AUTOMA: Automated Generation of Attack Hypotheses and Their Variants for Threat Hunting using Knowledge Discovery

Boubakr Nour Makan Pourzandi Rushaan Kamran Qureshi Mourad Debbabi

***Abstract*—Threat hunting is a proactive security defense line exercised to uncover attacks that could circumvent conventional detection mechanisms. It is based on an iterative approach to generate, inspect, and revise attack hypotheses. The quality of these hypotheses is essential to prove/refute the existence of an attack. Today, attack hypotheses are often generated manually by security analysts. The generation process requires elusive expertise, is costly, and is prone to produce a large number of irrelevant hypotheses without considering the attack variants. In this paper, we address the aforementioned challenges by designing AUTOMA, a solution that automates the generation of relevant hypotheses and their variants using knowledge discovery. AUTOMA incorporates the system telemetry in combination with a knowledge base of existing attacks, techniques, and their relationships to mine the most relevant hypotheses. In order to increase the relevance of the generated hypotheses, AUTOMA examines these hypotheses by applying matching-based similarity, success, likelihood, and criticality evaluations. These evaluations are based on the past occurrences of the techniques part of a hypothesis in the system telemetry and the knowledge base. Additionally, AUTOMA uses sequence success, sequence alignment, and hierarchical similarity approach for generating potential attack variants of a hypothesis taking into account the dynamism and stealthiness of attackers in coming up with alternative attack steps. We extensively evaluate the effectiveness and efficiency of AUTOMA using a real dataset for 284 attack campaigns distributed over 57 advanced persistent threats. The obtained results show that AUTOMA is able to generate the relevant hypothesis (top 3), with a large reduction rate (up to 99%), and fast execution time (up to 8 minutes for proposing the relevant hypothesis and 10 seconds for variants generation).**

***Index Terms*—Cybersecurity, Cyber Threat Intelligence, Threat Hunting, Advanced Persistent Threat, Proactive Investigation, Hypothesis Generation, Variants Generation**

I. Introduction

Security Operations Center (SOC) team is overwhelmed with a constant barrage of security threats and alerts. Micro Focus [1] reported that SOC receives millions of alerts (≈ 20,000 events per second) that need to be deeply investigated by security experts. The latter requires not only expertise but also a huge amount of time and effort that might result in alarm fatigue [2], resulting consequently in a lack of responsiveness and vigilance toward potential security incidents by missing critical security alerts and/or late investigation. To prevent alarm fatigue while strengthening security, it is essential to bring intelligence and automation to SOC aiming at prioritizing security incidents and alerts, reducing the analysis and investigation time, and proactively generating hypotheses for potential attacks. To address this, an efficient threat hunting system [3] could significantly help by proposing hypotheses for possible attacks to be investigated by the SOC team. The quality of these hypotheses has a major impact on the overall efficiency of the threat hunting process [4]. At the same time, it is critical to investigate all possible hypotheses to prove the existence of threats (*e.g.,* Advanced Persistent Threats – APT).

Many research efforts use the received Indicator of Compromise (IoC) as a trigger to investigate previous activities in the System Telemetry (ST) aiming at (i) detecting the attack based on previous activities witnessed in the ST [5], and/or (ii) predicting the attacker's next step in order to prevent its advance [6]. Other work has been proposed to automate threat detection based on system observations (*e.g.,* [7]); however, they heavily rely on experts' intervention or approaches to scrutinize myriad and tedious log files and system activities, discover threats, and then apply different techniques to investigate their relevance. Although statistical/analytical/behavior-based approaches might be used [3], the entire process is still cumbersome where thousands of log entries need to be inspected on a daily basis. Other proposed solutions are limited to generating hypotheses for validating the existing conditions (*e.g.,* [8]) or verifying very specific threats (*e.g.,* [9]). Other attempts have been presented to generate attack hypotheses (*e.g.,* [10]) but they are not automated making them cumbersome to find relevant hypotheses. Overall, these solutions (*i.e.,* detection and prediction) are based on the received IoCs and confined to what has been seen in the system telemetry not leveraging the knowledge around existing threats.

Motivating example: APT41 [11] is a state-sponsored espionage group against telecommunications, high-tech, and healthcare sectors that have been active since mid-2016. The objective of this APT is to establish and maintain strategic access using different techniques and software. According to MITRE ATT&CK[1], APT41 comprises 59 techniques, 13

The research reported in this article is supported by Ericsson Research and the Security Research Centre of Concordia University. This support stems from a collaborative partnership with the National Cybersecurity Consortium (NCC) under the Cyber Security Innovation Network (CSIN).

B. Nour and M. Pourzandi are with Ericsson Security Research, Montréal, QC H4S 0B6, Canada (emails: boubakr.nour@ericsson.com, makan.pourzandi@ericsson.com).

RK. Qureshi and M. Debbabi are with the Gina Cody School of Engineering and Computer Science, Concordia Institute for Information Systems Engineering, Concordia University, Montréal, QC H3G 1M8, Canada (emails: rushaan.kamranqureshi@concordia.ca, mourad.debbabi@concordia.ca).

(Corresponding author: Boubakr Nour)

[1] APT41: https://attack.mitre.org/groups/G0096/

software, and 211 relationships.

We assume that a network is under APT41 attack campaign and the SOC receives an IoC (*e.g.,* security technique) of "`System Network Configuration Discovery`". Note that the same technique can be used by multiple APTs, which opens the scope for attackers to build different variants of the same APT. For example, the received technique "`System Network Configuration Discovery`" is part of APT18, APT39, APT41, among others. This leads to two challenges: (i) the need for extensive security knowledge to investigate all possible/potential attacks, and (ii) the need to generate the most relevant attack hypotheses including their variants in order to reduce the required time and resources to investigate all received security events/alerts. In addition, APT41 has recently evolved to use Google Command and Control, Google Sheets, and Drive instead of already-used techniques [12]. This dynamism in threat actors' behavior motivates the need to proactively find potential APT variants. Moreover, threat hunting would become more complex due to the immense number of applications, users, and data, in addition to the massive virtualization and cloudification of networking, storage, and computation in modern networks [13]. The threat hunting should be fast enough to reduce the attack window. The more relevant and fewer hypotheses generated by threat hunting, the faster threats can be proactively investigated.

To fill the gap, there is a need to automate the generation of attack hypotheses and their potential variants based on the current system telemetry and available knowledge (*i.e.,* security expertise) with minimum human intervention. In this work, we aim to generate attack hypotheses and their variants for potential APTs that might exist in the system. Toward this, we build a security context based on the received IoC by connecting what has been seen as activities in the ST with what the SOC knows about existing attacks in the Knowledge Base (KB). Upon the reception of a technique, we build a graph that represents the context of the received technique from both ST and KB using the MITRE ATT&CK framework [14]. Our proposed solution generates a graph of possible steps by mining existing ST and KB and then applies a multi-level evaluation (*i.e.,* pruning) to reduce the number of hypotheses. More specifically, the first evaluator applies a matching-based technique to split the hypotheses based on their similarity score with known attack sequences. The second evaluator focuses on hypotheses with high-similarity scores and employs success, likelihood, and criticality techniques to preserve relevant hypotheses with the highest criticality score considering their success and occurring likelihood. The third evaluator scrutinizes hypotheses with low-similarity scores using sequence success, sequence alignment, and hierarchical similarity approach to identify potential attack variants. Note that we are not aiming at detecting the attack or predicting the attacker's next step(s) but rather examining all possible hypotheses to notify the SOC team with fewer yet relevant attack hypotheses for potential attacks and their variants.

Existing security tools (*e.g.,* Security Information and Event Management – SIEM) constitute another set of security mechanisms that generally tackle advanced reactive threat detection, but they do not automatically generate relevant hypotheses using any documented knowledge discovery techniques [3]. In our work, we consider threat hunting as a separate activity that is used to further enforce security and proactively pursue advanced threats and adversaries. Therefore, our work for hypotheses and variants generation could integrated with SIEM systems [15].

