Article

A Framework for Developing Strategic Cyber Threat Intelligence from Advanced Persistent Threat Analysis Reports Using Graph-Based Algorithms

Burak Gulbay * and Mehmet Demirci

Posted Date: 17 July 2024

doi: 10.20944/preprints202407.1408.v1

Keywords: cyber threat intelligence; CTI; advanced persistent threat; APT; graph algorithms; cybersecurity

Article

A Framework for Developing Strategic Cyber Threat Intelligence from Advanced Persistent Threat Analysis Reports Using Graph-Based Algorithms

Burak Gulbay [1,*] and Mehmet Demirci [2]

[1] Department of Information Security Engineering, Graduate School of Natural and Applied Sciences, Gazi University, Ankara 06500, Turkey; burak.gulbay1@gazi.edu.tr
[2] Department of Computer Engineering, Faculty of Engineering, Gazi University, Ankara 06500, Turkey; mdemirci@gazi.edu.tr
* Correspondence: burak.gulbay1@gazi.edu.tr; Tel.:+90-553-599-7156

Abstract: Advanced persistent threat (APT) attacks are sophisticated and organized attacks commonly motivated by political, financial, and strategic objectives. In order to comprehend their tactics, techniques, and procedures (TTP) and indicators, APT reports are valuable sources. While blue teams typically rely on server logs, firewall rules and user authorizations managed in database tables, attackers have a graph-based mindset. In this work, we propose a framework for discovering and evaluating APTs using graph-based algorithms. Cyber threat intelligence (CTI) was extracted from 40,358 pages of APT reports and transformed into a graph. Centrality, community, and similarity analyses were executed on the graph. As a result, critical and influential APT groups and indicators of compromise (IoC) were discovered. Similar attacks and APT groups were revealed. Analysis results were interpreted to create new strategic CTI that can be utilized in future security operations.

Keywords: cyber threat intelligence; CTI; advanced persistent threat; APT; graph algorithms; cybersecurity

---

1. Introduction

Advanced persistent threats (APT) represent a higher threat level due to their sophistication, persistence, and targeted nature, distinguishing them from more common, less advanced cyber attacks [1]. APT is a cyber attack characterized by its prolonged duration and sophisticated tactics. The attackers aim to gain unauthorized access to a network and maintain their presence undetected for an extended period [2]. APTs are highly targeted, methodical, and complex, often designed to steal sensitive information, disrupt operations, or establish a foothold for future exploitation. The primary actors are state-sponsored actors, advanced cybercrime groups, hacktivists, and insider threats [3]. APTs differ from ordinary cyber attacks in several vital ways.

APTs utilize advanced techniques, including zero-day exploits, custom malware, and sophisticated social engineering tactics, which are often beyond the capabilities of ordinary cyber attackers [4]. Attackers continuously adapt their methods to evade detection and maintain access over weeks, months, or even years [5]. APTs are highly targeted, focusing on specific organizations, industries, and individuals with valuable information or strategic importance. In contrast, ordinary cyber attacks often involve indiscriminate targeting, such as broad phishing campaigns [6]. Ordinary cyber attackers usually operate with limited resources. The motivations behind APTs often include espionage, political gain, or strategic advantage, reflecting the interests of high-level criminal groups [7]. Ordinary cyber attacks are frequently motivated by immediate financial gain. APTs involve coordinated efforts by skilled teams, often employing multi-stage attack vectors and advanced operational security practices. Ordinary cyber attacks tend to be more straightforward, with less coordination and complexity.

Defending against APTs requires a multi-layered approach combining technology, processes, and people [8]. CTI integration is crucial to defend organizations against APTs. Consuming high quality CTI is an effective way of integration as incorporate CTI feeds that provide up-to-date information on emerging threats, IoCs, and TTPs used by APT actors [9]. Producing CTI by analyzing internal data to identify patterns and share insights with the broader security community to help improve overall defense mechanisms is another essential way to adapt CTI to defend against APTs [10].

CTI is studied under four categories in literature as technical, tactical, operational, and strategic [11]. Technical CTI provides particular details about threat actors, their tools, and methods. It focuses on the immediate IOCs that can be used to detect and respond to threats [12]. Tactical CTI focuses on the TTPs used by threat actors. It provides context about how attacks are executed and how different components of an attack fit together [13]. Operational CTI provides insights into specific, ongoing threats and campaigns. It focuses on understanding threat actors' motivations, capabilities, and activities in near real-time. Strategic CTI provides a high-level overview of the threat landscape, including broader trends, potential risks, and future threat scenarios. It focuses on long-term planning and decision-making [14].

Actionable CTI can be achieved by combining these four types of CTI and contextualizing them at a high level. As a result of digital forensics studies, significant knowledge has been gained about the APTs that have been encountered to date. This knowledge is available and embedded in APT analysis reports. There are many references in APT analysis reports pointing to IoCs like IP (Internet Protocol) addresses, domain names, subdomains, URLs (Uniform Resource Locator), file hashes, malwares, TTPs, etc.

Cyber security professionals on the defense side, also known as blue teams, think through tables. Monitoring tools, firewall rules, server logs, alerts, and other indicators are based on tables, whereas attackers think through a graph-based mindset. Therefore, managing and analyzing CTI data in a graph data structure allows us to comprehend how critical entities participate in advanced attacks [15,16].

The novelty of this study lies in examining the CTI extracted from APT reports, which ranged across many perspectives. Graph-based algorithms focusing on centrality, community detection, and similarity analysis were utilized to produce high-quality strategic CTI. This paper introduced a framework for discovering and evaluating insidious APTs using graph-based algorithms. Framework offers six main stages: collecting APT analysis reports, extracting knowledge, constructing CTI graphs, graph analysis, revealing, and interpreting analysis results. Consumers of newly generated strategic CTI are security operations center (SOC) analysts, purple and blue team specialists, and government-level decision-makers responsible for cyber defense.

The first stage handles collecting APT analysis reports from different sources. Duplicate reports are eliminated in this stage. Corrupted APT report files are removed from the collected report corpus. In the second stage, knowledge extraction is done. Triples consisting of entities and relationships are produced. Entities are categorized and related to corresponding APT reports. In the third stage, triples are saved in the graph. While saving the <entity, relationship, entity> triples, the uniqueness of nodes was guaranteed, and no duplicate nodes were saved. A graph with weighted nodes and weighted relationships was constructed. The more a node and relation were encountered, the higher weight is assigned. In the fourth stage, the resulting graph is analyzed using graph algorithms to detect the most central and bridging nodes on the graph and discover the most influential nodes. The last two stages are about gathering and interpreting numerical analysis results in the CTI context.

Significant challenges faced during the study are extracting useful knowledge from reports and maximizing the connectivity within the graph. A comprehensive literature review is performed to determine target node types. Answers have been found about the implications of the data obtained from APT reports when converted into nodes and relationships. Maximizing the number of relationships within the graph is important to contextualize the CTI. Common and recognizable node types with the potential for high levels of contextualization are determined from reports like malware family names, APT groups, countries, and TTPs. Another challenge was about interpreting analyses results. In order to generalize the information we obtained as the result of analyses and present it as strategic CTI, the result of the experiment should not be an exceptional situation. To ensure the general validity of the analysis results, the created graph is weighted on the basis of nodes and relationships.

The major contributions presented in this work involving different stages of the framework are listed below:

1. We proposed a system architecture to collect APT-related analysis reports from open-source intelligence (OSINT) sources. Synthetic datasets are not used.
2. We produced a weighted graph to strengthen the impact of the most encountered data. This helped us to produce strategic CTI.
3. We discovered the most influential IoCs and APT groups on the graph of CTI using centrality analysis. These nodes were also the most bridging nodes playing central roles in different attacks.
4. We detected communities of APT actors who seem to be acting together and have similar malicious activities using community detection techniques.
5. We exposed similar APT attacks with too much common technical and tactical CTI knowledge using graph-based similarity analysis techniques.
6. We created a systematic and empirical methodology to produce strategic CTI from technical and tactical CTI by processing APT analysis reports.

The rest of the paper is organized as follows. Section 2 discusses related work on extracting knowledge from APT reports and graph analysis algorithms. Section 3 presents details of the implementation of the proposed multi-stage framework. In Section 4, we provide an evaluation of the framework and experimental results. Finally, Section 5 presents conclusions and future research.

2. Background and Related Works

APTs represent a rough challenge in modern network security due to their stealthy behaviors [17]. These sophisticated cyber attacks often leverage zero-day exploits and threaten organizations since they can evade traditional detection techniques [18]. APTs typically involve long-term campaigns that aim to steal sensitive data, disrupt operations, conduct cyber espionage, or undermine critical infrastructure. In order to effectively combat these insidious threats, companies and governments must employ advanced detection, mitigation techniques, and CTI strategies.

In the fast-moving world of cyber security, graph-based algorithms have recently become more prominent in assessing APTs. Researchers proposed CSKG4APT and developed cyber security knowledge graphs using open-source CTI for improving threat intelligence extraction and attribution processes [19]. Due to their sophisticated and prolonged attacks, APTs require comprehensive evaluation methods that extend beyond traditional approaches. Many studies concentrate on how one can safeguard their systems from APTs by using block lists and CTI feeds through detection or prevention; however, this paper suggests interpreting previous high-quality attack analysis reports [20]. Graph-based algorithms offer a unique perspective to analyze relationships between diverse CTI entities. They enable a holistic understanding of APT behaviors and facilitate identifying critical nodes and influential actors within the graph of CTI. APTs have evolved to target cyber-physical systems (CPS), the Internet of Things (IoT), and industrial control systems. Integration of process-context information to graph analysis has emerged as a promising approach for APT detection. Li et al. proposed a framework called ConGraph, which combines graphs with many process-context information, such as file access activities and network interactions, enhancing the semantic richness of the graph [21]. Also, Wang et al. presented TBDetector, which leverages transformer-based analysis and anomaly scoring on graphs to detect slow-acting APT attacks [22]. Graph-based studies highlight the significance of historical information and contextual features in identifying abnormal activities of APTs.

Graph-based algorithms are vital in combating APTs by providing a powerful abstraction to understand complex attack scenarios. As evident from the research outlined by Zhang et al., using graph representations like attack graphs enables comprehensive analysis of TTPs belonging to APTs [23]. Graph-based algorithms facilitate the depiction of attacker plans and aid in allocating security resources effectively to fortify network defense. Xuan et al. emphasized in their study that the application of deep learning models, coupled with graph analysis techniques like bidirectional long short-term memory (BiLSTM) and graph convolutional networks (GCN), enhances the detection of

APT attacks through network traffic flow analysis [24]. By harnessing the capabilities of graph-based algorithms, security professionals gain critical insights, identify top influential entities, and enhance threat detection mechanisms. Graph algorithms can uncover hidden patterns, detect anomalies and communities, and identify critical nodes within the network. Key graph-based techniques include centrality measures, community detection, and node similarity analysis [25–27]. The graph algorithms applied in this study are divided into three main categories in following subsections.

2.1. Centrality Analysis

Centrality is a fundamental concept of analysis in graph theory. It is used to quantify the importance of a node within the graph. Centrality measures were used for identifying key IoCs and APT groups that play bridging roles in the graph of CTI about connectivity and information flow [28]. Centrality metrics focus on evaluating the relative importance of a node within the graph. The concept stems from the observation that not all nodes in a graph contribute equally to the graph's connectivity. Centrality algorithms assign scores to nodes based on their structural positions, relationship with neighbors, and roles within the graph [29]. Many centrality measures have been developed to capture node importance in different ways. Among the most commonly used are degree centrality (DC), betweenness centrality (BC), closeness centrality (CC), and eigenvector centrality (EC).

