine Subject-Verb-Object (SVO) triplets in each CTI report to serve as pre-cursors for extraction of attack behavior attack graphs. At the core of the graph extraction method is Semantic Role Labeling (SRL), which assigns role-related labels (e.g., subject, object, action) to enable the extraction of the attack behavior graph.

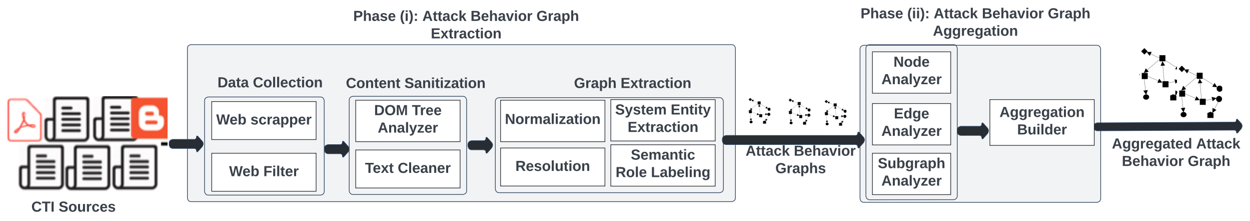


Figure 3-1: Overview of the CYTAG pipeline.

As part of phase (*ii*), given multiple attack behavior graphs of the same CTI report, we analyze the nodes and edges of the generated graphs, and more importantly subgraphs in each graph that can be stitched together to result in a semantically-rich and comprehensive attack behavior graph. The main intuition here is that by systematically aggregating complementary attack behavior captured across multiple sources, CYTAG offers improved detection and forensics capabilities when the aggregated attack behavior graph is used. It is noteworthy that in order to enrich the aggregated graph with threat-relevant graph elements we use a varying range of CTI sources that involve diverse threat actors such as malware, botnets, and ransomware. After the graph aggregation, the effectiveness is evaluated by comparing the aggregated graph against the individual graphs generated by Extractor [9] and graphs generated by AttacKG[8] —a recent work that builds on Extractor.

In the rest of this chapter, we expand our discussion on the different components of CYTAG.

3.2 CTI Dataset Collection

The common problem faced in CTI information gathering is that CTI reports are come in heterogeneous forms such as white-papers, technical reports, and blog posts.

Moreover, they come as a mix of structured (e.g., CSVs) and mostly unstructured (e.g., natural language text). Although there exists an inventory of CTI reports via the centralized OSINT Framework which provides APTnotes [28], there are several other CTI reports that are not accounted for because of reasons such as authors naming of the CTI title differently, reports coming much later than the first CTI report, or simply overlooked by the maintainers of OSINT and APTNotes.

Enhancement to CTI Data Collection: To acquire a more comprehensive CTI dataset for CYTAG, we scale up the data collection through a combination of headless browsing, web scrapping, file format conversion, and automated search. Leveraging the popular headless browser automation engine, Selenium [30], we automate the process of visiting and rendering a CTI source URL. Using the Beautiful Soup [31]web scrapping engine, we parse the web pages of CTI reports rendered by Selenium and extract data to a text format and use it as a universal intermediate representation for processing CTI reports. Using PDFMiner [32], we convert CTI reports in PDF to text format. We leverage automated search APIs where we submit as search query the title of a CTI report or name of a high-profile APT (e.g., one of the names from APTNotes) and focus on URLs of the top search results from search engines such as Google and Bing.

3.3 CTI Content Sanitization

After the dataset collection step, to avoid including content like website name that do not pertain to the attack, the dataset is cleaned to remove details about the site such as promotions, contact details, login pages etc. This is done by analyzing the DOM tree of each website to understand the common HTML tags to be included in the scrapped data. Furthermore, an NLP-based text cleaner is designed to remove white spaces from the saved text files by analyzing each sentence in the text file and reducing the length. Doing so further reduces the graph generation time as the system goes through each sentence in the text file.

3.4 Attack Behavior Graph Extraction

To achieve extraction of graphs from the sanitized text of CTI reports, we build on the state-of-the-art approach in Extractor [9]. The design of Extractor is powered by NLP techniques such as normalization, resolution, summarization, and semantic role labeling, all of which are tailored for CTI report analysis.

The first round of transformation that simplifies long sentences to a canonical form by breaking them into shorter sentences is Normalization, while resolution handles text ambiguity. Then, summarization removes text in the sentences that are not related to the attack behavior i.e., do not contribute to the semantics of the attack narrative. The output of the summarized texts are in the form of Subject, Verb, and Object (SVO). Finally, the graph extraction uses semantic Role labelling (SRL) to discover the roles of words in sentences, grouping them into arguments and a set of dictionaries (system call dictionary and CTI noun dictionary) to properly represent the verbs and actions in the sentences.

Enhancement to Extractor: This thesis improves the extraction accuracy of Extractor [9] in two ways. First, since Extractor does not support extraction of IOCs from images embedded in CTI reports, we add this feature to enrich the scope of the analysis. We do so by leveraging the tesseract library for text extraction from images [33]. Second, we add more words in the set of dictionaries needed for identifying the verbs found in the sentences as security reporters have varying vocabulary.