Contributions: The main contribution of this work is not to design a new correlation algorithm but rather the mixing and pruning of techniques to retrieve the most relevant attack hypotheses as well as scrutinizing the low-similarity score hypotheses to identify any potential attack variants. This provides actionable intelligence on current or potential threats, allowing effective preventative measures based on prioritized hypotheses. To this end, in this paper:

- We design AUTOMA, an automated attack hypotheses and their variants generation solution based on knowledge discovery. To the best of our knowledge, this is the first work of its kind that automates hypotheses and their variants generation and evaluates many possible hypotheses to provide the most relevant hypotheses (*i.e.,* the most relevant hypotheses are placed as the highest ranking with reasonable response time) to the SOC team.
- We propose the most likely attack variants of generated hypotheses by applying sequence alignment that aims at adding/substituting relevant technique(s) within the system telemetry and then applying hierarchical similarity using the KB to find potential variants.
- We conduct extensive experiments with real dataset [16] to show the effectiveness and efficiency of AUTOMA in reflecting the relevant hypothesis ranking, number of generated hypotheses, reduction rate, and execution time.

The rest of the paper is organized as follows: Section II reviews related work. Section III defines the threat model. Section IV describes our methodology and details our proposed solution. Section V presents the implementation of AUTOMA, and Section VI discusses the experimental results using real a dataset. Finally, the conclusion is provided in Section VII.

II. Related Work

Threat hunting vs alert prioritization: Alert prioritization [17] involves evaluating the risks posed by various threats based on their probability and potential impact. It is not explicitly engineered to detect a cyber kill chain, whereas threat hunting is an active process of finding and neutralizing threats or indicators before they can inflict significant damage [18]. Contrary to threat analysis, we primarily concentrate on the proactive identification of potential threats prior to their advancement in the kill chain. We take into account that certain techniques may not have occurred yet, which would not lead to related intrusion alarms. Our approach shares some common elements with Intrusion Detection Systems (*e.g.,* ranking and impact analysis), yet we foster proactive measures rather than reactive ones.

Threat hunting-related efforts: Work in [3] provides a comprehensive survey on threat hunting in enterprise networks. The study provides a taxonomy of existing hunting

efforts along with a discussion on standardization efforts and research directions. For instance, *Machine Learning (ML)-based threat hunting* (*e.g.,* [7], [19], [20]) helps in enabling an analytically advanced defense with minimum time and human efforts. Those solutions integrate threat intelligence with ML techniques to provide data analysis [21], prediction [22], and forecasting [23] capabilities. These techniques extract the possible attacks from past events, and they do not propose possible threats based on a knowledge base. However, our approach provides the possibility of detecting attack variants by combining both KB and ST. *Graph-based threat hunting* (*e.g.,* [24], [25], [26]) combines graph representation with graph theory to provide threat analysis and investigation. Graphs usually represent entities and events along with semantic information. Data mining and inspection can be applied to understand different activities and then predict the attacker's next step and construct that attack story [7]. Contrary to the above efforts, our approach focuses on generating attack hypotheses and their variants in an automated fashion and avoiding the heavy implications of the experts. *Rule-based threat hunting* (*e.g.,* [27], [28], [29]) uses a set of security rules (*i.e.,* policies) that identify malicious nodes or activities using various attribute-value knowledge representations. The hunting process relies on comparing the sensitivity of the objects being accessed to the corresponding attributes aiming to identify suspicious/malicious activity. Yet, those rules are too permissive and straightforward, which might be easy to exploit and bypass. *Statistical-based threat hunting* (*e.g.,* [30], [31], [32]) employs statistical analysis to identify suspicious entities/activities. These techniques require a conceptual understanding of the vulnerabilities in order to map them to a statistical perspective [33]. Those techniques do not have the capabilities to provide contextual information, which might result in a high level of false-positive alerts. Unlike those efforts, our work aims at mixing and pruning the already seen activities from system telemetry and known attack activities from knowledge to build a security context on the observed technique and generate relevant attack hypotheses and their variants.

Correlation-based efforts: Although the aforementioned solutions aim at strengthening security by automating threat detection based on system observations [5], [14], [34], they heavily rely on security expert intervention and the existence of enriched system telemetry by applying different correlation techniques. For instance, various efforts apply correlation techniques to enforce security capabilities by correlating threat intelligence indicators and system/network data into actionable insights [35]. Coefficient-based correlation analysis (*e.g.,* Pearson coefficient, Jaccard coefficient, Tanimoto coefficient) [36] can be used to extract correlative information between data (*e.g.,* logs) and CTI information (*e.g.,* attack definition), which leads to threat detection and false positive reduction. Overall, the majority of correlation-based solutions primarily concentrate on identifying threats and anomalies through the examination of historical activities. This analysis is generally aimed at either detecting the existence of an attack or predicting the next step that would be part of an attack. However, these approaches often overlook the critical aspect of threat hunting, implicitly relying on the presence of a security expert to manually formulate hypotheses regarding potential attacks, which has been automated in our work.

Unlike the aforementioned efforts that focus on manual threat hunting while ignoring attack hypothesis generation, our approach emphasizes the automation of generating hypotheses and their variations. We go beyond merely correlating past activities or predicting future steps by connecting the ST with the KB to produce the most relevant attack hypotheses and their variants that are not evident in prior activities. By doing so, we foster proactive measures rather than reactive ones and ensure generating fewer yet relevant attack hypotheses as well as identifying potential attack variants.

III. Threat Model & Assumptions

Hypothesis generation is the first step that triggers the process of threat hunting. It is an important part of the hunting loop since starting with a non-relevant or precariousness hypothesis will lead not only to spending recklessly time and resources but also to allowing more time for attackers to gain a foothold within the system [4].

The in-scope threats include existing APTs and their related variants that are launched by either external attackers or insiders through exploiting misconfigurations or vulnerabilities in the network (*e.g.,* enterprise, virtualized, or telco networks).

The out-of-scope threats include any attacks that may be effectively prevented through security patches, firewalls, intrusion prevention systems, etc. We also exclude the completely new APTs from our threat model. It is worth highlighting that the detection is out of the scope of this work.

Assumptions: We assume a monitoring tool (*e.g.,* Falco[2], SysFlow[3], DeTTECT[4]) is used to capture events and then map them into techniques used by our designed solution (*e.g.,* [37]). We also assume that the KB is constructed utilizing the MITRE ATT&CK framework and verified by security experts. It is continually updated by the SOC team based on new threat intelligence (from industrial/governmental partners). This intelligence stems from internal and external security reports/assessments, and the discovery of new attack patterns [25]. This continuous updating and enrichment facilitate adaptation to the evolving threat landscape. Finally, we assume that the ST is collected in real-time by the SOC team and provided to AUTOMA as an input in NoSQL format, enabling straightforward knowledge addition and extraction.

IV. Automated Attack Hypotheses and Their Variants Generation

This section describes AUTOMA, a solution that proactively generates attack hypotheses and their variants using knowledge discovery. It is worth mentioning that, in this paper, we focus on proactively generating hypotheses and variants

[2]Falco Project: www.falco.org
[3]SysFlow: www.sysflow.io
[4]DeTTECT: www.github.com/rabobank-cdc/DeTTECT