DC counts the number of direct connections of a node. In the case of an undirected graph, this is the number of edges connected to the vertex. In a directed graph, there are two measures: in-degree (number of incoming edges) and out-degree (number of outgoing edges) [30–32]. A high DC indicates that the node is highly connected within the graph. Nodes with high DC can be seen as local hubs or influencers within their close neighborhood [33]. This implies that APT groups and IoCs with high DC may lead us to find the origins of APT attacks.

BC measures the grade to which any node lies on the shortest paths between each node pair in a graph [34]. It captures nodes acting as a bridge within the graph. A node with high BC has significant control over the flow of information in a graph [35]. Such APT group and IoC nodes are crucial for maintaining connectivity and can influence communication between different parts of the graph of CTI.

CC measures how close a node is to all other nodes. CC is the opposite of the average shortest path length from the source node to all other nodes [36]. A node with high CC can quickly interact with all other nodes in the network. These nodes often have a high potential to be disseminators of malicious activities.

EC goes beyond counting direct connections, as in Degree Centrality, and instead measures the influence of a node within a network based on both the quantity and quality of its connections [37,38]. In essence, a node is considered important if it is connected to other important nodes. The connections to high-scoring nodes contribute more to the score of the node in question.

Centrality measures are vital tools in network analysis since they provide insights into the roles and significance of nodes within various graphs (directed, undirected, weighted, unweighted, etc.). DC highlights the most connected nodes, BC identifies key intermediaries, CC finds nodes that can efficiently interact with the entire network, and EC identifies influential nodes by accounting for their neighbors' influence [39]. We gained a deeper understanding of unknown interactions between APTs by applying these centrality measures. Identifying critical nodes helped us to develop strategic CTI and enhance countermeasures.

2.2. Community Detection Analysis

In our study, community detection (CD) helps us find APT actors acting together. Also, it is possible to determine sets of IoCs that tend to be involved in different APT attacks together. CD is a concept in graph theory that aims to identify communities of nodes that are more densely connected internally than the rest of the graph [40]. The detected communities represent functional groups with common behaviors. Studying CD is essential for understanding the organization of complex graphs, revealing hidden patterns and functional groups [41]. There are several algorithms to detect communities in graphs; each has its criteria for defining communities. K-core decomposition (KCD), label propagation (LP), and the Leiden algorithms are used in our study.

KCD is an algorithm that recursively removes nodes from the graph with fewer connections than k, where k is a predefined integer. The resulting subgraph with all remaining nodes having at least k connections is called the k-core [42]. KCD has three essential steps: initialization, node removal, and iteration. The initialization step means starting with the original graph. The node removal process involves removing nodes with a degree less than k. The iteration step means repeating the node removal process until no nodes with connections fewer than k remain. At the and, nodes in the k-core are more closely connected than the rest of the graph. Higher values of k reveal more tightly united communities [43].

LP is an algorithm that assigns labels to nodes and iteratively updates the labels based on the majority label of their neighbors. Over time, nodes within the same community will converge to the one dominant label [44,45]. LP has three essential steps: initialization, propagation, and convergence. The initialization step states assigning a unique label to each node. Propagation means updating the label of each node to the most frequently encountered label among its neighbors. Finally, the convergence step repeats the propagation step until the labels stabilize and no more changes occur. Resulting nodes with the same label form a community.

The Leiden algorithm is an advanced community detection method. It is an improved version of the Louvain algorithm [46]. It aims to optimize the modularity of the partition and ensure better quality for community detection. Leiden algorithm has three main steps: local moving of nodes, refinement, and aggregation. In the first step, nodes are moved between communities to optimize modularity locally. Then, nodes within the same community are refined to improve the partition quality. In the last step, communities are aggregated into super-nodes. These three steps are repeated on the newly reduced graph [47]. The Leiden algorithm provides an accurate and stable community detection by addressing frequently encountered issues such as disconnected communities. This phenomenon tends to occur with the Louvain algorithm [48].

2.3. Similarity Analysis

Similarity analysis is another pivotal area of study in graph theory. It provides insight into the relationships and entities within complex networks. In the literature, various methodologies and applications of similarity analysis in graphs exist. Common similarity measures were encountered, such as Jaccard Similarity, Cosine Similarity, Adamic-Adar Similarity, and Weighted Jaccard Similarity.

Jaccard Similarity examines similarity based on two sets by comparing the size of their intersection against their union [49,50]. It is mainly used for information retrieval from graphs and social network analysis [51–53]. Cosine similarity measures the cosine angle of the angle between two vectors, implicating node embeddings of nodes in this case, with a focus on orientation rather than magnitude. Cosine Similarity performs well with high-dimensional data [54,55]. Adamic-Adar Similarity is believed to contribute to less-connected nodes with a weighted common neighbors measure [56,57]. Weighted Jaccard Similarity uses edge weights to extend the Jaccard index for a more finely-grained similarity measure [58].

Node similarity algorithms help correlate threat data from diverse sources, identify related IoCs, and uncover hidden connections between different cyber threats [59]. Our study used similarity algorithms to detect similar APT groups and similar attacks.

3. Proposed Methodology

The proposed framework has six stages, starting from real-world data collection to computing the results of graph-based analysis as follows:

- Data Collection Stage: A widely inclusive APT report corpus is built using open-source repositories.

- Knowledge Extraction Stage: APT analysis reports are scanned to recognize IoCs and other contextual entities like malwares, APT groups, and countries.
- Graph Generation Stage: Using previously extracted CTI knowledge, a graph of nodes and edges is formed.
- Graph Analysis Stage: The most influential APT groups and IoCs are detected. Results are interpreted in the context of APT attacks.
- Gathering Analysis Results: Numerical analysis results are captured in this stage. Outputs of applied algorithms needed to be interpreted.
- Interpreting Analysis Results: Outputs of applied algorithms were interpreted in this stage. Generated scores at the previous stage needed more contextual explanation. The most remarkable output of the stage was strategic CTI generated for long-term usage.

High-level architecture of the framework indicating process flow between stages is presented in Figure 1. Our methodology is based on the theoretical principles of graph theory, and the proof of concept is studied empirically. The entire process covering the framework was implemented with a series of programming languages and techniques. The results of future described analyses in the study were presented numerically. The strategic CTI was produced by interpreting the results of the analysis.

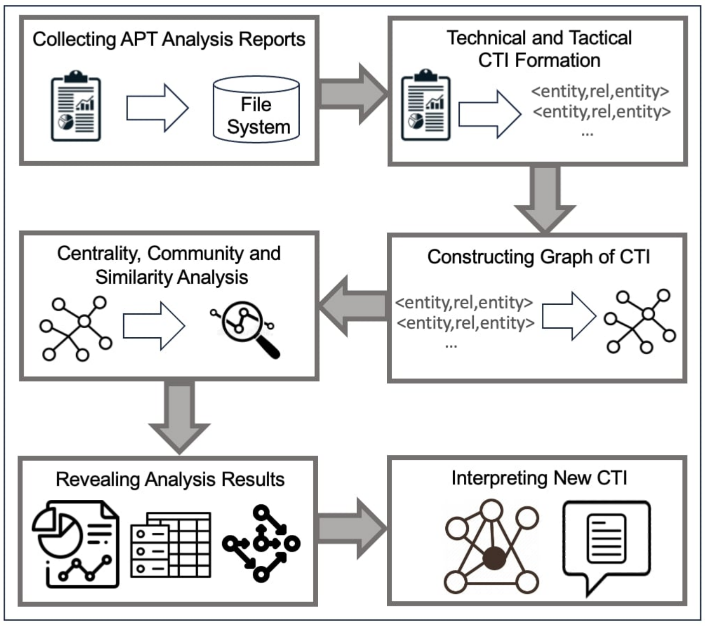

Collecting APT Analysis Reports
File System
Technical and Tactical CTI Formation
<entity,rel,entity>
<entity,rel,entity>
...
Constructing Graph of CTI
<entity,rel,entity>
<entity,rel,entity>
...
Centrality, Community and Similarity Analysis
Revealing Analysis Results
Interpreting New CTI

Figure 1. The architecture of the proposed framework indicating process flow between stages.

3.1. *Collecting APT Analysis Reports*

APT analysis reports provide knowledge on APTs within the CTI domain for a few reasons. The reports provide in-depth insights into the TTPs used by sophisticated threat actors, thereby assessing and combating complex cyber threats. Typically, APT reports would describe the methods and strategies of threat actors in detail, which could be mapped against frameworks such as MITRE ATT&CK. The reports provide an understanding of the behavior and the capabilities of the adversaries. Specifically, it is intended to help organizations in threat detection and mitigation by providing particular data points that include, but are not limited to, IP addresses, domain names, file hashes, and malware signatures. Reports provide the capability for understanding the motivations, origins, history, and activities of threat groups, which enable organizations to understand the potential risks and impact of an attack.

There are two APT report repositories with a high reputation on GitHub maintained by the cyber security community. They are available on github.com/aptnotes/data and github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections. APT reports were collected from these sources in PDF file format. After removing corrupted files and copies, 1980 APT analysis reports were gathered. Collected reports count per year is presented in Figure 2

Interpreting Figure 2 is valuable since it tells us a reality about APT attacks. It takes an average of 10 days for an APT attack to be noticed by organizations in 2023. It is called dwell time, the period between infiltration and detection [60]. According to Figure 2, there has been a decrease in publicly available APT reports count in the last few years. The main reason is that current APT attacks all around the globe are still unperceived. Understandably, there is a time gap between the detection of attacks and the public release of APT reports.

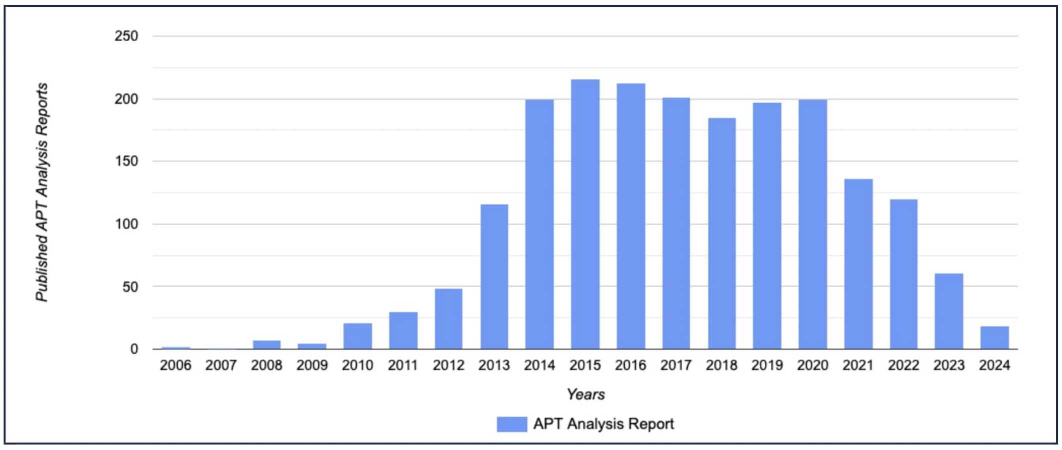


Figure 2. Published APT reports per year.

Dwell time change for APT detection from 2011 to 2023 is presented in Table 1. CTI studies and threat hunting techniques have significantly decreased dwell time. Another critical trend about dwell time shows us that internal detection is still the most effective approach. With this motivation, APT reports were collected, deduplicated, and saved to the file system by the end of the first stage.

Table 1. Dwell time for APT attacks between 2011-2023.

Detection Source	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
External Notification	-	-	-	-	320	107	186	184	141	73	28	19	13
Internal Detection	-	-	-	-	56	80	57.5	50.5	30	12	18	13	9
All	416	243	229	205	146	99	101	78	56	24	21	16	10

3.2. Technical and Tactical CTI Formation

Technical CTI refers to the specific and actionable knowledge that can be directly utilized to detect threats. Technical CTI is highly detailed and granular, focusing on the immediate indicators of malicious activities and artifacts observed in networks or systems [61]. IoCs are critical elements within technical CTI. They include specific data such as IP addresses, domain names, subdomains, URLs, email addresses, and file hashes (e.g., MD5, SHA-1, SHA-256). Malware signatures are fingerprints used to identify known malwares. Signature-based detection is a fundamental technique in anti-malware solutions, leveraging predefined patterns to detect malicious software [62].