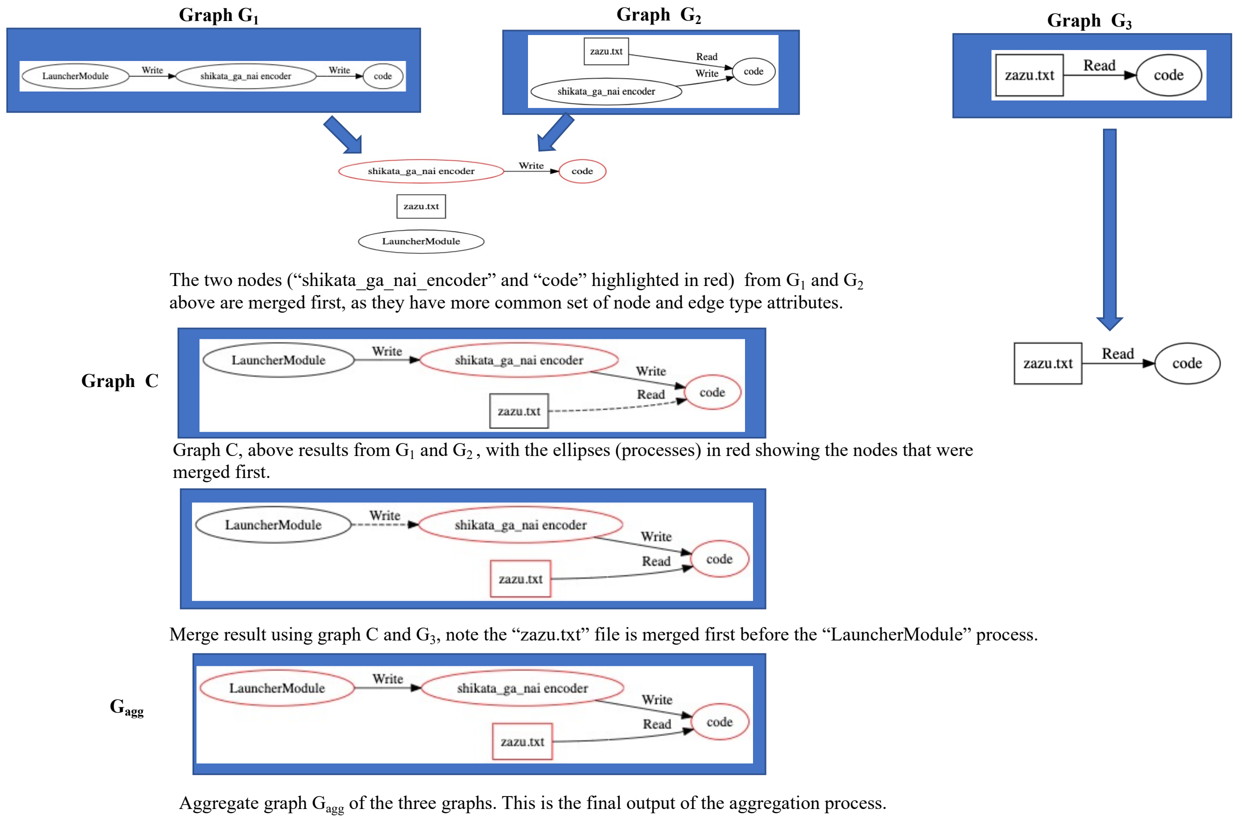
3.5 Attack Behavior Graphs Aggregation

In this section, we highlight the properties of a graph (G) based on Nodes (V) and Edges(Y) and present solutions from [34] to implement on a set of graphs N(G) and generate aggregated attack graphs with no overlapping nodes or edges while the attack description (semantics) of the APT and likewise improving it in an efficient and effective method.

Setting: Given a set of graphs ($G^1,...,G^n$) each graph G^i has multiple nodes and edges represented as V= ($V_1,...,V_m$), Y= ($Y_1,...,Y_k$) respectively, with G^1 represented as $G^1(V_1^1, ...V_m^1, Y_1^1, ...Y_k^1)$ and G^n as ($V_1^n, ..., V_m^n, Y_1^n, ..., Y_k^n$). Note that n is number of graphs (e.g., n = 3 if we generate attack graphs from three distinct CTI sources), m is number of nodes for a given graph, and k is number of edges for a given graph. The values of m and k vary depending on the size of the graph at hand.

In addition, each node has a set of attributes (e.g., node type) represented as A_V. Similarly, A_Y represents the set of attributes for edges that determine the relationship (edge label) to form a node pair between the respective nodes. With several Nodes (V) and Edges (Y), the nodes and edges have relationships of the form (n_i – edge – n_j), that connect node n_i with node n_j to form node pairs.

CYTAG Aggregation Intuition: Before sketching the approach formally, below we give an intuitive illustration of how the graph aggregation is done in CYTAG. Suppose we are given three sources of an APT report represented as attack graphs G_1, G_2, and G_3. In Figure 3-2, we have G_1 generated from a CTI snippet of the LuckyMouse APT in 2018 from first source (APTNotes [29]). Likewise, G_2 and G_3 are graphs generated from CTI snippets from two other sources of the same APT. As defined in Section 2.1, the nodes the graphs are represented as ellipse for processes/programs and using box for files.



The two nodes (“shikata_ga_nai_encoder” and “code” highlighted in red) from G_1 and G_2 above are merged first, as they have more common set of node and edge type attributes.

Graph C, above results from G_1 and G_2, with the ellipses (processes) in red showing the nodes that were merged first.

Merge result using graph C and G_3, note the “zazu.txt” file is merged first before the “LauncherModule” process.

Aggregate graph G_{agg} of the three graphs. This is the final output of the aggregation process.

Figure 3-2: Intuitive illustration of attack behavior graph aggregation in CYTAG.

CYTAG Aggregation Method: Algorithm 1 sketches the core of the multisource attack behavior aggregation approach in CYTAG. Notations used in the algorithm are defined in Table 3-1. For the sake of illustration, the algorithm is demonstrated for three graphs G_1, G_2, and G_3, but in principle the idea should work for *n* graphs generated from *n* CTI sources of a certain APT. The aggregation happens progressively by taking two graphs at a time and using the aggregation of the two graphs as a base for the next cycle of aggregation, until no individual graph remains to be aggregated. As line 1 shows, G_1 and G_2 are first taken to find a grouping of nodes from the two graphs by based on homogeneous edges using node and edge attributes respectively. Line 2 is the iterative part of the algorithm where the process of grouping continues until no further grouping is feasible. Line 3 shows what happens within the loop of line 2, where the actual union of two graphs at hand is performed to generate a temporary composition *C*, which will be used as a base graph for the next grouping and then union. After each union computation, notice in line 7 that the graph structure needs to be updated so as to document the progress of the aggregation. Finally, line 9 produces the aggregated graph G_{agg}.

Table 3-1: Notations in graph aggregation.

Notation	Description
Φ	The aggregation function performs the merge of two graph inputs.
G_n	The number of graph inputs
V	Node representation of graphs.
Y	Edge representation of graphs
A_V	Set attributes of node-types such as file, process, etc.
C	Initial output obtained from the union of two graphs.
G_{agg}	Aggregated graph output of Φ
A_Y	Edge-type attribute e.g., write, read, send, execute etc.