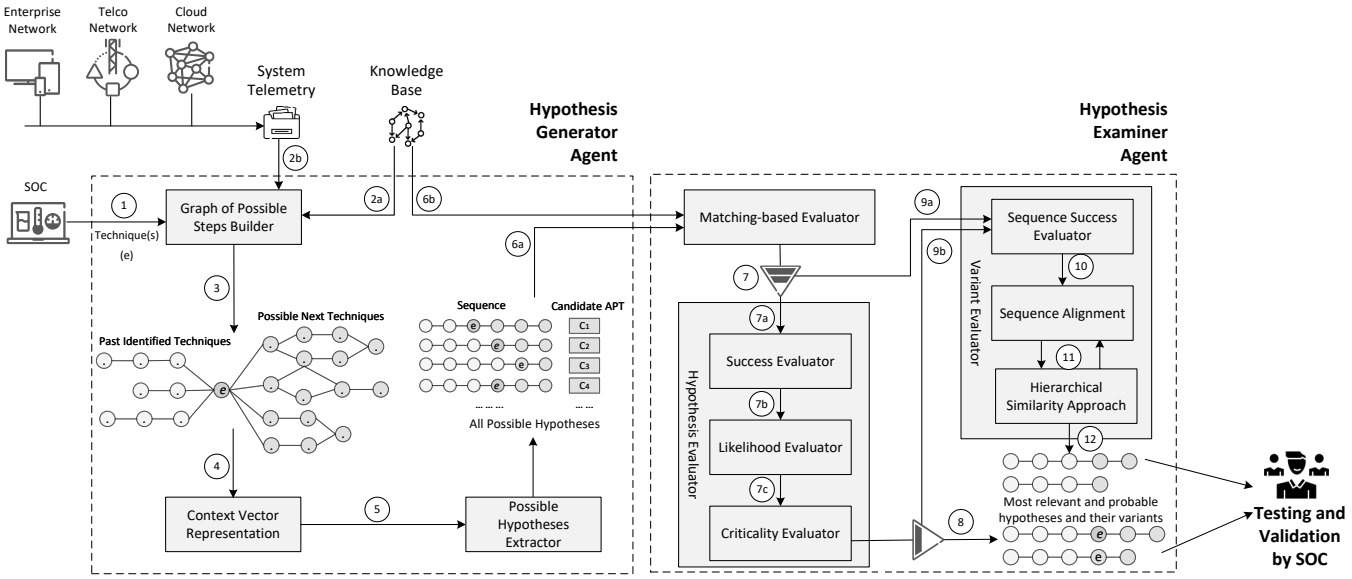


Fig. 1: AUTOMA's overview architecture, including the components of hypotheses generator and examiner agents.

and we consider other threat hunting steps (*e.g.,* hypotheses testing, investigation [4], [38]) are out of the scope of this work. In the rest of the paper, we consider one observed security technique referred to by e_0, though our approach can be easily generalized to multiple techniques (see section VI-F).

A. High-level Overview

Fig. 1 illustrates the building components of AUTOMA. To generate attack hypotheses and potential variants, AUTOMA uses a KB containing the description of attacks in a Tactics, Techniques, and Procedures (TTP) format from MITRE ATT&CK [39]. It further uses ST inputs to enrich and narrow down the knowledge discovery. It is important to highlight that we use techniques as defined by MITRE ATT&CK as the basic granularity of our solution (*e.g.,* nodes in KB and ST are all techniques).

Indeed, AUTOMA maintains two main agents: *Hypothesis Generator* (HG) and *Hypothesis Examiner* (HE), which will be explained in sections IV-C and IV-D, respectively. For a given security event (e_0), received from the SOC (Step ①) – in the APT41 attack campaign example mentioned in the introduction is "`System Network Configuration Discovery`", and assisted by KB and ST (steps ②a-②b), the HG agent is responsible for building a graph of possible steps (Step ③), performing context vector representation (Step ④), and then extracting all possible hypotheses (Step ⑤) – in the example of APT41, up to 2 million possible attack hypotheses. Using the list of all possible hypotheses, the HE agent examines/prunes those hypotheses in order to minimize the number of generated hypotheses while ensuring their relevance. The first pruning is a matching-based evaluation (Steps ⑥a-⑥b) that splits the hypotheses based on their similarity score with attacks defined in the KB. Hypotheses with high-similarity score are further pruned by the Hypothesis Evaluator (Step ⑦). The latter applies three evaluations using success, likelihood, and criticality evaluation (Steps ⑦a, ⑦b, and ⑦c, respectively). The result of this evaluator is the most relevant hypotheses (Step ⑧) and is sent to the SOC team for further testing and validation – in the example of APT41, around 100 hypotheses where the APT41 is ranked 3rd. The key idea is to (i) use, on one hand, the knowledge of different possible attacks expressed in KB (*i.e.,* what we know), and on the other hand, the knowledge of what happened captured in ST (*i.e.,* what we saw) related to the observed technique e_0, and (ii) generate all possible attack hypotheses of the ongoing attacks (based on the appearance of e_0). The evaluation of the relevance of these generated hypotheses is based on the past appearances of the technique e_0 in KB and its related past identified techniques in ST.

Additionally, the list of hypotheses with low-similarity scores from the matching-based evaluator (Step ⑨a) and the list of relevant APTs from the Hypothesis Evaluator (Step ⑨b) are further scrutinized to find any potential variants of those APTs. The objective of this evaluator is to scrutinize if the hypotheses with low-similarity scores are still potential variants of known APTs. The intuition here is that as attackers keep generating new attack variants by changing their techniques to achieve the same goals, AUTOMA will then evaluate hypotheses that are most probable to be variants of valid attacks. The Variant Evaluator evaluates the success of each received sequence to determine if it is a potential variant, applies sequence alignment to find any potential new specific sequences for different techniques (Step ⑩), and then performs hierarchical similarity to examine the impact of each aligned sequence (Step ⑪) – in the example of APT41, the Variant Evaluator finds six potential variants. The result of this evaluator is the most relevant threat variants (Step ⑫) and is sent to the SOC team for further testing and validation.

Design Challenges: We present the main challenges for developing AUTOMA and how we address each. (1) *formulating an attack hypothesis*: Formulating a hypothesis necessitates a comprehensive understanding of attack patterns/TTPs and

the enterprise environment. Unlike other work that used different distinguished formats to represent attacks and ST [10], AUTOMA employs a novel approach that combines ST and KB. This unified representation is better than fragmented representations in expressibility in hypothesis formulation and representing different complex situations factoring in potential attacks. (2) *hypothesis assessment*: The growing scale of network activities and associated data complexity presents challenges in hypothesis assessment due to data quality/relevancy uncertainties. Previous work used methods based on previous occurrences [40], we developed an evaluator that considers similarity and criticality, enabling focus on the most critical threats. (3) *automating hypothesis generation*: The dynamic nature of cyber threats makes it challenging to generate relevant hypotheses. Through modeling the expert knowledge in KB and the past events in ST, AUTOMA puts in place a mechanism that manages to unify these two worlds to generate attack hypotheses and their variants.

B. Connecting System Telemetry to Knowledge Base

ST carries significant amount of relevant information about past users' activities and behaviors. Analyzing and interpreting this information helps in identifying potential ongoing security threats (*e.g.,* based on historical activities). KB is a centralized repository that compiles various information about security threats (*e.g.,* tactics and techniques). The KB can further be extended to include technical details about specific attacks (*e.g.,* vulnerabilities, tools, and malware). Connecting ST and KB, known as Knowledge Discovery [41], helps in building holistic and effective security knowledge and decision-making. In this work, we use a Graph of Possible Steps (GoPS) to connect information from both ST to KB and build partial knowledge about possible attacks involving a security technique.

Definition 1 (GoPS). Given a security technique e_0, a search window r, and a search strategy s (*e.g.,* breadth-first, depth-first), GoPS is built by extracting past identified techniques (P_{e_0}) from the ST and possible next techniques (F_{e_0}) from the KB. GoPS is centered on e_0 and contains information related only to e_0 including observed techniques (P_{e_0}) in the system and known facts (F_{e_0}) about the next techniques that could appear after e_0 to launch attacks. Each path (*i.e.,* sequence) $p \in \mathsf{P}_{e_0}$ must have a maximum length of r and must contain e_0 as an ending (bottleneck) node. Similarly, each path (*i.e.,* sequence) $p' \in \mathsf{F}_{e_0}$ must be at a maximum of the length r and must start with e_0 as a starting node.

Note that once the GoPS is generated, the HG agent performs context vector representation to capture the semantic level of knowledge. Each node in the GoPS is converted into a vector format [42]. Capturing semantics helps in building a better understanding of techniques constituting hypotheses in the evaluation phase and their relation with previous and future security techniques.

Fig. 2 depicts an illustrative example of GoPS using a search window r of 3 and a breadth-first search strategy. The GoPS revolves around the received technique e_0 by constructing past identified techniques (P_{e_0}) that are extracted from ST (left side of node e_0) and possible next techniques (F_{e_0}) that are

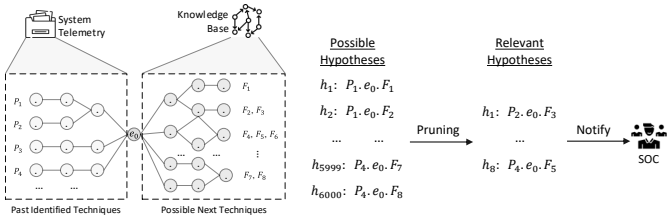


Fig. 2: An illustrative example of connecting ST and KB to generate hypotheses for a received technique (e_0).

extracted from the KB (right side of node e_0). By using a GoPS that accommodates ST and KB, AUTOMA is able to generate relevant attack hypotheses and identify them, and create a data model that represents specific threats (*e.g.,* threat actors, tactics, techniques) along with run-time activities and relationships. Appendix A provides further discussion with an example of GoPS.