Tactical CTI focuses on the TTPs followed by threat actors. It provides a broader understanding of how adversaries operate, allowing organizations to anticipate and defend against sophisticated threats [63]. Tactical CTI is less granular than technical CTI but offers more profound insights into attackers' methodologies. Tactics are threat actors' overarching strategies and goals, such as espionage, financial gain, or disruption. Techniques are specific methods to achieve their goals, such as phishing, exploiting vulnerabilities, or leveraging zero-day exploits [64]. Techniques are often documented in frameworks like MITRE ATT&CK [65]. Procedures are detailed sequences of actions threat actors take to execute their techniques. Procedures can include step-by-step methods to deploy malware, establish command and control (C2) infrastructure, and exfiltrate data.

In our study, we used the IoC Searcher proposed by Caballero et al [66] to extract CTI entities. IoC Searcher finds IoCs by regular expression (RegEx) matching. In case of a match, apt_report was related with one of the node types presented in Table 2. We also added four additional steps to detect APT groups, countries, malwares and MITRE techniques. A list with 1656 unique entries of all known APT groups is gathered from trusted CTI providers like Crowd Strike, Symantec, Fire Eye, Kaspersky, etc [67]. A list with 196 unique entries from all countries is created. A list with 3048 unique entries of known malware families is gathered from Malpedia [68]. Another list with 358 unique entries of adversarial techniques is gathered from MITRE [69]. APT reports were scanned and in case of a match, apt_report was related to one of the following node types: apt_group, country, malware, and mitre_technique. The technical and tactical CTI formation process is presented in Figure 3. All CTI knowledge extracted from APT reports is provided at [86] as separate files for each report.

Table 2. Occurrence counts of node labels in graph.

Node Label	**Count**
apt_group	785
apt_report	1980
bitcoin	5
bitcoincash	1
copyright	384
country	133
cve	573
dashcoin	1
domain	15901
email	1480
ethereum	7
facebook_handle	61
github_handle	246
ip	9985
ip4net	365
linkedin_handle	19
litecoin	1
malware	882
md5	17893
mitre_technique	181
monero	1
onion_address	14
package_name	3
phone	197
pinterest_handle	1
ripple	1
sha1	7582
sha256	12755

Table 2. *Cont.*

Node Label	Count
subdomain	13190
telegram_handle	2
trademark	211
twitter_handle	138
url	19762
uuid	482
youtube_channel	3
youtube_handle	8
zcash	1

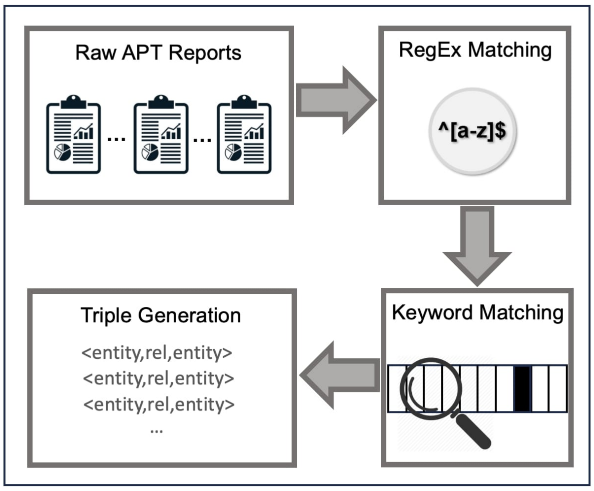


Figure 3. Technical and tactical CTI knowledge extraction.

3.3. Constructing Graph of CTI

As of the beginning of this stage, technical and tactical CTI have been extracted from APT reports, labeled, and transformed into triples. Throughout this stage, triples were converted into nodes and relationships and integrated into a graph. If recurrent nodes were encountered, they were deduplicated. A total of 37 types of nodes were observed on the created graph. The occurrence count of each node type in the graph is presented in Table 2. The count of total nodes is 105,232, and the total relationship is 156,911.

While building the graph, a weight score is assigned to nodes and relationships. The weight refers to the number of times a unique node and relationship is encountered in APT reports. Categorized node labels and values were saved as a dictionary in a raw text file by the end of the last subsection. Each file has the name of the corresponding APT report. In this stage, we parsed files using Golang, converted them into the graph data structure, and saved them to the Neo4j database. The source code for this transformation and resulting graph database were provided at [86].

3.4. Centrality, Community and Similarity Analysis on Graph of CTI

Graph analysis using centrality, community, and similarity algorithms is valuable in CTI because it unleashes the potential of graphs. Community detection can reveal clusters of related IoCs and threat actors, indicating coordinated campaigns or attack vectors. Identifying threat actor communities guides us to understand the motivation behind their acting together. Centrality analysis led us to focus on the impact of an attack by setting light on the path from key nodes to victims.

3.4.1. Centrality Analysis of APT Group and IoC Nodes

Centrality in graph theory is used to identify the most important nodes within a network. Centrality aims to determine which nodes hold the most influence among others. Our study uses DC, BC, CC, and EC to gain a comprehensive perspective. Each centrality provided different insights into the structure and dynamics of the CTI network.

DC measures the number of direct connections a node has. Using DC, we identified highly connected nodes that can quickly spread threats within APT attacks. An IoC or APT Group with high DC is highly influential due to many direct contacts. This also indicates that the IoCs and APT groups were observed in multiple attack reports. The DC of a vertex v, for a given graph $G := (V, E)$ with $|V|$ vertices and $|E|$ edges, is defined as $C_D(v) = \deg(v)$.

Calculating DC for each node in the graph takes $\Theta(V^2)$ in a dense adjacency matrix representation of the graph, and for edges, takes $\Theta(E)$ in a sparse matrix representation [70]. The concept of centrality at the individual node level can be broadened to encompass the entire graph, resulting in what is known as graph centralization. Let $v*$ be the node with the highest DC in G. Let $X := (Y, Z)$ be the Y-node connected graph that maximizes the following quantity H where $y*$ is the node with highest DC in X [71]. H is calculated using Equation (1).

$$H = \sum_{j=1}^{|Y|} [C_D(y*) - C_D(y_j)] \tag{1}$$

The DC of the graph G is calculated in Equation (2):

$$C_D(G) = \frac{\sum_{i=1}^{|V|} [C_D(v*) - C_D(v_i)]}{H} \tag{2}$$

DC measurements for the graph of CTI is provided in Table 3. Also APT group nodes with the highest DC are presented in Table 4.

Table 3. Degree centrality measure statistics observed on graph of CTI.

Minimum Score	Maximum Score	Mean Score
0.0000	1865.0078	2.9822

Table 4. Top 10 APT group nodes with highest DC.

APT Group	Degree Centrality
lazarus group	61
turla	56
sofacy	45
apt28	40
shamoon	38
carbanak	34
muddywater	33
hacking team	31
oilrig	31
sandworm	27

The BC measures the extent to which a node lies on the shortest paths between other nodes in graph [72]. It was used to identify nodes that act as bridges within the network, controlling attack flow.

An APT group with high BC is crucial for coordinating malicious activities. The BC of a vertex v in the graph $G := (V, E)$, where V stands for vertices and E stands for edges, is computed as described in following steps:

1. s stands for source vertex, t stands for target vertex and for each (s, t) pairs , compute the shortest paths σ_{st} between s and t.
2. For each (s, t) pairs, calculate the fraction of shortest paths that pass through the current vertex v.
3. Sum the calculated fraction for all (s, t) pairs.

The BC is calculated using following Equation (3) [73]:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (3)$$

Considering the above formula σ_{st} indicated the total count of shortest paths from s to t and $\sigma_{st}(v)$ is the count of the paths those pass through the current vertex v. Calculated BC measurements for the graph of CTI are provided in Table 6. Also, APT group nodes with the highest BC are presented in Table 5

Table 5. Top 10 APT group nodes with highest BC.

APT Group	**Betweenness Centrality**
lazarus group	17064084.27
apt28	6594142.41
turla	6393610.82
hacking team	5876781.64
sofacy	5218912.24
patchwork	5018645.94
oceanlotus	3999530.36
kimsuky	3945141.78
charming kitten	3878083.47
carbanak	3174435.89

Table 6. Betweenness centrality measure statistics observed on graph of CTI.

Minimum Score	**Maximum Score**	**Mean Score**
0.00	1186373632.00	202920.06

The CC measures how close a vertex is to all other vertices in the graph [74]. It was used to identify nodes that can quickly interact with other nodes. Finding nodes with high CC guided us in understanding which nodes in the graph can facilitate the swift spread of malicious activities. The CC of vertex v is calculated as follows:

$$C_C(v) = \frac{N - 1}{\sum_s d(s, t)} \quad (4)$$

Considering the Equation (4), s stands for source node, t stands for target node and $d(s, t)$ is the distance between nodes s and t. N is the count of nodes in graph G. CC measurements for the graph of CTI are provided in Table 7. Also, APT group nodes with the highest CC are presented in Table 8.

Table 7. Closeness centrality measure statistics observed on graph of CTI.

Minimum Score	Maximum Score	Mean Score
0.0000	1.0000	0.2080

Table 8. Top 10 APT group nodes with highest CC.

APT Group	Closeness Centrality
lazarus group	0.2564
apt28	0.2542
winnti group	0.2542
snake	0.2538
kimsuky	0.2536
hacking team	0.2532
turla	0.2529
carbanak	0.2528
fin7	0.2527
shamoon	0.2525

The EC measures the influence of a node based on the number and quality of its connections. It helped pinpoint critical infrastructure nodes that are highly influential within the graph. The nodes with high EC are compromised; this may significantly impact the entire graph [75]. $G := (V, E)$ is a graph with V vertices and E edges, define $A = (a_{v,t})$ be the adjacency matrix where $a_{v,t} = 1$ if vertex v is linked to target t, else the adjacency matrix becomes $a_{v,t} = 0$. In Equation (5), x represents the eigenvector, and the relative centrality score of v is calculated as follows:

$$x_v = \frac{1}{\lambda} \sum_{t \in N(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t \tag{5}$$

Considering the Equation (8), $N(v)$ is set of the neighbors of v, λ is a constant value and the eigenvector equation is $Ax = \lambda x$. There exist various eigenvalues λ for a non-zero eigenvector solution. There is a unique and the largest eigenvalue, which results in the desired centrality measure [76]. EC measurements for the graph of CTI are provided in Table 10. Also, APT group nodes with the highest EC are presented in Table 9. Detailed evaluation of centrality analyses is provided in subsection 4.1.

Table 9. Top 10 APT group nodes with highest EC.

APT Group	Eigenvector Centrality
lazarus group	0.0189
turla	0.0178
apt28	0.0177
sofacy	0.0172
shamoon	0.0171
snake	0.0166
apt10	0.0165
hacking team	0.0160
winnti group	0.0160
naikon	0.0158

Table 10. Eigenvector centrality measure statistics observed on graph of CTI.

Minimum Score	Maximum Scorre	Mean Score
1.521e-12	0.3422	0.0005

3.4.2. Community Detection on Graph of CTI

The community detection algorithms were utilized to assess the clustering of nodes. The algorithms assisted in measuring the tendency of nodes to unite or separate. Our study utilizes KCD, LP, and Leiden algorithms to identify communities on the graph of CTI.