Algorithm 1 Φ (G_n, V, Y, A_V, A_Y, C, G_{agg})

```
        Input: $G_1$, $G_2$, $G_3$: 3 graphs; $A_V \subseteq A(G_n)$: a set of node attributes; $A_Y \subseteq A(G^n)$ : a set of edge attributes;
        Output: An aggregate graph, $G_{agg}$
1: Compute the compatible grouping of $G_1$ and $G_2$ by identifying nodes from both graphs alongside the edges.
2: while there are node and edge grouping, set respective attributes types $A_V, A_Y$
   do
3:      Compute the union on $G_1$ and $G_2$, generate C
4:      Initialize the graph structures
5:      With C and $G_3$, begin node and edge grouping and assign attributes based on node, edge relationship
6:      Compute the union of C and $G_3$
7:      Update the graph structures
8: end while
9: Form the aggregate graph $G_{agg}$
10: return $G_{agg}$
```

Baseline aggregation alternative: In the evaluation in Section 4.3.1, we will compare the aggregation in Algorithm 1 (which we call "Aggregation Method I") with a baseline aggregation which we call "Aggregation Method II". In the baseline approach, instead of systematically performing the aggregation as in Algorithm 1, we take a rather simplistic approach of merging the sanitized text representation of each CTI source of an APT into one file and feed it to the enhanced Extractor pipeline. Intuitively, the graph generated from the merger of multiple sources needs to capture a good chunk of the threat semantics. However, it is also likely to result in redundant nodes and edges and is potentially error-prone because it is a naive merging of the CTI text from different sources.

Chapter 4: Evaluation and Results

This chapter presents the dataset, experimental setup, and evaluation results of CYTAG. The evaluation is guided by the following research questions:

- **RQ1**: How do the two aggregation methods compare?
- **RQ2**: How effective is the aggregation of multiple threat behavior graphs in CYTAG compared to state-of-the-art graph attack generation?
- **RQ3**: What is the runtime overhead of CYTAG graph aggregation?

4.1 Dataset

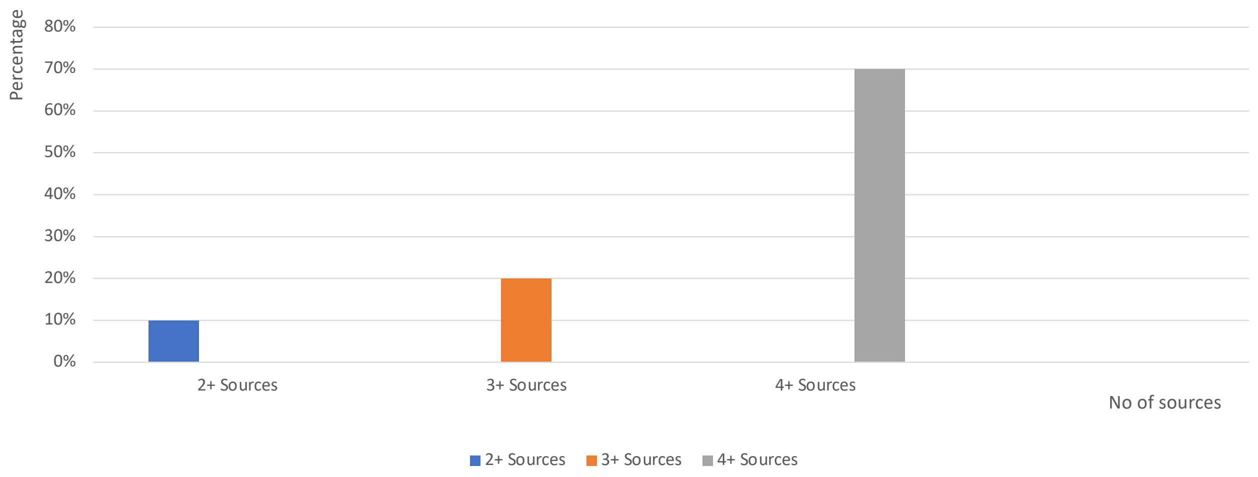


Figure 4-1: Percentage of CTI reports based on number of unique sources

To evaluate CYTAG, we collected CTI reports summarized in Table 4-1. The CTI reports are categorized into years from 2010 to 2021. These reports cover several APTs spanning diverse threat vectors such as malware, ransomware, botnets, and spyware attacks. The source of the dataset is the OSINT used by previous works [3] for IOC analysis and graph generation. In total, we collect 1456 reports with at least 2 sources per APT report. Figure 4-1 shows the distribution of number of distinct CTI sources. About 70% of APT reports have 4 distinct CTI sources while about 20% have 3 distinct sources. The remaining 10% have 2 sources as can be seen from the Figure. From the APTNotes dataset [29], we collected the individual PDF versions of the APTs and then expanded each with APT reports from other CTI sources. To do so, we employ the data collection enhancement described in Section 3.2 which takes advantage of a web scrapping, headless browsing, and automated search.

Table 4-1: Summary of CTI reports dataset

Year	# of CTI Reports from [28]	# of Multi-sources	Total
2010	10	23	33
2011	15	33	48
2012	25	62	87
2013	44	100	144
2014	90	186	276
2015	65	180	245
2016	52	108	160
2017	80	172	252
2018	21	52	73
2019	18	45	63
2020	4	10	14
2021	15	46	61
Total	**439**	**1017**	**1456**

4.2 Experimental Setup

CYTAG is implemented in Python reusing graph extraction components from Extractor [9]. Next, we describe our experimental setup.

- **Comparison of aggregation techniques**: We generated attack graphs using both compiled CTI report input and individual CTI report input fed into CYTAG to compare the node and edge comparison of the respective outputs.
- **Comparison with related work**: We compare CYTAG with the original and enhanced Extractor [9]. The enhanced Extractor comes with extraction of IOCs from images in CTI reports and expansion of the verbs in the action dictionary used in the graph extraction pipeline.

Comparison of Aggregation Techniques: In this experiment, we compare the output of the aggregation-by-aggregation approach I and II. For approach I, we extract the attack graphs of each CTI source of each APT and save them with their APT title as $[A(G_1),....,A(G_n)]$. The output of the graph extraction is saved in three formats: PDF (for visual inspection of attack progression), DOT (for automated analysis during aggregation), and JSON (to ease feeding an attack graph to IDSs). In approach II, we take a single CTI report compiled from multiple sources of each APT as the input to Extractor and generate an attack graph. The new aggregated attack graph with the compiled source $A(G_c)$ is produced in PDF and DOT format. For each APT, the respective DOT files of approaches I and II are then analyzed to compare their effectiveness based on the difference in nodes and edges between the two approaches.