APTs evolve all the time and threat actors dynamically explore new techniques that lead to the same objective. Hence, this approach (*e.g.,* pollination between different possible APTs) would allow not only taking into account the existing attacks but also their possible/potential variants. The pollination between different APTs in the generated GoPS helps to explore these new variants. Indeed, the next possible techniques in GoPS do not simply contain information on related APTs but the sum of all those APTs and how different techniques in these APTs are related to each other.

C. Attack Hypothesis Generation

Since the GoPS is built using techniques seen in ST and the next possible techniques from the KB, each path $h \in$ GoPS is a sequence that represents an attack hypothesis that might lead to possible APT(s).

Definition 2 (Attack hypothesis). An attack hypothesis (h) is defined with the tuple $h :< h_s,\ h_a >$, where h_s is the sequence of techniques extracted from GoPS and h_a is a potential APT for the given sequence h_s.

Here, we extract all possible hypotheses (*i.e.,* sequences) and their potential APTs from the GoPS [43]. The process finds potential APTs relying on GoPS and the APT definitions provided by the KB. Thus, for each hypothesis sequence $h_s \in$ GoPS, we find the bijection of h_s in KB to extract the candidate APT match set $C(h)$.

Definition 3 (Candidate APT match set). The candidate APT match set for a hypothesis sequence h_s consists of all possible APTs extracted from the KB that share a common sequence and at least one edge with h_s's last node under a mapping function f.

Let v be the last node in h_s, and $f : n \rightarrow a$ be a mapping function from a node $n \in h_s$ to an APT a in the KB. The candidate APT match set $C(h)$ is defined as:

$$C(h_s) = \{a \in \text{KB} \mid \exists\ (v, \cdot)(\cdot, \cdot)^*(\cdot, a) \in \text{KB}, (v, f(a)) \in \text{KB}\}$$

Example: For the sake of simplicity, Fig. 3 shows an example of one attack hypothesis sequence (h_s)

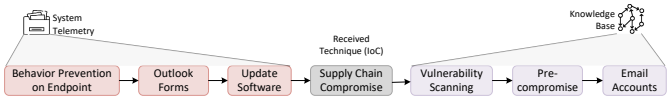


Fig. 3: Example of an APT41 hypothesis sequence generated for "`Supply Chain Compromise`" technique using search window of 3 leading to 129 possible APTs.

generated after the observation of e_0: "`Supply Chain Compromise`" using a search window of 3. The hypothesis sequence h_s is [`Behavior Prevention on Endpoint` → `Outlook Forms` → `Update Software` → `Vulnerability Scanning` → `Pre-compromise` → `Email Accounts`], where [`Behavior Prevention on Endpoint` → `Outlook Forms` → `Update Software`] happened before "`Supply Chain Compromise`" and is recorded in the ST, while the [`Vulnerability Scanning` → `Pre-compromise` → `Email Accounts`] is extracted from the KB. The candidate APT match set $C(h_s)$ is 129 possible APTs. In addition, the technique e_0: "`Supply Chain Compromise`" is part of APT41 and the sequence that appears just after e_0 is part of APT41 attack kill-chain, which makes h_s a potential hypothesis for APT41. Note that the hypothesis sequence h_s should not explicitly exist in the KB as a definition of an APT. However, the sub-sequence just after e_0 does.

D. Attack Hypothesis Examination

The hypothesis examination tends to examine each hypothesis h generated by the HG (Step 6a in Fig. 1) in order to provide the SOC team with a fewer yet relevant list of possible APTs for a given security technique. We study the importance of techniques constituting the hypothesis sequence h_s in order to define the overall importance of the hypothesis. Indeed, to address the scarcity and incompleteness of techniques in ST/KB due to the stealthiness of attacks, we calculate the success score using each technique frequency in an attack and the importance of a successful attack through each technique in the hypothesis. Fig. 4 illustrates a running example of how hypotheses are evaluated in order to generate relevant threat hypotheses and their variants.

1) Matching-based Evaluation: (Step 6a in Fig. 1) The objective of the matching-based evaluation is to find a hypothesis that might happen due to the similarity between sequence (h_s) and an APT definition (a_s). Hence, we apply similarity-based correlation [44]. In doing so, we calculate the similarity score for each hypothesis sequence h_s in GoPS and the associated APT instances extracted from the candidate APT match set $h_a \forall a \in C(h_s)$.

Definition 4 (Hypothesis similarity score). The hypothesis similarity score ($\rho_{h,a}$) quantifies how similar the sequence (h_s) is to a given APT sequence (a_s). The score is calculated by measuring the statistical relationship (*i.e.,* association, strength, similarity) and providing information about the magnitude of the relationship and its direction (*i.e.,* positive or negative) [45].

For a given hypothesis h and an APT a, the resulting similarity score ($\rho_{h,a}$) can be used to find the correlation between a hypothesis and the APT instance. Appendix B provides an empirical study of different similarity-based functions that can be used in our work. Based on our study, we found out that Cosine similarity performs better in the hypothesis evaluation context.

For the sake of illustration, the Cosine similarity [46] measures similarity between two non-zero vectors (*e.g.,* hypothesis sequence h_s and APT sequence a_s) in a multi-dimensional space, as shown in Eq. (1):

$$\rho_{h,a} = \cos(\theta) = \frac{(h_s \cdot a_s)}{(||h_s||\ ||a_s||)} \tag{1}$$

where:

- $\cos(\theta)$ is the angle between the two vectors,
- $(h_s.a_s)$ is dot product between sequences h_s and a_s and calculated as: $h_s \cdot a_s = h_s^T a_s = \sum_{i=1}^{n} h_{s_i} a_{s_i} = h_{s_1} a_{s_1} + \cdots + h_{s_n} a_{s_n}$,
- $||h||$ and $||a||$ represent the L2 norm or magnitude of the vector h_s and a_s, respectively, which is calculated as: $||h_s|| = \sqrt{h_{s_1}^2 + \cdots + h_{s_1}^n}$ and $||a_s|| = \sqrt{a_{s_1}^2 + \cdots + a_{s_1}^n}$

The similarity score value ranges from $+1$ to -1. A value of $+1$ indicates a perfect positive similarity (h_s and a_s are very similar), -1 indicates a perfect negative similarity (h_s and a_s are very dissimilar), 0 indicates no similarity, and an in-between value indicates intermediate similarity/dissimilarity.

Since the similarity score is calculated for all possible hypotheses and candidate APT match set, we generate the similarity score matrix that will be used later for pruning, as shown in Eq. (2):

$$\begin{array}{c} \\ h_1 \\ \vdots \\ h_n \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & & \cdots & a_m \end{array} \\ \begin{bmatrix} \rho_{h_{s_1},a_{s_1}} & \cdots & \rho_{h_{s_1},a_{s_m}} \\ \vdots & \ddots & \vdots \\ \rho_{h_{s_n},a_{s_1}} & \cdots & \rho_{h_{s_n},a_{s_m}} \end{bmatrix} \end{array}, \forall h_i \in \text{GoPS}, \forall a_j \in C(h_i) \tag{2}$$

2) Hypothesis Evaluation: The hypothesis evaluation aims at evaluating hypotheses with high-similarity scores with known attacks defined in the KB in order to find the most relevant hypotheses by applying the following evaluation.

a) Success Evaluation: (Step 7a in Fig. 1) Vulnerability metrics frameworks such as Common Vulnerability Scoring System (CVSS) [47] can be harnessed to extract the severity/score of events and consequently the success probability for the entire hypothesis. However, since we use a KB that includes different TTPs and APTs definitions, not all techniques in our KB could be associated with a vulnerability in CVSS (*e.g.,* CVE – Common Vulnerabilities and Exposures). This makes the use of CVSS as a success score calculator imperfect for our approach. To address this issue, we were inspired by graph theory [48] the concepts of frequency and importance. Thus, we consider a deterministic method providing a definition of the technique's success based on its appearance frequency and importance in the achievement of the attack (these values are not provided by security tools such as SIEM solution). Indeed, this deterministic method should be able to distinguish between different techniques that