The KCD handles grouping the nodes of the graph based on their degree sequence. m-core refers to a maximal subgraph of the initial graph where each node in this subgraph has a degree of at least m. The maximality property guarantees that no other subgraph with a greater number of nodes satisfies this degree condition. All nodes in the subgraph stated as m-core also belong to the subgraph specified by n-core for any $n<m$. Each node v is related to a core value that states the greatest value m where v belongs to the m-core [77]. The KCD algorithm eliminates the nodes with the smallest degree until the initial graph gets emptied. Eliminating a node from the graph results in removing all its connections, which leads to a decrease in the degree of its neighbor nodes by one. With this method, the various core groups were identified sequentially. The algorithm assisted in emphasizing the more tightly linked subgraphs of the graph, also known as cores [78]. KCD steps were presented as Algorithm 1:

Algorithm 1 Pseudo-code implementation of k-core decomposition. The input to the algorithm is a graph G consisting of a set of nodes V and a set of edges E. The output of the algorithm is core number for each node in V.

```
Require: Graph $G(V, E)$                              ▷ Input: graph with nodes $V$ and edges $E$
Ensure: Core number for each node in $V$
 1: $n \leftarrow$ number of vertices in $G$
 2: Initialize $degree[v]$ for all $v \in V$
 3: Initialize $K[v] \leftarrow 0$ for all $v \in V$
 4: Initialize $visited[v] \leftarrow$ false for all $v \in V$
 5: $min\_heap \leftarrow$ empty min-heap (priority queue)
 6: for each vertex $v \in V$ do
 7:     $min\_heap$.insert$(v, degree[v])$
 8: end for
 9: while not $min\_heap$.is_empty() do
10:     $(current\_deg, u) \leftarrow min\_heap$.extract_min()
11:     $K[u] \leftarrow current\_deg$
12:     $visited[u] \leftarrow$ true
13:     for each neighbor $w \in$ neighbors$(u)$ do
14:         if not $visited[w]$ then
15:             $degree[w] \leftarrow degree[w] - 1$
16:             $min\_heap$.decrease_key$(w, degree[w])$
17:         end if
18:     end for
19: end while
20: return $K$                                        ▷ Output: Core number for each node
```

We applied KCD to split the graph of CTI into subgraphs, determined tightly connected noted and obtained 15 communities. The greatest core value was calculated as 15 and 785 APT groups split into communities accordingly. Counts of APT groups acting together were presented in Table 11.

Table 11. Top 5 the most populous APT group communities acting together detected by k-core decomposition.

Community	APT Group Count
C1	195
C2	166
C3	106
C4	82
C5	71

The LP is a community detection algorithm that assigns the nodes to communities by considering their neighbors' labels. Initially, each node has a distinct label that spreads throughout the graph in a series of iterations. The algorithm works by nodes adopting the most common label among their neighbors, ultimately clustering nodes into communities [79]. The steps of LP were outlined as Algorithm 2:

Algorithm 2 Pseudo-code implementation of Label Propagation algorithm. The input to the algorithm is a graph G consisting of a set of nodes V and a set of edges E. The output of the algorithm is final community labels for each node in V.

```
Require: Graph $G(V, E)$                              ▷ Input: graph with nodes $V$ and edges $E$
Ensure: Community labels for each node in $V$
 1: Initialize $label[v] \leftarrow v$ for all $v \in V$
 2: Initialize $changed \leftarrow$ true
 3: while $changed$ do
 4:     $changed \leftarrow$ false
 5:     for each node $v \in V$ in random order do
 6:         $label\_count \leftarrow$ empty dictionary
 7:         for each neighbor $u \in$ neighbors$(v)$ do
 8:             $label\_count[label[u]] \leftarrow label\_count.\text{get}(label[u], 0) + 1$
 9:         end for
10:         $most\_frequent\_label \leftarrow$ label with the highest count in $label\_count$
11:         if $label[v] \neq most\_frequent\_label$ then
12:             $label[v] \leftarrow most\_frequent\_label$
13:             $changed \leftarrow$ true
14:         end if
15:     end for
16: end while
17: return $label$                                     ▷ Output: Final community labels for each node
```

As another option to discover APT group communities, we applied LP to split the graph into clusters. Execution of the LP resulted in 3 communities. The algorithm ran with five iterations, and by the end of the last iteration, the algorithm reached a stable state where node labels no longer changed. Counts of APT groups who tend to act together were presented in Table 12.

Table 12. Top 3 the most populous APT group communities acting together detected by Label Propagation.

Community	APT Group Count
C1	732
C2	52
C3	1

The Leiden algorithm was employed as the last algorithm to detect communities. It is an algorithm for detecting communities in large graphs. The algorithm divides nodes into separate communities to maximize a modularity score within each community [80]. The modularity score measures how connected nodes are within a community relative to the connections they would have in a random graph. The algorithm is hierarchical, merging communities into single nodes by optimizing modularity and repeating in the condensed graph. The procedure utilized in the Leiden algorithm consists of the following steps in Algorithm 3.

By using the Leiden algorithm, the graph was split into 87 communities. Counts of APT groups involved in different communities served in Table 13. Detailed evaluation of community detection analyses is provided in subsection 4.1.

Algorithm 3 Pseudo-code implementation of the Leiden algorithm. The input to the algorithm is a graph G consisting of a set of nodes V and a set of edges E. The output of the algorithm is the final community labels for each node in V.

```
Require: Graph $G(V,E)$                                  ▷ Input: graph with nodes $V$ and edges $E$
Ensure: Community labels for each node in $V$
 1: Initialize $label[v] \leftarrow v$ for all $v \in V$
 2: Initialize $improved \leftarrow$ true
 3: while $improved$ do
 4:     $improved \leftarrow$ false
 5:                                                       ▷ Phase 1: Local Moving of Nodes
 6:     $changed \leftarrow$ true
 7:     while $changed$ do
 8:         $changed \leftarrow$ false
 9:         for each node $v \in V$ in random order do
10:             $label\_count \leftarrow$ empty dictionary
11:             for each neighbor $u \in$ neighbors$(v)$ do
12:                 $label\_count[label[u]] \leftarrow label\_count.$get$(label[u], 0) + 1$
13:             end for
14:             $most\_frequent\_label \leftarrow$ label with the highest count in $label\_count$
15:             if $label[v] \neq most\_frequent\_label$ then
16:                 $label[v] \leftarrow most\_frequent\_label$
17:                 $changed \leftarrow$ true
18:             end if
19:         end for
20:     end while
21:                                                       ▷ Phase 2: Refinement
22:     for each community $C$ identified in Phase 1 do
23:         $sub\_communities \leftarrow$ split community $C$ into sub-communities
24:         for each sub-community $S \in sub\_communities$ do
25:             Move nodes between $S$ and other sub-communities to improve quality function
26:         end for
27:     end for
28:                                                       ▷ Phase 3: Aggregation
29:     Aggregate the graph using the refined communities to create a new graph $G'$
30:     Re-initialize $V$ and $E$ with the nodes and edges of $G'$
31:     if the structure of $G'$ is different from $G$ then
32:         $improved \leftarrow$ true
33:     end if
34: end while
35: return $label$                                       ▷ Output: Final community labels for each node
```

Table 13. Top 5 the most populous APT group communities acting together detected by Leiden algorithm.

Community	APT Group Count
C1	226
C2	83
C3	50
C4	49
C5	44

3.4.3. Similarity Analysis of APT Attacks

Similarity algorithms compute the similarity of pairs of nodes based on their neighborhoods or properties. Several similarity metrics can be used to compute the similarity score. Our study uses weighted Jaccard similarity to detect similarities between APT groups. Clustering similar IoCs helped us reveal new relationships and correlations between threat actors and their tactics. Similarity analysis enables proactive threat hunting by identifying new IoCs similar to known malicious indicators. This allows discovery of threats that have not yet been observed.

The weighted Jaccard similarity algorithm examines a group of nodes by looking at their shared connections [81]. Two nodes are deemed similar when they have many shared neighbors. Relationship characteristics were utilized to adjust the similarity influenced by specific relationships by considering their value to assess significance. s and t are two vectors, d is equal to the size of the union of the two sets, and weighted Jaccard similarity is formulated in Equation (10) [82].

$$J_W(s,t) = \frac{\sum_{i=1}^{d} min(s_i, t_i)}{\sum_{i=1}^{d} max(s_i, t_i)} \tag{6}$$

In Equation (10) using two nodes and their sets of weighted neighbors $N_w(s)$ and $N_w(t)$, we combined the sets into s and t, then iterated over the combined set of neighbors $N_w(s) \cup N_w(t)$ and assigned a weight of 0 to any non-neighbors. Typically, security analysts focus on tracing the TTPs of attackers while investigating APT attacks. However, both technical and tactical CTI play a role in developing countermeasures for threat actors. We applied a weighted Jaccard similarity measure to compare reports of APT attack nodes on the graph by considering technical and tactical CTI. Experiment results led us to the most similar APT attack reports in Table 14. Detailed evaluation of similarity analysis is provided in subsection 4.1.

Table 14. Top 10 the most similar APT reports identified by weighted Jaccard similarity.

APT Report	APT Report	Similarity
ThaiCERT-A_Threat_Actor_Encyclopedia(06-19-2019).pdf	Threat Group Cards.pdf	0.9958
BlackEnergy2_Plugins_Router.pdf	be2-custom-plugins-router-abuse-and-target-profiles.pdf	0.9804
iranian-threat-group-updates-ttps-in-spear-phishing-campaign.pdf	MuddyWaters Recent Activity.pdf	0.9215
CERTFR-2019-CTI-005.pdf	cta-2019-0206.pdf	0.4516
Donot Team in South Asia.pdf	Musical Chairs Playing Tetris.pdf	0.3649
Latest Trickbot Campaign Delivered via Highly Obfuscated-JS File.pdf	Obfuscation Tools Found in the Capesand Exploit Kit Possibly Used in KurdishCoder Campaign.pdf	0.3548
More than a Dozen Obfuscated APT33 Botnets Used for Extreme Narrow Targeting.pdf	wp-drilling-deep-a-look-at-cyberattacks-on-the-oil-and-gas-industry.pdf	0.3441
Elfin Relentless Espionage Group Targets Multiple Organizations in Saudi Arabia and US.pdf	cta-2019-0626.pdf	0.3355
Deep in Thought_ Chinese Targeting of National Security Think Tanks.pdf	Mo Shells Mo Problems - Web Server Log Analysis.pdf	0.3333
Latest Trickbot Campaign Delivered via Highly Obfuscated JS File .pdf	URSNIF EMOTET DRIDEX and BitPaymer Gangs Linked by a Similar Loader.pdf	0.3333

4. Evaluation of Proposed Framework and Experiments

The framework contains the execution of various algorithms for different purposes. We provided an interpretation of the experiment results for each contribution. In this section, we evaluated the entire proposed framework. The data set was collected from open sources. The use of synthetic data was avoided to reveal the study's success. The codebase used throughout the process was implemented in Python, Golang, and Cypher languages and executed on a Macbook Pro with an M2 Pro CPU, 32 GB of memory, and 512 GB of SSD. We recommend better hardware configuration with high-end CPUs and memory to decrease execution time. Running graph algorithms on large dataset was a time consuming task.

Knowledge extraction from APT reports was successfully fulfilled. A total of 1980 APT reports with 40,358 pages were transformed into a weighted graph structure by applying knowledge extraction operations. The graph with 105,232 vertex and 156,911 edges was constructed. The top 10 weighted node labels and values were presented in Table 15.

Table 15. Top 10 node labels and names with the highest weight.

Node Label	Node Name	Weight
malware	trojan	414
malware	rat	324
domain	github.com	233
country	united states	183
domain	gmail.com	150
country	ukraine	135
apt_group	turla	112
cve	CVE-2012-0158	106
malware	mimikatz	106
mitre_technique	t1059-Command and Scripting Interpreter	90

4.1. *Examining Analyses Results in the CTI Context*

Finding the most influential IoCs and APT groups was one of the main goals of our study. We have identified the key nodes using centrality algorithms. Four different IoC sets consisting of nodes with high centrality measures were intersected. Each set had top-k nodes of DC, BC, CC, and EC. The intersection of nodes with top-k centrality measurements on the graph of CTI is presented in Figure 4.