Comparison with Related Work: In this experiment, we evaluate CYTAG and Extractor [9] (by reproducing the public source code [1]) CTI reports used by the authors of Extractor. Specifically, we first analyze the differences in the nodes and edges of the respective graphs determining which has more information about the attack in question.

[1] https://github.com/ksatvat/Extractor

Runtime Overhead Measurement: In this thesis, we ran the implementation of CYTAG to gather the datasets, pre-process CTI reports, generate the attack graphs, and perform the aggregation on an 8-core CPU, 8-core GPU 16-core Neural Engine Apple M1 chip and 16GB of memory. To measure the performance overhead 12 of the 16GB memory was used.

4.3 Results and Discussion

This section presents the results of the experiments described above. Firstly, we show the comparison results of the aggregation approaches I and II. Second, we compare the performance of CYTAG with Extractor. Lastly, we report on runtime overhead of CYTAG.

4.3.1 Comparison of Aggregation Techniques

To test the effectiveness of the aggregation methods on the dataset from Table 4-1 we ran approach I, which takes each extracted graphs from multiple CTI reports and systematically analyses the attributes of the graphs to generate the aggregated graph in a manner that prevents overlapping edges and repeated nodes. In the results shown in Table 4-2, although approach II produces a larger graph output with an increased number of nodes and edges. Further analysis of the graph output by approach II, however, suggests that there are node repetitions in the DOT file. While the larger graph seems appealing on the surface, we note that such redundancy of nodes and edges could potentially slow-down searching/traversal by an IDS or performing forensics after attack detection. The conclusion is that the naive approach is not only less precise but also more expensive computationally.

Take-away: With respect to RQ1, our conclusion of the comparison between approach I and II suggests that although the output of approach II showed more node output by an average of 5% across the CTI reports, this difference was because of the repetitive entities and edges present in the result which would make the use of approach II in practice less feasible.

Table 4-2: Comparison of aggregation methods I and II

CTI Source	Aggregation Method I		Aggregation Method II	
	|V|	|Y|	|V|	|Y|
OceanLotus [35]–[37]	415	261	453	386
GhostDuke [38]–[40]	342	220	400	283
SeaTurtle [41]–[43]	257	196	266	250
Tortoiseshell [44]–[46]	202	285	251	334
Waterbug [47]–[49]	329	214	352	304

4.3.2 Comparison with Closely Related Work

Our comparison results are summarized in Table 4-3. The first column is the name of the CTI report as publicly documented. Note that, for fair comparison, we focus on five CTI reports that are also used by Extractor. The next six columns are the number of nodes V(G) and edges Y(G) of graphs extracted with original Extractor, CYTAG Graph Generator, and the nodes and edges of the aggregated graph generated with CYTAG. The "CYTAG Sources" column indicates the two additional CTI sources considered in addition to the one from APTNotes (shown in first column) —making it uniformly 3 CTI sources across the board. The last column shows the year of the two additional CTI sources respectively, overall covering 2013, 2015, 2016, and 2019.

As can be seen from the Table 4-3, the number of nodes and edges increased after improving the graph extraction process of this approach. On average there was an increment of about 12% and 32% in the nodes and edges of the graphs generated. The enhanced graph extraction process helped in getting better result with CYTAG, after generating the respective graphs of the sources G_n for the aggregation process $\Phi(G_n)$.

In Table 4-3, we can see that the number of Edges (Y) and Nodes (V) have increased significantly, which enhances the attack semantics. We observed a high node and edge addition with the aggregation of the graphs. The increased number of nodes and edges in the attack graph are a result of the improved source input that offers complementary attack narrative. Figure 4-2 and Figure 4-3 demonstrate the various APTs and show the increase in Nodes and Edges for CYTAG.