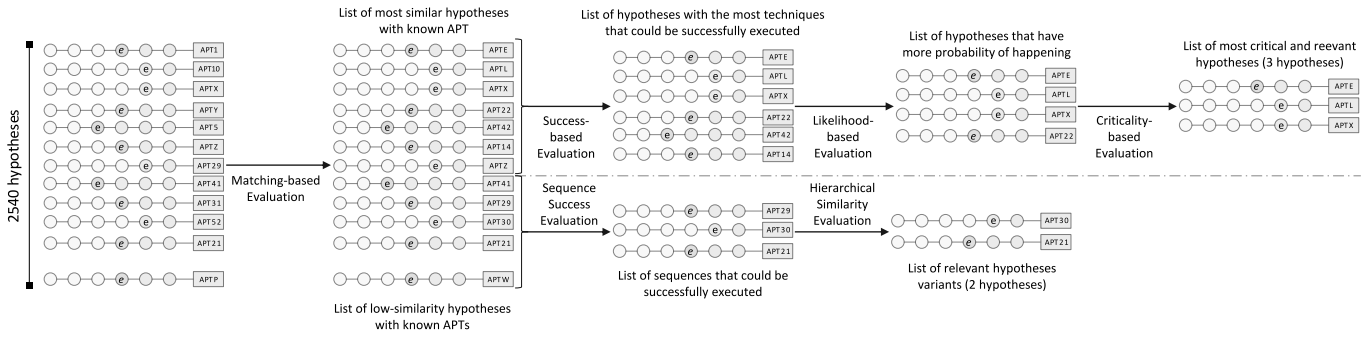


Fig. 4: An illustrative example of hypotheses examination to generate relevant threat hypotheses and their variants.

constitute the hypotheses after the segregation by the GoPS, and thus determine the overall likelihood for the success of a hypothesis. Hence, for each technique ($e \in h_s$), we define the success score (α) based on two values: the frequency of presence e in an attack a, ($\beta(e,a)$), and the importance of a successful attack a through the technique e, ($\delta(a,e)$).

(i) Measuring β: Given that the same technique e can be used by multiple attacks, $\beta(e,a)$ represents the frequency that the technique e is present in the attack a and is defined as the number of attacks that could be conducted via the said technique e in the GoPS divided by the number of all attacks in the entire GoPS, as shown in Eq (3).

$$\beta(e,a) = \frac{\text{No. of attacks via } e \text{ in GoPS}}{\text{No. of all attacks in GoPS}} \tag{3}$$

The motivation behind using the frequency value [48] is that techniques with the highest frequencies (*i.e.,* degrees) are often important entries/steps in the attack kill chain and play a critical role in the overall attack success.

(ii) Measuring δ: We measure the importance of an attack a by $\delta(a,e)$. δ is defined as the median of importance for all attacks executed through the technique e. The importance of an attack a_i is measured using $\gamma(e,a_i)$ and represents the attack progression (*e.g.,* advancement). $\delta(a,e)$ is then shown in Eq. (4).

$$\delta(a,e) = \text{median}\Big(\gamma(e,a_i)\ \forall a_i \in A_e\Big) \tag{4}$$

where A_e is the list of all attacks that could be executed via e, and $\gamma(e,a)$ defines the attack advance and is calculated using a graph centrality function.

(iii) Measuring γ: The success of an attack a is based on the importance/influence of the constituting techniques (*i.e.,* spread of an attack). Appendix C provides an empirical study of different centrality functions that could be applied in this work. Based on our study, we found out that the page-rank algorithm performs better in the hypothesis evaluation context. For the sake of illustration, the page-rank algorithm [49] measures the importance of a node based on the number and quality of incoming relations, as shown in Eq. (5).

$$\gamma(e,a) = \frac{(1-d)}{N} + d * \sum_{e_i \in N(e)} \frac{\gamma(e_i,a)}{\text{outdegree}(e_i)} \tag{5}$$

where d defines the damping factor (value between 0 and 1 – recommend to be set to 0.85 [49]) and represents the probability of transiting to a specific node and not a random node, and $N(e)$ list of neighbor nodes of e.

(iv) Measuring α: The success score of a technique $\alpha(e)$ – shown in Eq. (6), is then calculated as the product of the frequency of presence in an attack a and the importance of a successful attack through e:

$$\alpha(e,a) = \beta(e,a) * \delta(a,e) \tag{6}$$

Finally, the success of hypothesis h is then calculated as the median of success score of all techniques constituting the hypothesis sequence (*e.g.,* $\alpha(h,a) = \text{median}(\alpha(e_i,a),\ \forall e_i \in h_s)$). The use of the median helps in providing a relevant success score since it is not affected by outliers.

b) Likelihood Evaluation: (Step 7b in Fig. 1) The likelihood-based evaluation tends to calculate the attack existence likelihood. Note that we are not predicting the attack nor the attacker's next step, but rather calculating the likelihood of a technique based on how many times it occurred in the ST and does exist in the KB. Given that the technique e_0 happened, we apply the softmax function [50] on both past identified techniques and possible next techniques to calculate the conditional probability of the hypothesis h given e_0, ($P(h|e_0)$ – the conditional probability of h to occur given that e_0 happened now, how often it happened in the past, and how many times it appears in defined attacks). The reason behind using softmax [50] rather than Bayes' theorem [51], is that the required values to calculate the probabilities are already provided by previous evaluators and are applicable to our context without further computations. Indeed, conditional probability helps in inferring the most likely technique to occur given e_0, as shown in Eq. (7):

$$\begin{aligned} P(h|e_0) &= \text{softmax}(e_0, \mathsf{P}_{e_0,i}) * \text{softmax}(e_0, \mathsf{F}_{e_0,j}) \\ &= \frac{e^{(\rho(e_0,\mathsf{P}_{e_0,i}))}}{\sum_{n \in N_s^r(e_0)} e^{(\rho(n,\mathsf{P}_{e_0,i}))}} * \frac{e^{(\rho(e_0,\mathsf{F}_{e_0,j}))}}{\sum_{n \in N_s^r(e_0)} e^{(\rho(\mathsf{F}_{e_0,j},n))}} \end{aligned} \tag{7}$$

where:

- $\mathsf{P}_{e_0,i}$ and $\mathsf{F}_{e_0,j}$ are the past identified and possible next techniques, respectively, constituting the hypothesis sequence h_s for the received technique e_0,
- ρ is the similarity score value calculated by the matching-based evaluator,

- $N_s^r(e_o)$ is the list of neighbor nodes of e_0 with depth r using searching strategy s.

c) Criticality Evaluation: (Step 7c in Fig. 1) The criticality score of a hypothesis ($\mathrm{X}(h)$) is obtained using the assessment approach (*i.e.,* criticality rating) as defined in graph theory [52]. The criticality score is calculated by multiplying the success score by the attack existence probability, as shown in Eq. (8):

$$\mathrm{X}(h) = \alpha(h, a) * P(h|e_0) \tag{8}$$

The output of the Hypothesis Evaluation is the most relevant and probable hypotheses for ongoing attacks or attacks that could happen.

3) Variant Evaluation: The variant evaluation tends to generate potential variants for APTs that might be taking place in the system. Since the hypothesis evaluator prunes hypotheses by investigating only those with high-similarity scores with known APTs, the variant evaluator revises and scrutinizes those with low-similarity scores since they might be potential APT variants. The intuition behind this is that an APT variant by definition may present low similarity with the previous attacks (due to changes in the attacker's behavior) though it would be possible to occur (maintain the same objective). The variant evaluator evaluates the success of possible hypotheses, aligns their sequences, and then applies hierarchical similarity to assess their potential.

a) Sequence Success Evaluation: (Steps 9a-9b in Fig. 1) Similar to success evaluation, the sequence success evaluation evaluates the success score of the sequence h_s to be part of the APT a, where a is not included in candidate APT match $C(h_s)$.