Lazarus Groups, Turla, APT28, and Hacking Team nodes were observed as the most potent APT groups on the graph of CTI. In order to interpret the centrality analysis, we calculated the number of APT reports that were reached from these APT groups over n-hops as indicated in Figure 5 where the horizontal axis is several hops and the vertical axis is several APT reports. Another outcome of centrality analysis showed us the change in the number of attacks reachable by APT groups and dropped off by the sixth hop. The APT groups rapidly maximized their impact area after the first hop. We investigated the top 5 malicious IoCs involved in separate attacks. The nodes with the highest centrality scores belonging to DC, BC, CC, and EC analysis results are demonstrated in Table 16.

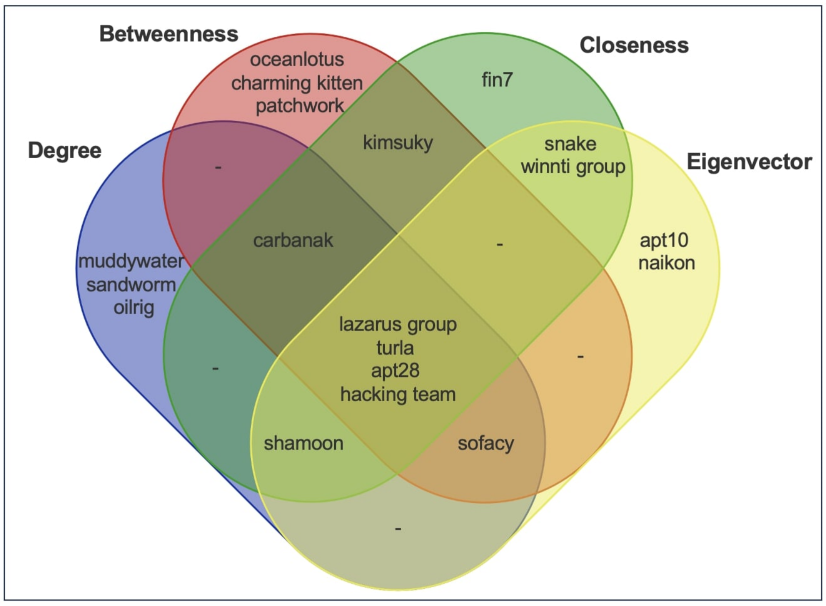


Figure 4. Intersection of nodes with top-k centrality measurements on graph of CTI.

Table 16. Centrality scores of top 5 malicious IoCs.

Centrality	Node Label	IoC	Measurement
Degree Centrality	malware	mimikatz	106.00
	cve	CVE-2012-0158	106.00
	country	iran	101.00
	malware	plugx	95.00
	malware	mimic	85.00
Betweenness Centrality	cve	CVE-2012-0158	56628354.08
	malware	Mimic Ransomware	31474216.60
	cve	CVE-2017-11882	29433514.48
	malware	mimikatz	29010596.68
	malware	plugx	26132601.01
Closeness Centrality	sha256	92d057720eab41e9c6bb684e834da632ff3d79 b1d42e0 27e761d21967291ca50	0.66
	sha256	d48bcca19522af9e11d5ce8890fe0b8daa01f93 c95e6a 338528892e152a4f63c	0.66
	uuid	BDE316E7-2665-4511-A4C4-8D4D0B7A9EAC	0.66
	domain	tawaranmurah.com	0.52
	ip	81.4.100.197	0.52
Eigenvector Centrality	counrty	iran	0.0316
	country	japan	0.0307
	cve	CVE-2012-0158	0.0265
	malware	mimikatz	0.0264
	malware	rogue	0.0252

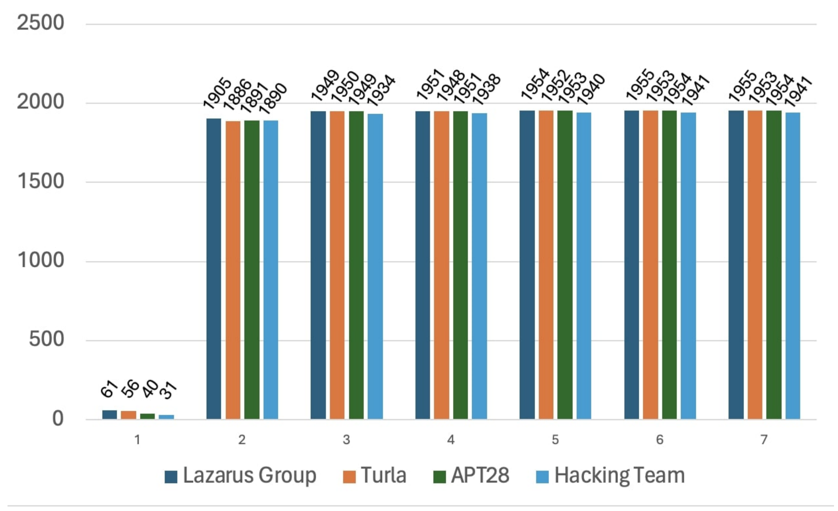


Figure 5. Reachable reports over n-hops by APT groups with top 5 centrality score.

Another contribution of our study is discovering communities of threat actors who are acting together. Running community analysis on APT groups also gave us insight into finding alias names of the same APT groups. We applied KCD, LP, and Leiden algorithms and acquired the communities with following populations: 16, 78, and 158. Discovered communities with their member nodes counts were presented in Figure 6, where the horizontal axis represents communities, and the vertical axis represents node counts. According to community detection experiments, three of the algorithms more or less agreed that there are five relatively large communities. The remaining communities are small compared to the others. APT groups with high Jaccard similaries formed distinct communities are provided below. They seem acting together or are aliases of each other.

- gold lowell, topgear
- unc2643, iamtheking
- hurricane panda, playful dragon, nightshade panda, anchor panda, tg-6952
- shell crew, brownfox, group 35, group 13, potassium, pinkpanther, shanghai group, kungfu kittens
- menupass team, cvnx, red apollo

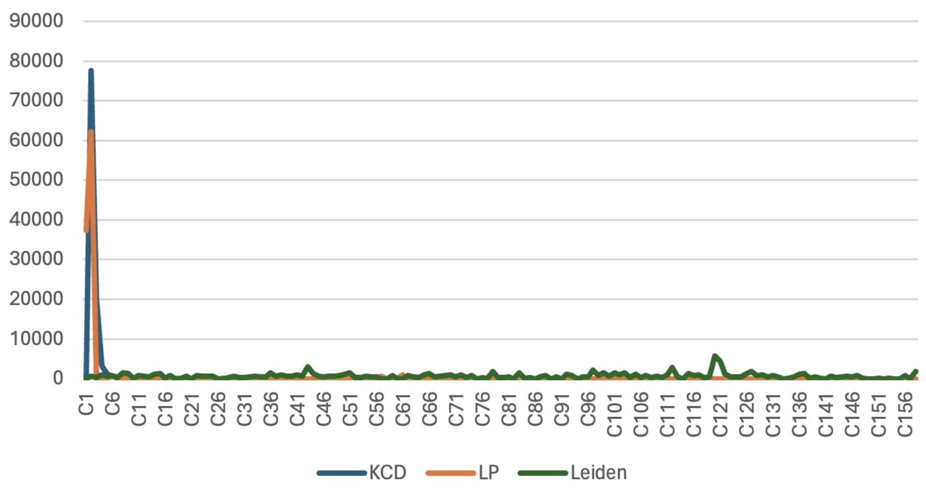
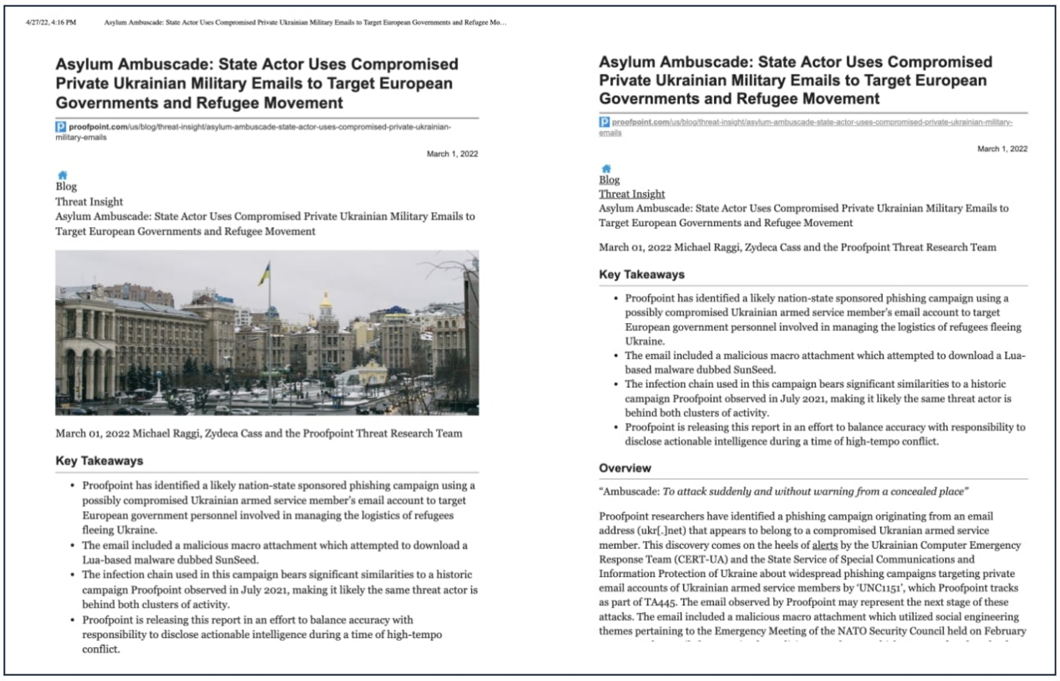


Figure 6. IoC communities and member nodes discovered by KCD, LP and Leiden algorithms.

Weighted Jaccard similarity performed well and allowed us to find similar reports regarding technical and tactical CTI. We noticed many reports that reported the same malicious activities but with slightly different layouts. We thought some reports were published more than once on different websites. Even though the content was nearly the same, the authors made some visual changes. For example, the first pages of two different reports are presented in Figure 7 side by side. Both reports were about investigating an attack targeting European governments over compromised Ukrainian military emails. These attacks had a weighted Jaccard similarity of 0.8461. We eliminated such comparisons from the results of an experiment to improve CTI quality.



4/27/22, 4:16 PM Asylum Ambuscade: State Actor Uses Compromised Private Ukrainian Military Emails to Target European Governments and Refugee Mo...

Asylum Ambuscade: State Actor Uses Compromised Private Ukrainian Military Emails to Target European Governments and Refugee Movement

proofpoint.com/us/blog/threat-insight/asylum-ambuscade-state-actor-uses-compromised-private-ukrainian-military-emails

March 1, 2022

Blog
Threat Insight
Asylum Ambuscade: State Actor Uses Compromised Private Ukrainian Military Emails to Target European Governments and Refugee Movement

March 01, 2022 Michael Raggi, Zydeca Cass and the Proofpoint Threat Research Team

Key Takeaways

- Proofpoint has identified a likely nation-state sponsored phishing campaign using a possibly compromised Ukrainian armed service member's email account to target European government personnel involved in managing the logistics of refugees fleeing Ukraine.
- The email included a malicious macro attachment which attempted to download a Lua-based malware dubbed SunSeed.
- The infection chain used in this campaign bears significant similarities to a historic campaign Proofpoint observed in July 2021, making it likely the same threat actor is behind both clusters of activity.
- Proofpoint is releasing this report in an effort to balance accuracy with responsibility to disclose actionable intelligence during a time of high-tempo conflict.