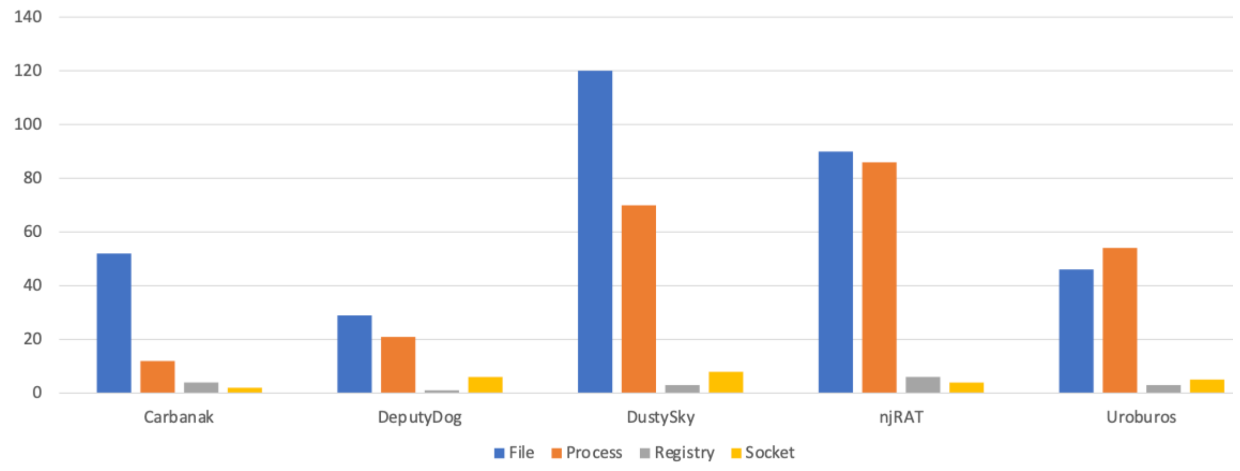


Figure 4-2: Node-type distribution of CTI reports

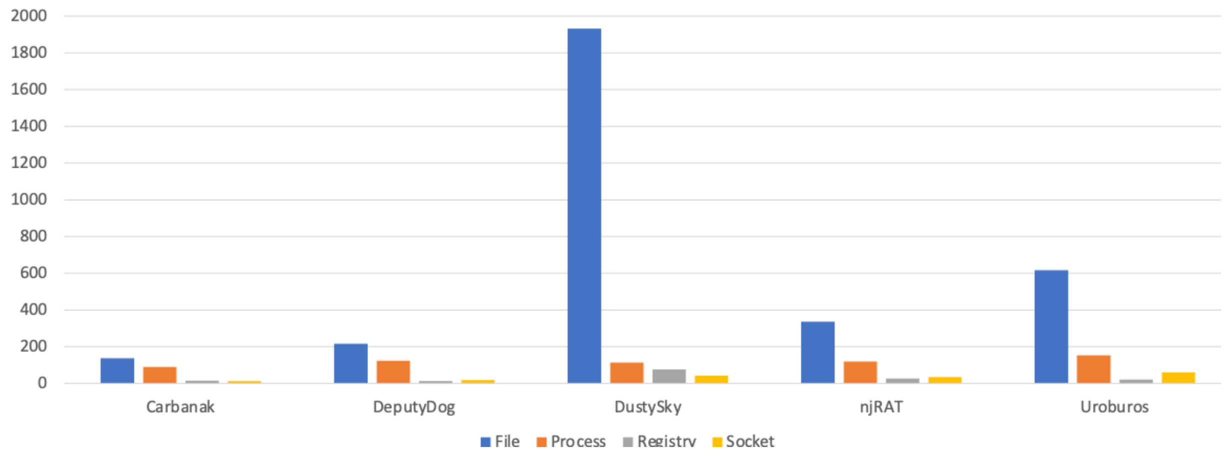


Figure 4-3: Distribution of node-types in merged/compiled CTI reports

Table 4-3: Effectiveness of Extractor, CYTAG Graph Generator, and CYTAG

CTI Source	Extractor		CYTAG Graph Generator		CYTA G		CYTAG Sources	CTI Report Year
	$\|V\|$	$\|Y\|$	$\|V\|$	$\|Y\|$	$\|V\|$	$\|Y\|$		
Carbanak [50]	22	31	42	99	64	113	[51], [52]	2019,2019
Deputy Dog [53]	11	12	16	24	27	49	[54], [55]	2013,2013
Dusty Sky [56]	12	21	30	58	178	344	[57], [58]	2016,2015
njRAT [59]	32	32	43	67	85	193	[60], [61]	2013,2019
Uroburos [62]	19	23	26	33	97	178	[63], [64]	2015,2015

Take-away: With respect to **RQ2**, as can be seen from the results in Table 4-3, the number of nodes and edges in the threat behavior graph generated by CYTAG is on average more than double of the results of Extractor, which speaks to the overall effectiveness of the CYTAG aggregation approach.

4.3.3 Cytag runtime overhead

To answer **RQ3**, we ran CYTAG on dataset from selected years (2013, 2015, 2018, 2019) shown in Figure 4-4. The goal is to measure the average time it takes to aggregate attack graphs across the datasets. These years were used to show the efficiency of CYTAG on current APT attacks and previous ones. From the result shown in Figure 4-5, the time it takes for a year is dependent not only on the number of CTI reports, but also the content of the attack description. On average, it took approximately 1101 seconds to complete the aggregation of about over 131 CTI sources with 2-4 sources per an aggregation. The per-aggregation average compute time to generate the aggregated graph for CTI reports in 2013, 2015, 2018, and 2019 were 1582, 2240, 318, and 264 in seconds, respectively.