Definition 5 (Sequence success score). Given a sequence h_s constituted of n techniques and a candidate APT a with a sequence a_s, the sequence success score $\alpha(h_s, a_s)$ is the sum of all success scores of constituting techniques.

$$\alpha(h_s, a_s) = \frac{\sum_{e_i \in h_s} \alpha(e_i, a_s)}{n}, \quad a \notin C(h_s)$$

The success score helps in determining if the sequence is an outlier or a typical sequence that contributes to the success of the APT a.

b) Sequence Alignment: (Step 10 in Fig. 1) Sequence alignment [53] aims to compare two sequences by aligning (*e.g.,* adding, removing, substituting nodes) one of them and finding a path that maximizes their similarity score. Given a sequence of a hypothesis (h_s) and a sequence for an APT (a_s), the objective is to align all possible and available sequences ($h'_s \in$ GoPS) to match the attack sequence (a_s), where h' is part of the hypotheses sent by the matching-based evaluator. The existence of the resulting aligned sequence is checked in the list of possible hypotheses. If they exist and are related to a known attack defined in the KB, they are considered to be possible variants.

Fig. 5 shows an illustrative example of sequence alignment. Each technique $e \in h_s$ could either be substituted by another technique e' or a set of techniques $\{e_1, \ldots, e_n\}$ given that those techniques already exist in the list of possible hypotheses

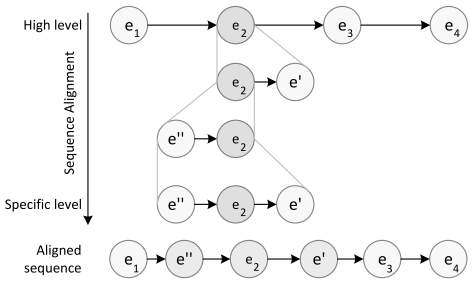


Fig. 5: Example of sequence alignment that substitute e_2 by $\{e'', e_2, e'\}$.

provided by the matching-based evaluator. We further assign a weight value to each edge that connects two techniques in h_s based on influence, importance, or success score.

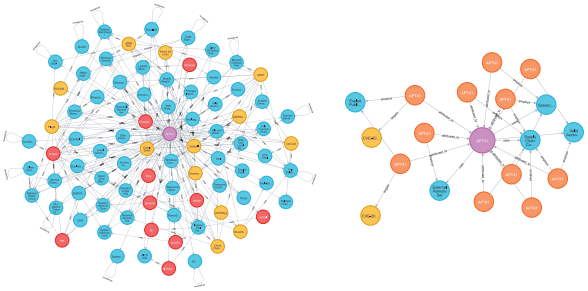
c) Hierarchical Similarity Approach: (Step 11 in Fig. 1) A sequence of an APT variant (h_s) might have a very low-similarity score with regards to the APT definition yet it could have a higher success score. Following the unified theory model [54], we apply a hierarchical similarity approach to revise and refine the matching process multiple times. Indeed, we focus on the features (*e.g.,* important nodes in the attack kill chain, critical malware/tools) and variations of the sequences (*e.g.,* alternative attack steps) that are most relevant to the variants in relation to the APT in question. For example, compromising the DNS server in APT41 is an important technique. Attackers might use different techniques (that are not defined in APT41) to reach this stage of the attack (which we consider an APT41 variant). In doing so, we compare the sequence (h_s) to multiple layers of the KB by iterating over sequence alignment and similarly scores.

As shown in Fig. 5, the process starts with a high-level sequence that captures the general sequence/patterns of the APT in question. If the sequence does not match the high-level model, we refine the model by adding, substituting, or removing more general/specific techniques (events/patterns) that capture the variations within the APT in question. For instance, technique e_2 can be achieved by combining it with e' or executing e'' prior it. After a few alignments, the high-level technique e_2 can be substituted by a specific sequence of technique $\{e'', e_2, e'\}$. For each alignment operation, we calculate the similarity score until we reach the specific level (*i.e.,* hierarchical approach). Note that we do not add random techniques; instead, we re-use the established techniques from the GoPS (since it has high pollination between different APTs) that have been omitted during the matching evaluation. These refinements are based on feedback sent by the matching-based evaluator that is captured in the KB as well. The process is repeated utill we reach a pre-defined threshold that can identify whether a sequence is a potential variant or not.

The output of the Variant Evaluation is the most relevant variant that could lead to an attack.

V. Implementation

(1) Implementation Environment: We implemented AUTOMA using Python 3 programming language. We performed



(a) From MITRE ATT&CK KB [39]. (b) From real telemetry [16]

Fig. 6: Graphical representation of APT41 [11]: (a) as per MITRE ATT&CK definition [39], (b) techniques related to APT41's campaigns as they occurred in real dataset [16], (group name: purple, techniques: blue, software: yellow, tools: red, campaign: orange).

our experiments on Intel Xeon E312xx, 6 vCPU @ 2.69Ghz, with 32GB RAM running Ubuntu 20.04. We used Neo4j 4.4.11 running on top of Docker 23.0.1 to manage ST and KB. In addition, we built a GUI tool for threat hypotheses and variants generation (see Appendix E). To map nodes in GoPS into an embedding space, we used NODE2VEC algorithm [55]. NODE2VEC generates embedding space of lower dimensions than the number of nodes in GoPS while preserving the initial structure by mapping similar nodes to similar embedding in the embedding space.

(2) Knowledge Base: We used the CTI KB from the MITRE ATT&CK framework [38]. The KB populates all the MITRE ATT&CK techniques [39] and generates a graph database hosted on Neo4j graph database management system[5]. We used the open-source JSON file from the MITRE CTI[6] and parsed all nodes (attack-pattern, campaign, intrusion-set, tools, etc.) and their relationships into a graph database. The CTI database contains up to 2060 nodes, 27411 relationships, and 366 APTs conducted using up to 861 unique techniques.

(3) Real Telemetry dataset: To validate the efficiency of our solution, we used a real dataset [16]. The dataset covers 86 APTs and 350 campaigns from 2008 to 2020. It includes information about attack vectors, exploited vulnerabilities, and affected software and their versions, formed in 7150 nodes, 79118 relationships, and 170 unique techniques.

Figs. 6(a) and 6(b) illustrate a graphical representation of APT41 from the MITRE ATT&CK KB [39] and its associated campaigns from ST [16], respectively. APT41 extracted from the KB is defined with 59 techniques, 13 software, and 11 tools, while its 11 campaigns extracted from the ST have 5 techniques and 2 tools (*e.g.,* vulnerabilities). As shown in Fig. 6(a), the AP41 definition is much larger in our KB than its representation in the ST as the attack did not fully execute.

[5]Neo4j Graph Data Platform: www.neo4j.com
[6]MITRE ATT&CK CTI: www.github.com/mitre/cti/

VI. Evaluation

In this section, we evaluate the performance of AUTOMA by measuring its *effectiveness* and *efficiency* in terms of generating relevant attack hypotheses and their variants.

- *Effectiveness*: the effectiveness of AUTOMA refers to the relevance of the generated attack hypotheses and variants. A hypothesis should represent a relevant threat taking (or will take) place. Effectiveness can be determined by calculating *the ranking* of the relevant hypothesis among the final list produced by AUTOMA and *the number of generated hypotheses*.
- *Efficiency*: the efficiency of AUTOMA refers to the ability to identify actual threats in a timely and efficient manner. In this work, we measure the efficiency in terms of the *hypotheses reduction rate* and the *execution time*.

It is important to highlight that existing threat hunting solutions rely on the security expert intervention to manually generate hypotheses and, to the best of the authors' knowledge, there is no algorithm in the literature that automates the process that we can use as a baseline to compare with. Thus, and to further gauge the efficiency of AUTOMA, we used a naive (graph mining algorithm) executed by the HG agent using the same KB and ST that extract all possible hypotheses. We also used the list of attacks/campaigns from the dataset [16] as our ground truth.