Asylum Ambuscade: State Actor Uses Compromised Private Ukrainian Military Emails to Target European Governments and Refugee Movement

proofpoint.com/us/blog/threat-insight/asylum-ambuscade-state-actor-uses-compromised-private-ukrainian-military-emails

March 1, 2022

Blog
Threat Insight
Asylum Ambuscade: State Actor Uses Compromised Private Ukrainian Military Emails to Target European Governments and Refugee Movement

March 01, 2022 Michael Raggi, Zydeca Cass and the Proofpoint Threat Research Team

Key Takeaways

- Proofpoint has identified a likely nation-state sponsored phishing campaign using a possibly compromised Ukrainian armed service member's email account to target European government personnel involved in managing the logistics of refugees fleeing Ukraine.
- The email included a malicious macro attachment which attempted to download a Lua-based malware dubbed SunSeed.
- The infection chain used in this campaign bears significant similarities to a historic campaign Proofpoint observed in July 2021, making it likely the same threat actor is behind both clusters of activity.
- Proofpoint is releasing this report in an effort to balance accuracy with responsibility to disclose actionable intelligence during a time of high-tempo conflict.

Overview

"Ambuscade: *To attack suddenly and without warning from a concealed place*"

Proofpoint researchers have identified a phishing campaign originating from an email address (ukr[.]net) that appears to belong to a compromised Ukranian armed service member. This discovery comes on the heels of alerts by the Ukrainian Computer Emergency Response Team (CERT-UA) and the State Service of Special Communications and Information Protection of Ukraine about widespread phishing campaigns targeting private email accounts of Ukrainian armed service members by 'UNC1151', which Proofpoint tracks as part of TA445. The email observed by Proofpoint may represent the next stage of these attacks. The email included a malicious macro attachment which utilized social engineering themes pertaining to the Emergency Meeting of the NATO Security Council held on February

Figure 7. APT reports with similar content but different layouts and visuals.

There is a APT report pair in Table 14 with the names "Latest Trickbot Campaign Delivered via Highly Obfuscated-JS File.pdf" and "Obfuscation Tools Found in the Capesand Exploit Kit Possibly Used in KurdishCoder Campaign.pdf". The first report is about the Trickbot banking trojan activity, a variant of the malware with the signature "TrojanSpy.Win32.TRICKBOT.TIGOCDC" discovered by Trend Micro. It is known to be distributed in a spam email that contains a Microsoft Word document with enabled macro. The other report is about an attack campaign named "KurdishCoder".

The malware sample was reversed in the report, and a method named Kirkuk (a city in Iraq) attracted attention. The attacker left many keywords, such as "Kirkuk" and "Kurd" in the source code. Knowing this background information, we investigated source and target nodes in the graph shown in Figure 8. Two shared vulnerabilities existed, CVE-2017-5689 [83], and CVE-2019-11932 [84], which were abused by both campaigns. These vulnerabilities showed us how APTs targeted enterprise-level infrastructures and popular Android applications like WhatsApp. These CVEs acted as a go-between for APT-33 group, Mirai and NjRAT malwares.

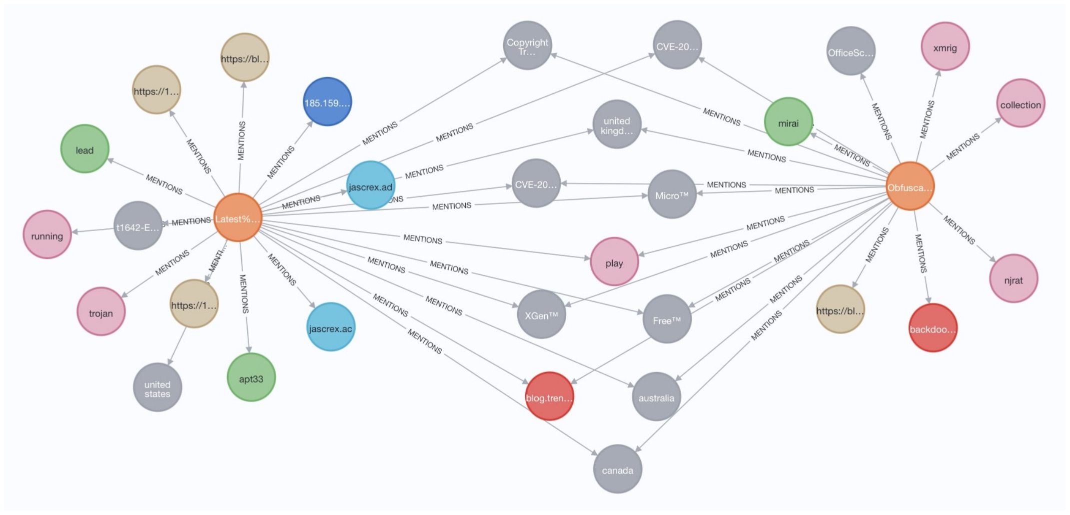


Figure 8. Relationships between two APT reports about Trickbot and KurdishCoder Campaigns.

4.2. Formation and Interpretation of the Strategic CTI

Strategic CTI helps its consumers look in the right direction. Taking action against a specific IoC is like looking at a target using binoculars. However, focusing too closely on a target closes our eyes to other threats directed toward us. Our study highly contextualized technical and tactical CTI on a graph to gain an appropriate perspective. We developed the following strategic CTI by watching the graph from the attackers' point of view.

1. By considering Table 1, threat hunting is critical since dwell time can be dramatically decreased with internal detection efforts. Even though external CTI feeds are effective in gaining confidence about cyber attacks, organizations must accept that external notifications are not enough to evade targeted APT attacks. Organizations had better invest in intrusion detection systems (IDS), endpoint detection and response (EDR) solutions, and threat hunting activities rather than relying solely on external CTI feeds.
2. Lazarus Group, Turla, APT 28, and Hacking Team APT groups have the highest centrality measurements. It has been believed that the Lazarus Group originates from North Korea, Turla and APT 28 from Russia, and the Hacking Team from Italy [85]. The most active APT groups are not attributed to a single country. We believe that international developments and political tensions targeting these countries may trigger corresponding threat actors. The existence of an APT group also causes the formation of an opposing force.
3. According to Table 15, Trojans and RATs were the top encountered malware types. This indicates that file-borne attack vectors are the most practical and impactful. Organizations must utilize proxy servers to pass client requests and responses through the Internet content adaptation protocol (ICAP) servers. By using an ICAP server, virus scanning on files in front lines provides proactive protection. Defending the systems against email attachments and file-based contents is a strategic decision to prevent further attacks.
4. The most influential malware families were Mimikatz, PlugX, Mimic, and Rogue. Mimikatz is an open-source credential dumping program to obtain account login and password information. PlugX is a remote administration tool (RAT). Mimic is ransomware-type malware. Rogue is a fake malware removal tool. All of the said malwares reached its goals by deceiving end users. We want to emphasize the importance of cybersecurity awareness training. Investing in human capital is significantly more cost-effective than covering the damage caused by APT attacks.
5. According to Table 16, Iran plays a key role in APT landscape as a country. This situation might be due to Iran being the attacked country, the starting point of the attacks, or a transit point for the attacks. Organizations might pay attention for incoming and outgoing network traffic related to IP blocks of Iran.

5. Conclusions and Future Research

Plenty of studies focus on APT attack detection for matching network indicators with CTI block lists and malware analysis, APT attribution, CTI sharing methods, and applications of machine learning techniques on technical CTI. However, investigating APT attacks to gain a strategic perspective is a rare area of research. Strategic CTI is crucial since it guides SOC analysts, purple and blue team specialists, and enterprise and government-level decision-makers. APT analysis reports are valuable sources for figuring out threat actors' past, present, and future activities.

Our study aimed to produce strategic CTI by transforming 40,358 pages of APT reports into a graph consisting of technical and tactical CTI. By discovering and evaluating the entire APT landscape, we had the opportunity to procure high-level intelligence. Graph algorithms about centrality, community detection, and similarity analyses were applied, and results were presented comparatively. Graph-based analysis techniques performed operations by handling graphs in terms of vertices and edges by ignoring the CTI context. The proposed framework ended up with the interpretation of strategic CTI. We discovered the most malicious APT groups, such as Lazarus Group, Turla, APT 28, and Hacking Team, as they took highly influential positions on the graph of CTI. These APT groups had direct relationships with approximately 2% of all APT reports. We observed that they could reach the rest of the APT report nodes after two hops of traversal. Also, Mimikatz, Plugx, Mimic, and Rogue outshone as the most potent malware.

When we discovered hidden communities in the graph, we observed divergent community counts and sizes. KCD, LP, and Leiden algorithms detected 16, 78, and 158 communities among 1980 APT reports, respectively. This means that the constructed graph of CTI had such a high contextuality that APT attacks have many common IoCs. Similar APT reports were composed by different authors to investigate different APT attacks. When an analyst looks at 40358 pages of analysis reports, these reports look unrelated at first glance. However, our approach revealed hidden contacts between report pairs.

We encountered some challenges throughout the study. Some APT reports have different names, but the entire content was the same. To eliminate such cases, we compared the SHA-256 hashes of the files in the reports pairwise and deleted duplicates. In order to produce high-quality strategic CTI, we needed to uncover what sort of technical and tactical CTI information increases connectivity between vertices in the graph. We determined 37 types of data points consisting of IoCs and TTPs.

In the future, we will enlarge the graph of CTI by integrating data mined from Darknet forums. Darknet forums will help us track the digital footprints of threat actors. We suspect that members of APT groups may unintentionally leave some evidence on Darknet. We may also encounter the same evidence on an APT report or network traffic. Catching a clue about the identity of a threat actor will help us interpret the motivation behind future attacks. Deep diving into attackers' cyber habitats may yield a strategic advantage.

Author Contributions: Conceptualization, B.G.; methodology, B.G.; software, B.G.; validation, B.G. and M.D.; formal analysis, B.G.; investigation, B.G.; resources, B.G.; data curation, B.G.; writing—original draft preparation, B.G.; writing—review and editing, M.D.; visualization, B.G.; supervision, M.D.; project administration, M.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data is accessible via the Zenodo repository at the following URL: https://doi.org/10.5281/zenodo.12741055 (accessed on 14 July 2024).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

APT	Advanced Persistent Threat
TTP	Tactics, Techniques, Procedures
CTI	Cyber Threat Intelligence
IoC	Indicators of Compromise
IP	Internet Protocol
URL	Uniform Resource Locator
SOC	Security Operations Center
OSINT	Open Source Intelligence
CPS	Cyber-Physical Systems
IoC	Internet of Things
BiLSTM	Bidirectional Long Short-Term Memory
GCN	Graph Convolutional Networks
DC	Degree Centrality
BC	Betweenness Centrality
CC	Closeness Centrality
EC	Eigenvector Centrality
KCD	K-Core Decomposition
LP	Label Propagation
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
MD5	Message Digest Method 5
SHA-1	Secure Hash Algorithm 1
SHA-256	Secure Hash Algorithm 256-bit
C2	Command and Control
RegEx	Regular Expression
CVE	Common Vulnerabilities and Exposures
IDS	Intrusion Detection System
EDR	Endpoint Detection and Response
RAT	Remote Administration Tool
ICAP	Internet Content Adaptation Protocol