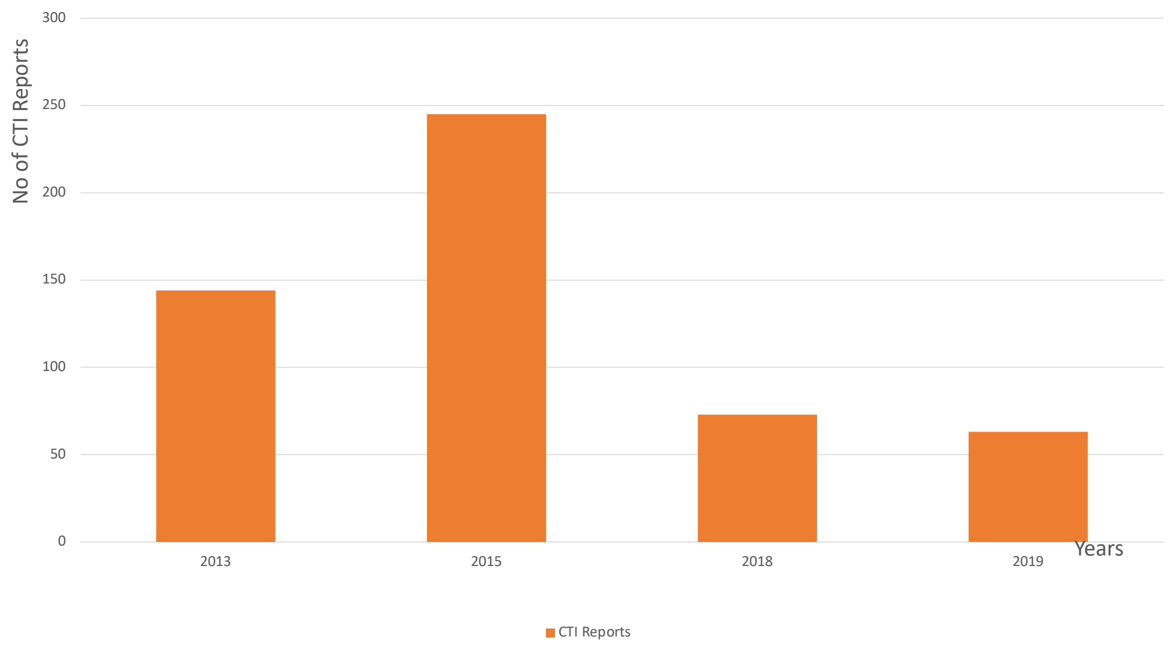


Figure 4-4: CTI reports analyzed with CYTAG to measure runtime overhead

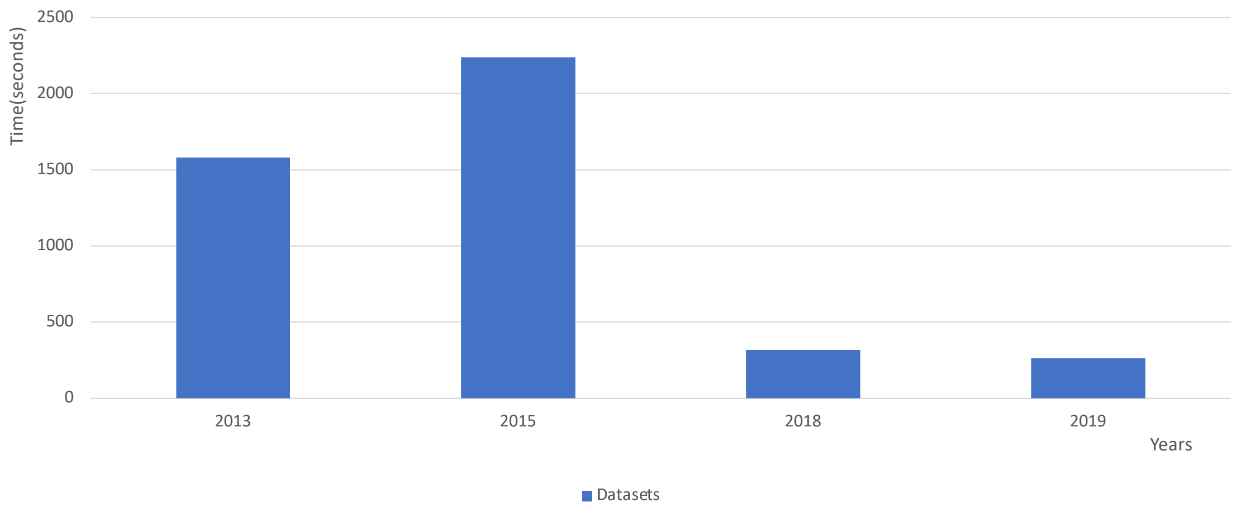


Figure 4-5: Average time taken for CYTAG to generate aggregated attack graph

4.3.4 Overall evaluation summary

Our evaluation suggests that CYTAG (a) outperforms the current state-of-the-art attack graph extraction, (b) enriches attack graphs through aggregation of graphs generated from multiple complementary CTI reports, and (c) incurs reasonable runtime overhead for a one-time (once in a while in the worst case) aggregation of attack graphs. Overall, CYTAG proved that systematically aggregating attack graphs from multiple CTI reports improves the semantic utility of a threat behavior graph towards improved detection and forensics of sophisticated attack campaigns that target enterprises or nation-states.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

This thesis presented CYTAG, a new contribution for multi-source aggregation of attack behavior graphs to improve cyber threat hunting capabilities using natural language cyber threat intelligence. CYTAG significantly enhances the fidelity of an attack graph given multiple CTI sources about a given attack such as an APT and achieves this while preserving attack semantics and minimizing redundancy of nodes and edges in the aggregated attack graph.

By evaluating CYTAG on diverse CTI reports covering multiple years and comparing its attack graph aggregation results with state-of-the-art attack behavior extraction approach, we demonstrate that CYTAG significantly improves the detection and forensics arsenal of cyber threat hunters with reasonable aggregation performance overhead. We applied CYTAG on 1K+ dataset of CTI reports which resulted in an improved attack graph extraction rate of an average of 21% more than Extractor [9] (current state-of-the-art attack behavior graph extraction approach).

5.2 Future work

Our study shows that CYTAG is able to generate aggregated attack graphs containing enhanced attack semantics from multiple sources, well beyond the state-of-the-art techniques can achieve. However, still our technique is not operating at a scalable performance for real-time detection. These problems in our study come from the limitation of underlying tools and techniques we use. Specifically, in the aggregation phase the use of a Graph Neural Network (GNN) with deep learning to further analyze graph structural data could improve the aggregation time significantly.

Also, as CYTAG uses some graph structure attributes for the grouping, adding more attributes for the grouping of nodes and edges in graphs can improve the aggregation. Another interesting observation as seen from the results is that an improved dictionary resulted in better graph generation ---which entails the need to continuously update the dictionary. Furthermore, in the data collection phase, our technique can be improved further by firstly finding more repositories of CTI reports to use the titles to perform the web scrapping. Secondly, during the web scrapping, the search result scraped should include more than the first page. For example, during the manual search of some CTI titles, some useful articles were found on the second page of the search result output. In addition, an initial cleaning of articles selected for the graph aggregation because of large number of inconsistent articles might lead to false discoveries. Future efforts are still required to tackle these issues.

In summary, while CYTAG produces aggregated attack behavior graphs from multiple CTI sources, and the aggregated graphs carry enhanced attack semantics and forensic evidence, we identify the following avenues to further enhance CYTAG:

- Using CYTAG aggregated attack behavior graphs, testing its effectiveness on a real/simulated intrusion detection system to filed-test its attack detection effectiveness.
- Further reduction of the aggregation time to enable scalable performance of CTI analysis in a real-time scenario.
- Experiment on signature detection and attack patterns from bad actors for novel attack behavior discovery.

References

[1] FireEye, “The OpenIOC Framework.” https://fireeye.market/apps/211404 (accessed Aug. 07, 2022).

[2] B. Jordan, R. Piazza, and T. Darley, “STIX - Structured Threat Information Expression (Archive) | STIX Project Documentation.” https://stixproject.github.io/ (accessed Aug. 07, 2022).

[3] C. Wagner, A. Dulaunoy, Wagener Gérard, and A. Iklody, “MISP Open Source Threat Intelligence Platform &amp; Open Standards For Threat Information Sharing.” https://www.misp-project.org/ (accessed Aug. 07, 2022).

[4] A. Buescher, “armbues/ioc_parser: Tool to extract indicators of compromise from security reports in PDF format.” https://github.com/armbues/ioc_parser (accessed Aug. 07, 2022).

[5] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, “Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence,” in *Proceedings of the ACM Conference on Computer and Communications Security*, Oct. 2016, vol. 24-28-October-2016, pp. 755–766. doi: 10.1145/2976749.2978315.

[6] Z. Zhu and T. Dumitras, “ChainSmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports,” in *Proceedings - 3rd IEEE European Symposium on Security and Privacy, EURO S and P 2018*, Jul. 2018, pp. 458–472. doi: 10.1109/EuroSP.2018.00039.

[7] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, “TTPDrill: Automatic and accurate extraction of threat actions from unstructured text of CTI Sources,” in *ACM International Conference Proceeding Series*, Dec. 2017, vol. Part F132521, pp. 103–115. doi: 10.1145/3134600.3134646.