A. Defining Search Parameter

As mentioned in Section IV, AUTOMA is built on top of KB and ST. Regardless of the used search strategy to extract GoPS, the search window (r) has a direct impact on the overall solution performance. The larger the search window is, the more comprehensive GoPS is built, and hence more information can be extracted about potential attacks. This helps in generating more relevant hypotheses, yet it would lead to an exponential number of generated hypotheses by the HG agent. For the same technique (e), we found that the number of extracted possible hypotheses from GoPS increases exponentially when we expand the search window. Table I shows the number of extracted nodes in GoPS and the number of generated possible hypotheses in function of the search window (r) for the same security technique ("`External Remote Services`") and the same search strategy (depth-first search). We showcase in appendix D that the best value for r is 3, which is used in the rest of the evaluation. Using the GoPS, the naive algorithm generates up to 67229 possible hypotheses for "`External Remote Services`". It is too costly (*i.e.,* time and manpower) to investigate this enormous number of hypotheses generated for only one security technique.

In the rest of the evaluation, we use the cosine as the similarity function and page rank as the centrality function. The rationale behind this section is explained in appendixes B and C, respectively.

B. APT41: Evaluation Scenario

In the evaluation scenario, we focus on APT41 [11] as it represents a complex APT attack conducted in 59 TTPs (steps)

TABLE I: Number of nodes in GoPS and possible hypotheses in function of search window (for "`External Remote Services`" using depth-first search strategy).

Search window (r)	No. of nodes in GoPS	No. of possible hypotheses
3	517	67229
4	700	518407
5	800	31845719

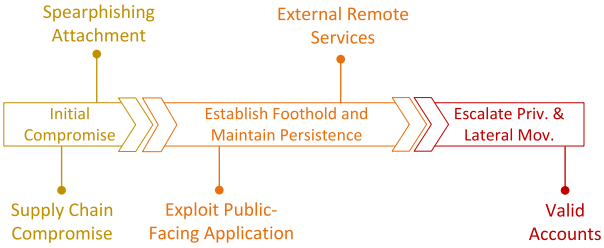


Fig. 7: Illustration of APT41 attack phases and selected techniques used for the evaluation [11].

through several systems and has potential variants [12]. As shown in Fig. 7, APT41 can be conducted in three stages: (i) Initial Compromise, (ii) Establish Foothold and Maintain Persistence, and (iii) Escalate Privileges and Lateral Movement. Each stage is mapped into multiple MITRE ATT&CK techniques, while the entire attack comprises 59 techniques. The telemetry dataset [16] contains 11 campaigns for APT41 between 2014 to 2022 (see Fig. 6(b)). Those campaigns use 5 unique techniques distributed on the three aforementioned stages, which we will use in the rest of the evaluation as different attacks progress:

(1) "`Supply Chain Compromise`" (SCC): is part of the initial access, where the attacker utilizes manipulation of products or product delivery systems to compromise data before reaching end-users.
(2) "`Spearphishing Attachment`" (SA): is part of the initial access, where the attacker can execute the attack at any phase of the supply chain, *e.g.,* development tools, source code repository, automatic software update system, and shipment interdiction.
(3) "`Exploit Public-Facing Application`" (EPFA): is part of the initial access, where the attacker mostly focuses on adding malicious code to the software directly in software distribution channels.
(4) "`External Remote Services`" (ERS): is part of the persistence, where the attacker can carry out a targeted attack against a group of victims or carry out a universal attack on a larger group of victims before performing additional on a targeted group of victims.
(5) "`Valid Accounts`" (VA): is part of privilege escalation, where open source dependencies are likely to be targeted to include malicious code as a dependency on the application.

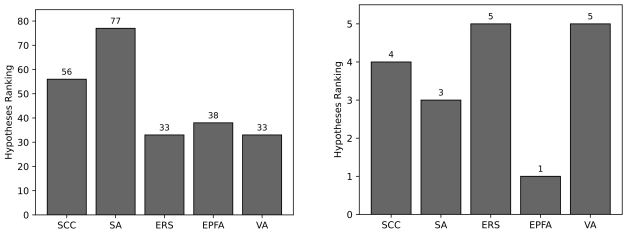
In the following, we show the results of those 5 techniques used in the 11 APT41 campaigns.

C. *Effectiveness of* AUTOMA

1) *Effectiveness of Hypothesis Generation:*

a) Hypothesis Ranking: Fig. 8 shows the ranking of the relevant hypothesis by AUTOMA. In this evaluation, we focused on two cases:

- Search window of 0 (*i.e.,* using only the received technique e_0): we calculated the ranking of the relevant hypothesis among the generated hypotheses by AUTOMA using only the received technique e_0 (GoPS has only e_0 as past identified techniques). As shown in Fig. 8(a), the relevant hypothesis upon receiving "`Supply Chain Compromise`" as e_0 is ranked 56, and for "`Exploit Public-Facing Application`" as e_0 is 38. We note that the average ranking of the relevant hypothesis for the 5 selected techniques is 47. This is due to two reasons: (i) we used only e_0 from ST, which means we do not know enough information about previous attacker activities, and (ii) the similarity function, due to its nature, has a tendency toward those APTs with short attack kill-chain (biases to short APTs with similar sequences), which leads to ranking them on top of the list of possible APTs while long attack kill-chain will have less similarity score (*e.g.,* APT41 has 59 techniques while APT12 has 5 techniques). We consider these results logical, as the scarcity of the available information (only one technique e_0) leads to many possible hypotheses.
- Search window of 3 (using three past identified techniques before e_0): we calculate the ranking of the relevant hypothesis among the generated hypotheses by AUTOMA using the received technique e_0 and three past identified techniques prior to e_0 (3 techniques prior e_0 as past identified techniques). As shown in Fig. 8(b), the relevant hypothesis for "`Supply Chain Compromise`" is ranked 4, and for "`Exploit Public-Facing Application`" is 1. We note that the average ranking of the relevant hypothesis for the 5 selected techniques is 3. Indeed, adding only three extra techniques to the GoPS helps in improving the ranking of relevant hypotheses from 47 to 3. This is due to the fact that the more we know about previous activities, the better ranking AUTOMA can perform. This shows that AUTOMA improves its effectiveness by up to 93%.



(a) Hypothesis ranking using only the received technique (search window of 0).
(b) Hypothesis ranking using 3 past identified techniques (search window of 3).

Fig. 8: Hypothesis ranking for APT41 (using cosine similarity function and page-rank centrality).

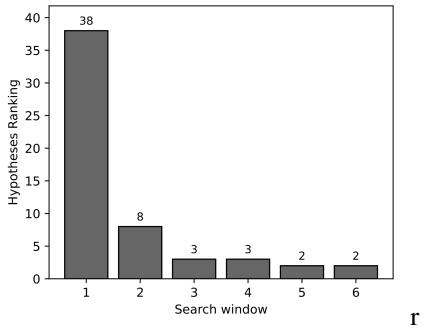


Fig. 9: Ranking position of relevant hypothesis by AUTOMA in function of search window.

This result is further proved by Fig. 9. The figure shows the average ranking for the 5 techniques used in APT41's campaigns in function of the search window. Starting from $r = 3$, the relevant hypothesis is always ranked among the top 3 hypotheses. This result leads to an average improvement of 99%. Note that we assumed that we start the hypothesis generation only from one technique (*i.e.,* IoC). Therefore, we assumed that SoC did not connect the three techniques together (see sections IV-B and IV-C). Overall, extracting enough information from the ST, connecting it with KB, and using the criticality evaluation help in tackling the biases of similarity function by performing better pruning at the HE level.

b) Number of generated hypotheses: Table II presents the number of generated hypotheses for each of the five unique techniques used in the 11 APT41's campaigns. These results are for the case of a search window of 3. We notice that for the technique "`Supply Chain Compromise`", the naive algorithm generates up to two thousand possible hypotheses. However, for techniques at the end of the attack kill chain such as "`Valid Accounts`", the naive algorithm generates up to two million hypotheses [1]. This is due to (i) the fact that "`Supply Chain Compromise`" is more commonly used by many APTs at different stages, leading to an increase of possible hypotheses and (ii) the pollination between different APTs explores not previously considered possibilities by combining all possible paths in GoPs. Further, AUTOMA performs efficient pruning by minimizing the number of generated hypotheses. For instance, out of two thousand possible hypotheses for "`Supply Chain Compromise`" generated by the naive algorithm, AUTOMA reduces the number to only 947 possible hypotheses. Similarly, for "`Valid Accounts`", AUTOMA diminishes only 17366 possible hypotheses compared with two million hypotheses generated by the naive algorithm. This is due to the fact that (i) AUTOMA considers different factors including attack success, advance, and its criticality and (ii) the more we know about attacks and previous activities, the better pruning AUTOMA can perform. On top of this successful pruning, AUTOMA maintains an effective ranking of relevant hypotheses.