References

1. Alshamrani, A; Myneni, S.; Chowdhary, A; Huang, D. A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Communications Surveys & Tutorials* **2019**, *21*, pp. 1851–1877. http://dx.doi.org/10.1109/COMST.2019.2891891
2. Yan, D; Liu, F.; Jia, K. Modeling an Information-Based Advanced Persistent Threat Attack on the Internal Network. *IEEE International Conference on Communications (ICC)*. **2019**, pp. 1–7. https://doi.org/10.1109/ICC.2019.8761077
3. Ruohonen, S.; Kirichenko, A.; Komashinskiy, D.; Pogosova, M. Instrumenting OpenCTI with a Capability for Attack Attribution Support. *Forensic Sciences*. **2024**, *4*, pp. 12–23. https://doi.org/10.3390/forensicsci4010002
4. Alzahrani, I.Y.; Lee, S.; Kim, K. Enhancing Cyber-Threat Intelligence in the Arab World: Leveraging IoC and MISP Integration. *Electronics*.**2024**, *13*, 2526. https://doi.org/10.3390/electronics13132526
5. Cho, J.; Gong, S. Dynamic Data Abstraction-Based Anomaly Detection for Industrial Control Systems. *Electronics*. **2024**, *13*, 158. https://doi.org/10.3390/electronics13010158
6. Duary, S.; Choudhury, P.; Mishra, S.; Sharma, V.; Rao, D. D.; Adedapo P. A. Cybersecurity Threats Detection in Intelligent Networks using Predictive Analytics Approaches. *International Conference on Innovative Practices in Technology and Management (ICIPTM)*. **2024**, *4*, pp. 1–5. https://doi.org/10.1109/ICIPTM59628.2024.10563348
7. Hasan, K.; Shetty, S.; Islam, T.; Ahmed, I. Predictive Cyber Defense Remediation against Advanced Persistent Threat in Cyber-Physical Systems. *International Conference on Computer Communications and Networks (ICCCN)*. **2022**, pp. 1–10. https://doi.org/10.1109/ICCCN54977.2022.9868886

8. Kao, D-Y. Performing an APT Investigation: Using People-Process-Technology-Strategy Model in Digital Triage Forensics. *IEEE 39th Annual Computer Software and Applications Conference*. **2015**, *3*, pp. 47–52. https://doi.org/10.1109/COMPSAC.2015.10
9. Bhardwaj, A.; Kaushik, K.; Alomari, A.; Alsirhani, A.; Alshahrani, M.M.; Bharany, S. BTH: Behavior-Based Structured Threat Hunting Framework to Analyze and Detect Advanced Adversaries. *Electronics*, **2022**, *11*, 2992.
https://doi.org/10.3390/electronics11192992
10. Gan, C.; Lin, J.; Huang, D.-W.; Zhu, Q.; Tian, L. Advanced Persistent Threats and Their Defense Methods in Industrial Internet of Things: A Survey. *Mathematics*, **2023**, *11*, 3115. https://doi.org/10.3390/math11143115
11. Hughes, C.; Robinson, N. Vulnerability Threat Intelligence. In *Effective Vulnerability Management: Managing Risk in the Vulnerable Digital Ecosystem*; 1st edn.; John Wiley & Sons: Hoboken, NJ, USA, 2024; pp. 145–154.
12. Fischer, D.; Sauerwein, C.; Werchan, M.; Stelzer, D. An Exploratory Study on the Use of Threat Intelligence Sharing Platforms in Germany, Austria and Switzerland. *International Conference on Availability, Reliability and Security*, **2023**, *8*, pp. 1–7. https://doi.org/10.1145/3600160.3600185
13. Kim, B.; Kim, N.; Lee, S.; Cho, H.; Park, J. A Study on a Cyber Threat Intelligence Analysis (CTI) Platform for the Proactive Detection of Cyber Attacks Based on Automated Analysis. *International Conference on Platform Technology and Service (PlatCon)*, **2018**, pp. 1–6. https://doi.org/10.1109/PlatCon.2018.8472766
14. Leong, Y. The Implementation of Strategic Threat Intelligence for Business Organization. *Journal of IT in Asia*, **2021**, *9*, pp. 41–48. https://doi.org/10.33736/jita.3398.2021
15. Liu, J.; Zhan, J. Constructing Knowledge Graph from Cyber Threat Intelligence Using Large Language Model. *IEEE International Conference on Big Data (BigData)*, **2023**, pp. 516–521. https://doi.org/10.1109/BigData59044.2023.10386611
16. Gao, Y.; Li, X.; Peng, H.; Fang, B.; Yu, P. S. HinCTI: A Cyber Threat Intelligence Modeling and Identification System Based on Heterogeneous Information Network. *IEEE Transactions on Knowledge and Data Engineering*, **2022**, *34*, pp. 708–722. https://doi.org/10.1109/TKDE.2020.2987019
17. Nazari, H.; Yazdinejad, A.; Dehghantanha, A.; Zarrinkalam, F.; Srivastava, F. arXiv—P3GNN: A Privacy-Preserving Provenance Graph-Based Model for APT Detection in Software Defined Networking. 17 Jun 2024. Available online: https://arxiv.org/abs/2406.12003 (accessed on 30 Jun 2024).
18. Zhou,F.; Chang,B.; Wen, Y.; Meng, D. ProDE: Interpretable APT Detection Method Based on Encoder-decoder Architecture. *IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)* **2023**, pp. 340–347. http://dx.doi.org/10.1109/ICPADS60453.2023.00059
19. Ren, Y.; Xiao, Y.; Zhou, Y.; Zhang, Z.; Tian, Z. CSKG4APT: A Cybersecurity Knowledge Graph for Advanced Persistent Threat Organization Attribution. *IEEE Transactions on Knowledge &amp; Data Engineering* **2023**, *35*, pp. 5695–5709. https://doi.org/10.1109/TKDE.2022.3175719
20. Jung, J.W.; Lee, S.W. Security Requirement Recommendation Method Using Case-Based Reasoning to Prevent Advanced Persistent Threats.*Applied Sciences* **2023**, *13*, 1505. https://doi.org/10.3390/app13031505
21. Li, L.; Chen, W. ConGraph: Advanced Persistent Threat Detection Method Based on Provenance Graph Combined with Process Context in Cyber-Physical System Environment.*Electronics* **2024**, *13* , 945. https://doi.org/10.3390/electronics13050945
22. Wang, N.; Wen, X.; Zhang. D.; Zhao, X. arXiv—TBDetector:Transformer-Based Detector for Advanced Persistent Threats with Provenance Graph. 6 Apr 2023. Available online: https://arxiv.org/pdf/2304.02838 (accessed on 30 Jun 2024).
23. Hangsheng, Z.; Haitao, L.; Jie, L.; Ting, L.; Liru, G.; Yinlong, L.; Shujuan, C. Defense Against Advanced Persistent Threats: Optimal Network Security Hardening Using Multi-stage Maze Network Game. *EEE Symposium on Computers and Communications (ISCC)* **2020**, pp. 1–6. https://doi.org/10.1109/ISCC50000.2020.9219722
24. Cho, D.; Nguyen, H.D.; Dao, M.H. APT Attack Detection Based on Flow Network Analysis Techniques Using Deep Learning. *Journal of Intelligent and Fuzzy Systems* **2020**, 39, *pp. 1–17. http://doi.org/10.3233/JIFS-200694*
25. *Lutu, N.; Patricia, E. Using Twitter Mentions and a Graph Database to Analyse Social Network Centrality.* 6th International Conference on Soft Computing and Machine Intelligence (ISCMI) **2019**, *pp. 155–159. https://doi.org/10.1109/ISCMI47871.2019.9004313*
26. *RafieeFard, J.; Teimourpour, B. OCRR, A Fast Algorithm for Centrality-Based Graph Reduction in Social Networks.* 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP) **2024**, *pp. 1–5. https://doi.org/10.1109/AISP61396.2024.10475252*

27. *Munir, S.; Malick, R.A.S.; Ferretti, S. A Network Analysis-Driven Framework for Factual Explainability of Knowledge Graphs.* IEEE Access **2024**, , 11, *pp. 28071–28082. https://doi.org/10.1109/ACCESS.2024.3367971*
28. *Bendahman, N.; Lotfi, D. Unveiling Influence in Networks: A Novel Centrality Metric and Comparative Analysis through Graph-Based Models.* Entropy **2024**, 26, *486. https://doi.org/10.3390/e26060486*
29. *Verma, A.K.; Saxena, R.; Jadeja, M.; Bhateja, V.; Lin, J.C.W. Bet-GAT: An Efficient Centrality-Based Graph Attention Model for Semi-Supervised Node Classification.* Applied Sciences **2023**, 13, *847. https://doi.org/10.3390/app13020847*
30. *Ai, J.; Su, Z.; Wang, K.; Wu, C.; Peng, D. Decentralized Collaborative Filtering Algorithms Based on Complex Network Modeling and Degree Centrality.* IEEE Access **2020**, 8, *pp. 151242–151249. https://doi.org/10.1109/ACCESS.2020.3017701*
31. *Krnc, M.; Škrekovski, R. Group Degree Centrality and Centralization in Networks.*Mathematics **2020**, 8, *1810. https://doi.org/10.3390/math8101810*
32. *Cai, B.; Zeng, L.; Wang, Y.; Li, H.; Hu, Y. Community Detection Method Based on Node Density, Degree Centrality, and K-Means Clustering in Complex Network.* Entropy **2019**, 21, *1145. https://doi.org/10.3390/e21121145*
33. *Powell, J.; Hopkins, M. Graph analytics techniques. In* A Librarian's Guide to Graphs, Data and the Semantic Web; *1st edn.; Chandos Publishing: Hull, East Yorkshire, England, 2015; pp. 167–174.*
34. *Umunnakwe, A.; Sahu, A.; Davis, K. Multi-Component Risk Assessment Using Cyber-Physical Betweenness Centrality.* IEEE Madrid PowerTech **2021**, *pp. 1–6. https://10.1109/PowerTech46648.2021.9494796*
35. Perez, C; Germon, R. Graph Creation and Analysis for Linking Actors: Application to Social Data. In *Automating Open Source Intelligence*; Layton, R., Watters, P. A. 1st edn.; Syngress: Boston, MA, USA, 2016; pp. 103–129.
36. Nandini, Y.V.; Lakshmi, T.J.; Enduri, M.K.; Sharma, H. Link Prediction in Complex Networks Using Average Centrality-Based Similarity Score. *Entropy*, **2024**, *26*, 433. https://doi.org/10.3390/e26060433
37. Wu, W.; Wang, S.; Liu, B. Software Fault Localization Based on Weighted Association Rule Mining and Complex Networks. *Mathematics*, **2024**, 12, 2113. https://doi.org/10.3390/math12132113
38. Segarra, S.; Ribeiro, A. Stability and Continuity of Centrality Measures in Weighted Graphs. *IEEE Transactions on Signal Processing*, **2016**, *64*, pp. 543–555. https://doi.org/10.1109/TSP.2015.2486740
39. Zhang. Z.; Wang, J.; Xu, Y. Research on node influence in complex network based on multi-scale centrality algorithm. *Procedia Computer Science*, **2023**, *228*, pp. 1128–1133. https://doi.org/10.1016/j.procs.2023.11.147
40. Srichandra, I. V.; Bhadra, P. Community Detection Using Graph Attention Networks Clustering Algorithm. *IEEE International Conference for Convergence in Technology (I2CT)*, **2024**, pp. 1–6. https://doi.org/10.1109/I2CT61223.2024.10543468
41. Rajita, B.S.A.S.; Panda, S. Community Detection Techniques for Evolving Social Networks. *International Conference on Cloud Computing, Data Science & Engineering* , **2019**, pp. 681–686. https://doi.org/10.1109/CONFLUENCE.2019.8776896
42. Gao, S.; Xu, J.; Li, X.; Fu, F.; Zhang, W.; Ouyang, W.; Tao, Y.; Cui. B. K-Core Decomposition on Super Large Graphs with Limited Resources. arXiv **2021**, https://doi.org/10.48550/arXiv.2112.14840
43. Laishram, R.; Soundarajan, S. On Finding and Analyzing the Backbone of the k-Core Structure of a Graph. *IEEE International Conference on Data Mining (ICDM)*, **2022**, pp. 1017–1022. https://doi.org/10.1109/ICDM54844.2022.00124
44. Bhatia, V.; Rani, R. An efficient influence based label propagation algorithm for clustering large graphs. *International Conference on Infocom Technologies and Unmanned Systems (ICTUS)*, **2017**, pp. 1–7. https://doi.org/10.1109/ICTUS.2017.8286044
45. Xie, T.; Wang, B.; Kuo, C.-C. J. GraphHop: An Enhanced Label Propagation Method for Node Classification. *IEEE Transactions on Neural Networks and Learning Systems*, **2023**, *34*, pp. 9287–9301. https://doi.org/10.1109/TNNLS.2022.3157746
46. Blekanov, I.; Bodrunova, S.S.; Akhmetov, A. Detection of Hidden Communities in Twitter Discussions of Varying Volumes. *Future Internet*, **2021**, *13*, 295. https://doi.org/10.3390/fi13110295
47. Kadem, B. A.; Al-sultany, G. Enhancing Community Detection Using Maximal and Maximum Cliques into Hierarchical Algorithms. *IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, **2024**, pp. 1–6. https://doi.org/10.1109/AIMS61812.2024.10512947
48. Gupta, S. K.; Singh, D. P. CBLA: A Clique Based Louvain Algorithm for Detecting Overlapping Community. *International Conference on Machine Learning and Data Engineering*, **2023**, *218*, pp. 2201–2209. https://doi.org/10.1016/j.procs.2023.01.196