[8] Z. Li, J. Zeng, Y. Chen, and Z. Liang, “AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports,” Nov. 2021, [Online]. Available: http://arxiv.org/abs/2111.07093

[9] K. Satvat, R. Gjomemo, and V. N. Venkatakrishnan, “EXTRACTOR: Extracting Attack Behavior from Threat Reports,” Apr. 2021, [Online]. Available: http://arxiv.org/abs/2104.08618

[10] O. Catakoglu, M. Balduzzi, and D. Balzarotti, "Automatic extraction of indicators of compromise for web applications," *25th International World Wide Web Conference, WWW 2016*, pp. 333–343, 2016, doi: 10.1145/2872427.2883056.

[11] S. M. Milajerdi, R. Gjomemo, B. Eshete, and V. N. Venkatakrishnan, "Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1795–1812, Nov. 2019, doi: 10.1145/3319535.3363217.

[12] Wikipedia, "Advanced persistent threat - Wikipedia." https://en.wikipedia.org/wiki/Advanced_persistent_threat (accessed Aug. 07, 2022).

[13] S. Momeni Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: Real-time APT detection through correlation of suspicious information flows," *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2019-May, pp. 1137–1152, May 2019, doi: 10.1109/SP.2019.00026.

[14] R. McMillan, "Definition: Threat Intelligence." https://www.gartner.com/en/documents/2487216 (accessed Aug. 07, 2022).

[15] L. Obrst, P. Chase, R. Markeloff, and B. Bedford, "Developing an Ontology of the Cyber Security Domain".

[16] R. M. Czekster, R. Metere, and C. Morisset, "cyberaCTIve: a STIX-based Tool for Cyber Threat Intelligence in Complex Models," Apr. 2022, doi: 10.48550/arxiv.2204.03676.

[17] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification", Accessed: Aug. 07, 2022. [Online]. Available: http://projects.ldc.upenn.edu/gale/

[18] V. M. Igure and R. D. Williams, "Taxonomies of attacks and vulnerabilities in computer systems," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 1, pp. 6–19, Mar. 2008, doi: 10.1109/COMST.2008.4483667.

[19] M.-C. de Marneffe and C. D. Manning, "The Stanford typed dependencies representation", Accessed: Aug. 07, 2022. [Online]. Available: http://www.w3.org/RDF/

[20] T. Dumitras and D. Shou, "Toward a standard benchmark for computer security research: The worldwide intelligence network environment (WINE)," *Proceedings of the 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2011*, pp. 89–96, 2011, doi: 10.1145/1978672.1978683.

[21] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," *COLING/ACL 2006 - 21st International Conference on Computational Linguistics and 44th*

Annual Meeting of the Association for Computational Linguistics, Proceedings of the Interactive Presentation Sessions, pp. 69–72, May 2002, doi: 10.48550/arxiv.cs/0205028.

[22] Stanford NLP Group, "stanfordnlp/CoreNLP: Stanford CoreNLP: A Java suite of core NLP tools." https://github.com/stanfordnlp/CoreNLP (accessed Aug. 07, 2022).

[23] T. Mikolov, I. Sutskever, K. Chen, Greg S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality." https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html (accessed Aug. 07, 2022).

[24] H. Jo, J. Kim, P. Porras, V. Yegneswaran, and S. Shin, "GapFinder: Finding Inconsistency of Security Information from Unstructured Text," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 86–99, 2021, doi: 10.1109/TIFS.2020.3003570.

[25] M. Honnibal *et al.*, "explosion/spaCy: v2.1.0: New models, ULMFit/BERT/Elmo-like pretraining, faster tokenization, better Matcher, bug fixes &amp; more." Zenodo, 2019. doi: 10.5281/ZENODO.2597447.

[26] S. Bird, E. Klein, E. Loper, and W. Wagner, "Steven Bird, Ewan Klein and Edward Loper: Natural Language Processing with Python, Analyzing Text with the Natural Language Toolkit," *Language Resources and Evaluation 2010 44:4*, vol. 44, no. 4, pp. 421–424, May 2010, doi: 10.1007/S10579-010-9124-X.

[27] The MITRE Corporation, "MITRE ATT&CK®." https://attack.mitre.org/ (accessed Aug. 07, 2022).

[28] s0lray, mrpnkt, R. Romanov, and J. Nordine, "lockfale/OSINT-Framework: OSINT Framework." https://github.com/lockfale/osint-framework (accessed Aug. 07, 2022).

[29] K. Bandla and S. Castro, "aptnotes/data: APTnotes data." https://github.com/aptnotes/data (accessed Aug. 07, 2022).

[30] Selenium Open Source community, "WebDriver | Selenium." https://www.selenium.dev/documentation/webdriver/ (accessed Aug. 07, 2022).

[31] L. Richardson, "Beautiful Soup Documentation Release 4.4.0," 2019.

[32] Y. Shinyama, "pdfminer · PyPI." https://pypi.org/project/pdfminer/ (accessed Aug. 07, 2022).

[33] Matthias A Lee, "madmaze/pytesseract: A Python wrapper for Google Tesseract." https://github.com/madmaze/pytesseract (accessed Aug. 07, 2022).

[34] A. Hagberg, P. Swart, and D. Chult, “(PDF) Exploring Network Structure, Dynamics, and Function Using NetworkX.” https://www.researchgate.net/publication/236407765_Exploring_Network_Structure_Dynamics_and_Function_Using_NetworkX (accessed Aug. 07, 2022).

[35] BALAJI N, “OceanLotus Group Using Steganography Techniques to Deliver Malware.” https://gbhackers.com/oceanlotus-apt-hackers-group-steganography/ (accessed Aug. 07, 2022).

[36] Cylance, “OceanLotus Steganography Malware Analysis White Paper | GoVanguard Threat Center.” https://govanguard.com/threat-center/2019/04/05/oceanlotus-steganography-malware-analysis-white-paper/ (accessed Aug. 07, 2022).

[37] P. Paganini, “OceanLotus leverages a steganography-based loader to deliver backdoors.” https://securityaffairs.co/wordpress/83246/breaking-news/oceanlotus-steganography-backdoors.html (accessed Aug. 07, 2022).

[38] M. Faou, M. Tartare, and T. Dupuy, “OPERATION GHOST The Dukes aren’t back-they never left,” 2019.

[39] ESET Research, “Operation Ghost II: The Dukes aren’t back – they never left | WeLiveSecurity.” https://www.welivesecurity.com/2019/10/17/operation-ghost-dukes-never-left/ (accessed Aug. 07, 2022).

[40] Cyware Hacker News, “Operation Ghost: Research Finds Dukes Group Never Stopped Espionage Activities | Cyware Alerts - Hacker News.” Accessed: Aug. 07, 2022. [Online]. Available: https://cyware.com/news/operation-ghost-research-finds-dukes-group-never-stopped-espionage-activities-5bc12ea

[41] D. Adamitis and P. Rascagneres, “Cisco Talos Intelligence Group - Comprehensive Threat Intelligence: Sea Turtle keeps on swimming, finds new victims, DNS hijacking techniques.” https://blog.talosintelligence.com/2019/07/sea-turtle-keeps-on-swimming.html (accessed Aug. 07, 2022).

[42] CTM360, “Dns hijacking abuses trust in core internet service.” https://www.ctm360.com/ (accessed Aug. 07, 2022).

[43] W. Mercer and P. Rascagnères, “VB2019 paper: DNS on fire.” https://www.virusbulletin.com/virusbulletin/2019/11/vb2019-paper-dns-fire/ (accessed Aug. 07, 2022).

[44] Fahmida Y. Rashid, “Tortoiseshell Targets IT Providers in Supply Chain Attack.” https://duo.com/decipher/tortoiseshell-targets-it-providers-in-supply-chain-attack (accessed Aug. 07, 2022).

[45] P. Paganini, “TortoiseShell Group targets IT Providers in supply chain attacks.” https://securityaffairs.co/wordpress/91611/apt/tortoiseshell-supply-chain-attacks.html (accessed Aug. 07, 2022).

[46] Threat Hunter Team Symantec, “Tortoiseshell Group Targets IT Providers in Saudi Arabia in Probable Supply Chain Attacks.” https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/tortoiseshell-apt-supply-chain (accessed Aug. 07, 2022).

[47] Eduard Kovacs, “Waterbug Threat Group Targeted Systems in Over 100 Countries.” https://www.securltyweek.com/waterbug-threat-group-targeted-systems-over-100-countries-symantec (accessed Aug. 07, 2022).

[48] Symantec DeepSight Adversary Intelligence Team, “Waterbug: Espionage Group Rolls Out Brand-New Toolset in Attacks Against Governments.” https://www.cyberreport.io/news/waterbug-espionage-group-rolls-out-brand-new-toolset-in-attacks-against-governments?article=2449 (accessed Aug. 07, 2022).

[49] Network Protection Security Labs, “Waterbug: Espionage Group Rolls Out Brand-New Toolset in Attacks Against Governments | Broadcom Software Blogs.” https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/waterbug-espionage-governments (accessed Aug. 07, 2022).

[50] K. Goncharov, “The Great Bank Robbery: Carbanak APT | Kaspersky official blog.” https://www.kaspersky.com/blog/the-great-bank-robbery-carbanak-apt/3598/ (accessed Aug. 07, 2022).

[51] M. Bailey and James T. Bennett, “CARBANAK Week Part Two: Continuing the CARBANAK Source Code Analysis | Mandiant.” https://www.mandiant.com/resources/carbanak-week-part-two-continuing-carbanak-source-code-analysis (accessed Aug. 07, 2022).

[52] FireEye, “CARBANAK-Week-3-Behind-CARBANAK-Backdoor.” https://app.box.com/s/rel5slouyleepdl3u7xilbmzsuk091n3 (accessed Aug. 07, 2022).

[53] P. Paganini, “FireEye revealed APT Operation DeputyDog against Japanes entities.” https://securityaffairs.co/wordpress/17975/hacking/fireeye-operation-deputydog-japan.html (accessed Aug. 07, 2022).

[54] D. Caselden, “Operation DeputyDog Part 2: Zero-Day Exploit Analysis (CVE-2013-3893) | LaptrinhX.” https://laptrinhx.com/operation-deputydog-part-2-zero-day-exploit-analysis-cve-2013-3893-3911306784/ (accessed Aug. 07, 2022).

[55] FireEye, “Operation DeputyDog Part 2: Zero-Day Exploit Analysis (CVE-2013-3893).” https://www.fireeye.fr/blog/threat-research/2013/09/operation-deputydog-part-2-zero-day-exploit-analysis-cve-2013-3893.html (accessed Aug. 08, 2022).

[56] S. Ginty, “Operation DustySky Notes | RiskIQ.” https://www.riskiq.com/blog/analyst/operation-dustysky-notes/ (accessed Aug. 08, 2022).

[57] Cybereason, “Operation Dustysky.” https://www.cybereason.com/blog/what-is-operation-dustysky (accessed Aug. 07, 2022).

[58] ClearSky Research Team, “Operation DustySky Part 2 ClearSky Cybersecurity,” 2016, Accessed: Aug. 07, 2022. [Online]. Available: www.clearskysec.com/dustysky2

[59] Fidelis Cybersecurity Solutions, “njrat-uncovered.” https://app.box.com/s/vdg51zbfvap52w60zj0is3l1dmyya0n4 (accessed Aug. 07, 2022).

[60] Fidelis Cybersecurity Solutions, “njRAT-The-Saga-Continues.” https://github.com/jack8daniels2/threat-INTel/blob/master/2013/FTA-1010-njRAT-The-Saga-Continues.pdf (accessed Aug. 07, 2022).

[61] E. Yosef, “nJRAT Report: Bladabindi - Cynet.” https://www.cynet.com/attack-techniques-hands-on/njrat-report-bladabindi/ (accessed Aug. 07, 2022).

[62] G Data Blog, “The Uroburos case: new sophisticated RAT identified.” https://www.gdatasoftware.com/blog/2014/11/23937-the-uroburos-case-new-sophisticated-rat-identified (accessed Aug. 07, 2022).

[63] A. Allievi, “Snake Campaign: A few words about the Uroburos Rootkit.” https://blog.talosintelligence.com/2014/04/snake-campaign-few-words-about-uroburos.html (accessed Aug. 07, 2022).

[64] deresz and tecamac, “Uroburos: the snake rootkit,” 2014, Accessed: Aug. 07, 2022. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2014/08/20082358/uroburos.pdf