49. Dermawan, F.; Kwang, C. H.; Adijanto, M. D.; Rakhmawati, N. A.; Basara, N. R. Product Recommendations through Neo4j by Analyzing Patterns in Customer Purchases. *International Conference in Emerging Technologies for Sustainability and Intelligent Systems*, **2024**, pp. 1–4. https://doi.org/10.1109/ICETSIS61505.2024.10459357
50. Edie, K.; Mckee, C.; Duby, A. Extending Threat Playbooks for Cyber Threat Intelligence: A Novel Approach for APT Attribution. *11th International Symposium on Digital Forensics and Security*, **2023**, pp. 1–6. https://doi.org/10.1109/ISDFS58141.2023.10131867
51. Chachoo, M. A. Social Network Analysis Based Criminal Community Identification Model with Community Structures and Node Attributes. *4th International Conference on Smart Systems and Inventive Technology*, **2022**, pp. 334–339. https://doi.org/10.1109/ICSSIT53264.2022.9716286
52. Kim, H. M.; Song, H. M.; Seo, J. W.; Kim, H. K. Andro-Simnet: Android Malware Family Classification using Social Network Analysis. *16th Annual Conference on Privacy, Security and Trust* , **2018**, pp. 1–8. https://doi.org/10.1109/PST.2018.8514216
53. Shoeibi, N.; Shoeibi, N.; Chamoso, P.; Alizadehsani, Z.; Corchado, J.M. A Hybrid Model for the Measurement of the Similarity between Twitter Profiles. *Sustainability*, **2022**, *14*, 4909. https://doi.org/10.3390/su14094909
54. Haque, M. A.; Shetty, S.; Kamhoua, C. A.; Gold, K. Adversarial Technique Validation & Defense Selection Using Attack Graph & ATT&CK Matrix. *International Conference on Computing, Networking and Communications*, **2023**, pp. 181–187. https://doi.org/10.1109/ICNC57223.2023.10074241
55. Desku, A.; Raufi, B.; Luma, A.; Selimi, B. Cosine Similarity through Control Flow Graphs For Secure Software Engineering. *International Conference on Engineering and Emerging Technologies*, **2021**, pp. 1–4. https://doi.org/10.1109/ICEET53442.2021.9659648
56. Adamic, L.A.; Adar, E. Friends and neighbors on the web. *Social Networks*. **2003**, *25*, pp. 211–230. https://doi.org/10.1016/S0378-8733(03)00009-1
57. Nandini, Y.V.; Lakshmi, T.J.; Enduri, M.K.; Sharma, H. Link Prediction in Complex Networks Using Average Centrality-Based Similarity Score.*Entropy* **2024**, *26*, 433. https://doi.org/10.3390/e26060433
58. Chalkiadakis, G.; Ziogas, I.; Koutsmanis, M.; Streviniotis, E.; Panagiotakis, C.; Papadakis, H. A Novel Hybrid Recommender System for the Tourism Domain. *Algorithms*, **2023**, *16*, 215. https://doi.org/10.3390/a16040215
59. Azevedo, R.; Medeiros, I.; Bessani, A. PURE: Generating Quality Threat Intelligence by Clustering and Correlating OSINT. *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications*, **2019**, pp. 483–490. https://doi.org/10.1109/TrustCom/BigDataSE.2019.00071
60. Hofer-Schmitz, K.; Kleb, U.; Stojanović, B. The Influences of Feature Sets on the Detection of Advanced Persistent Threats. Electronics 2021, 10, 704. https://doi.org/10.3390/electronics10060704
61. Maglaras, L.; Ferrag, M. A.; Derhab, A.; Mukherjee, M.; Janicke, H.; Rallis, S. Threats, Countermeasures and Attribution of Cyber Attacks on Critical Infrastructures. *EAI Endorsed Transactions on Security and Safety*, **2018**, *5*. https://doi.org/10.4108/eai.15-10-2018.155856
62. Baghirov, E. Techniques of Malware Detection: Research Review. *15th International Conference on Application of Information and Communication Technologies (AICT)*, **2021**. https://doi.org/10.1109/AICT52784.2021.9620415
63. Goel, N.; Sethi, N. Cyber Threat Intelligence: a Survey on Progressive Techniques and Challanges. *International Conference on Big Data, IoT, Cyber Security and Information Technology*, **2022**, *13*. pp. 65–70.
64. Tatam, M.; Shanmugam, B.; Azam, S.; Kannoorpatti, K. A review of threat modelling approaches for APT-style attacks. *Heliyon*, **2021**, *7*. pp. 1–19. https://doi.org/10.1016/j.heliyon.2021.e05969
65. MITRE ATT&CK Framework. Available online: https://attack.mitre.org/.
66. Caballero, J.; Gomez, G.; Matic, S.; Sánchez, G.; Sebastián, S; Villacañas, A. The Rise of GoodFATR: A Novel Accuracy Comparison Methodology for Indicator Extraction Tools. *Future Generation Computer Systems* **2023**, *7*, pp. 74–89. https://doi.org/10.48550/arXiv.2208.00042
67. APT Groups and Operations. Available online: https://docs.google.com/spreadsheets/d/1H9_xaxQHpWaa4O_Son4Gx0YOIzlcBWMsdvePFX68EKU/edit?gid=1864660085#gid=1864660085 (accessed on 1 July 2024).
68. Malpedia Malware Families. Available online: https://malpedia.caad.fkie.fraunhofer.de/families (accessed on 1 July 2024).
69. MITRE Att&ck Enterprise, Mobile and ICS Techniques. Available online: https://attack.mitre.org/techniques/enterprise (accessed on 1 July 2024).
70. Yan, D.; Wu, T.; Liu, Y.; Gao, Y. An efficient sparse-dense matrix multiplication on a multicore system. *17th International Conference on Communication Technology*, **2017**. pp. 1880–1883. https://doi.org/10.1109/ICCT.2017.8359956

71. Borgatti, S.; Everett, M. A Graph-Theoretic Perspective on Centrality. *Social Networks* , **2006**, *28*. pp. 466–484. https://doi.org/10.1016/j.socnet.2005.11.005
72. Freeman, L. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, **1977**, *40*. pp. 35–41. https://doi.org/10.2307/3033543
73. Brandes, U. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, **2004**, *25*. pp. 163–177. https://doi.org/10.1080/0022250x.2001.9990249
74. Bavelas, A. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, **1950**, *22*. pp. 725–730. https://doi.org/10.1121/1.1906679
75. Negre, C.F.A.; Morzan, U.N.; Hendrickson, H.P.; Pal, R.; Lisi, G.P.; Loria, J.P.; Rivalta, I.; Ho, J.; Batista, V.S. Eigenvector Centrality for Characterization of Protein Allosteric Pathways. *The Proceedings of the National Academy of Sciences (PNAS)*, **2018**, *115*, pp. 12201–12208. https://doi.org/10.1073/pnas.1810452115
76. Newman, M.E.J. The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, **2008**, *2*. pp. 1–12. https://doi.org/10.1057/978-1-349-95121-5_2565-1
77. Kabir, H.; Madduri, K. Parallel k-Core Decomposition on Multicore Platforms. *IEEE International Parallel and Distributed Processing Symposium Workshops* , **2017**. pp. 1482–1491. https://doi.org/10.1109/IPDPSW.2017.151
78. Dasari, N. S.; Desh, R.; Zubair, M. ParK: An efficient algorithm for k-core decomposition on multicore processors. *IEEE International Conference on Big Data*, **2014**. pp. 9–16. https://doi.org/10.1109/BigData.2014.7004366
79. Raghavan, N.; Albert, R.; Kumara, S. Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks. *Physical Review E*, **2007**, *76*. pp. 2375–2387. https://doi.org/10.1103/PhysRevE.76.036106
80. Traag, V.; Waltman, L.; and van Eck, N. J. From Louvain to Leiden: Guaranteeing Well-Connected Communities. *Scientific Reports*, **2019**, *9*. pp. 5233–5258. https://doi.org/10.1038/s41598-019-41695-z
81. Li, X.; Li, P. Rejection Sampling for Weighted Jaccard Similarity Revisited. *Proceedings of the AAAI Conference on Artificial Intelligence*, **2021**, *35*. pp. 4197–4205. https://doi.org/10.1609/aaai.v35i5.16543
82. Frigo, M.; Cruciani, E.; Coudert, D.; Deriche, R.; Natale, E.; Deslauriers-Gauthier S. Network alignment and similarity reveal atlas-based topological differences in structural connectomes. *Network Neuroscience*, **2021**, *5*. pp. 711–733. https://doi.org/10.1162/netn_a_00199
83. CVE-2017-5689 Vulnerability Details. Available online: https://www.cvedetails.com/cve/CVE-2017-5689/ (accessed on 11 July 2024).
84. CVE-2019-11932 Vulnerability Details. Available online: https://www.cvedetails.com/cve/CVE-2019-11932/ (accessed on 11 July 2024).
85. MITRE APT Groups. Available online: https://attack.mitre.org/groups/ (accessed on 11 July 2024).
86. Gulbay, B. Dataset and Source Code for the Paper: A Framework for Developing Strategic Cyber Threat Intelligence from Advanced Persistent Threat Analysis Reports Using Graph-Based Algorithms. Zenodo, 2024. https://doi.org/10.5281/zenodo.12741055

Short Biography of Authors

Burak Gulbay Received the BS degree in Computer Engineering from Atılım University in 2013 and also completed his minor in Software Engineering. He received MSc degree in Computer Engineering from Gazi University in 2013. He has been studying Information Security Engineering as PhD candidate at Gazi University. He is currently working as Next Generation Technologies Lead in defence industry. He participated in various software development projects as a software engineer, team manager, project manager, and consultant in both the public and private sectors. His research interests include Cyber Threat Intelligence, Graph Theory, Data Intensive Cloud Native Systems, and Digital Transformation

Mehmet Demirci Mehmet Demirci received his B.S. degree in computer science and mathematics (double major) from Purdue University in 2006, and his M.Sc. and Ph.D. degrees in computer science from Georgia Institute of Technology in 2009 and 2013, respectively. He is currently an assistant professor with the Department of Computer Engineering, Faculty of Engineering, Gazi University. His current research interests are SDN, NFV, network security, network architecture & future Internet.