2) Effectiveness of Variant Generation:

a) Variant Validity and Ranking: In this experiment, we use the list of hypotheses generated using a search window of 3 to calculate the ranking of potential variants. We also set the similarity threshold to 0.5, which is widely used in the literature. This value is defined by the security expert based on the level of granularity of the desired variants.

TABLE II: Number of generated APT41 hypothesis for different techniques (using the cosine similarity function, page-rank centrality, and search window of 3.

Received Technique	Number of Hypotheses		Ranking of Relevant Hypothesis
	Naïve	AUTOMA	
SCC	2540	947	4
SA	371219	51262	3
ERS	67229	978	5
EPFA	1116155	2592	1
VA	2328938	17366	5

TABLE III: Ranking of generated variants for APT41.

Technique	Initial Hypotheses	Ranking of generated variants
SCC	1593	4
SA	319957	6
EPFA	1115177	6
ERS	67229	4
VA	2311572	4

AUTOMA generated 6 attack variants for APT41. Empirically, we have verified those 6 generated variants and found they are possible variants for APT41. Indeed, these variants are generated using knowledge captured in the KB. Therefore, as long as the knowledge in the KB is exact, the variants generated using knowledge discovery techniques would be real too (see section IV-B). However, not any variant generated is likely to happen. We further then estimated the likelihood of those variants ranking them. Table III presents the ranking of the relevant variants of APT41 compared to the overall number of received hypotheses sent by the matching-based evaluator. For instance, AUTOMA's VE agent received 1593 hypotheses for "`Supply Chain Compromise`" out of which it finds 4 potential variants. Fig. 10 illustrates an example of the initial APT41 hypothesis' sequence (`Supply Chain Compromise` $\rightarrow$ `DNS`) and 4 potential variants for that sequence that might lead to APT41.

Using the sequence alignment, the attacker might amend their techniques by adding or substituting a few techniques to reach the objective (*e.g.,* compromise the DNS server). For instance, the attacker shifted to very specific techniques to compromise the system by (a) performing active scanning to find any breach (*e.g.,* variants 1 and 2) to compromise the system, (b) compromising software dependencies and tools (*e.g.,* variant 3), or (c) combining both of them (*e.g.,* variant 4). Similarly, and due to the efficient sequence alignment, AUTOMA finds six possible variants of APT41 after receiving the 1115177 hypotheses of "`Exploit Public-Facing Application`".

For this experiment, instead of sending an average number

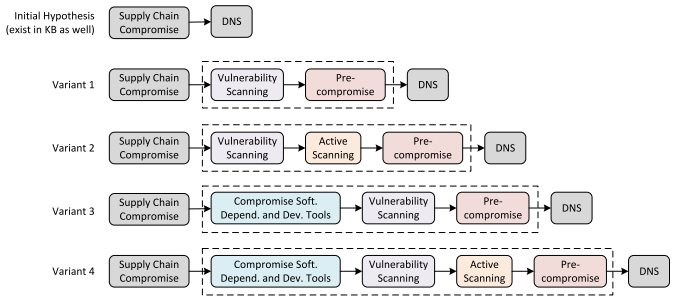


Fig. 10: List of potential variants for an APT41 hypothesis (dashed sequences are not part of APT41 definition but exist in KB).

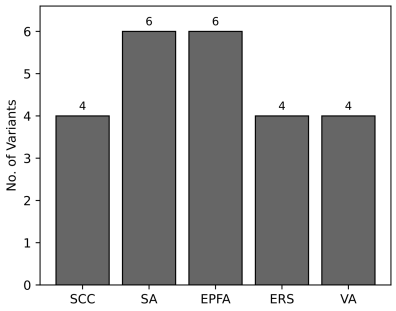


Fig. 11: Number of generated variants.

of 763105 variants to the SOC team for investigation (which requires a lot of time and effort), AUTOMA notifies the SOC team with an average of four variants, which improves its effectiveness by up to 99% and also fewer time and efforts required by SOC for investigation.

b) Number of generated variants: Fig. 11 shows the number of generated variants per technique for APT41. Due to the sequence alignment that scrutinizes each possible sequence and the hierarchical similarity approach that follows the unified theory model, AUTOMA is able to find potential variants for APT41. For instance, AUTOMA finds 4 possible variants when receiving the "`Supply Chain Compromise`" and 6 potential variants for "`Exploit Public-Facing Application`". Those variants have the same objective as APT41 but different attack sequences that an attacker might execute according to what is defined for APT41 in the KB. Other hypotheses (see Table III) have been eliminated since they have a lower similarity score with APT41 and either a lower success and/or no alignments for APT41. The SOC team will investigate those small number of potential variants which in return helps in early attack detection.

D. *Efficiency of* AUTOMA

1) *Efficiency of Hypothesis Generation:*

a) Reduction rate: The reduction rate is calculated as the decreased percentage of the number of hypotheses generated by AUTOMA in relation to the original number generated by the naive algorithm. Regardless of the technique and the attack advance, AUTOMA is able to reduce the number of generated hypotheses by an average of 89% in relation to the original

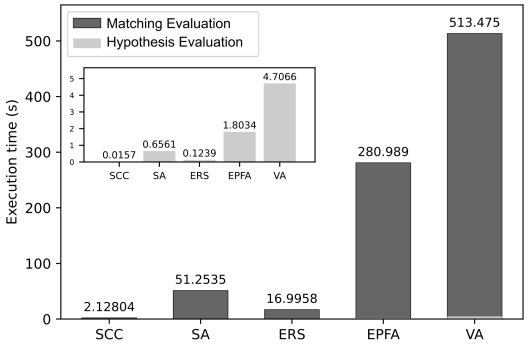


Fig. 12: Execution time for matching and hypothesis evaluators (using cosine similarity function, page-rank centrality, and search window of 3).

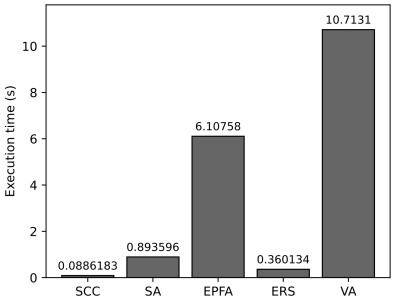


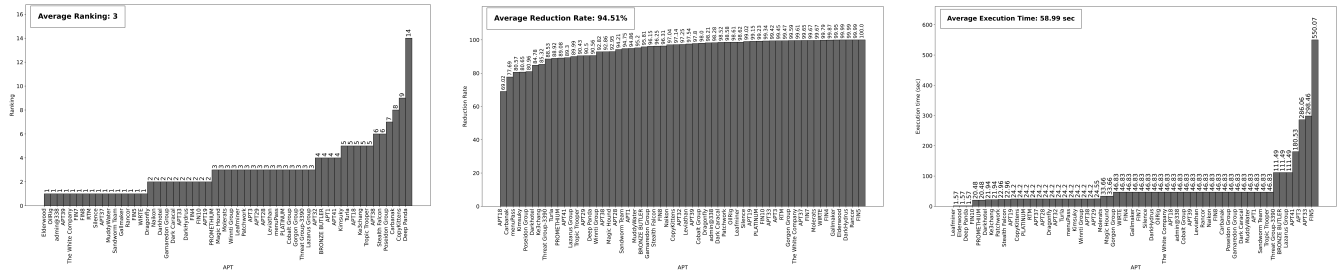
Fig. 13: Execution time for variant evaluation.

number of hypotheses while ranking the relevant hypotheses at the top of the list sent to the SOC team.

b) Execution time: We measure the execution time as the required time by AUTOMA to evaluate all possible hypotheses and produce the final list of hypotheses. As shown in Fig. 12, the overall execution time varies from one technique to another. For instance, evaluating the 2 thousand hypotheses generated for "`Supply Chain Compromise`" requires up to 2 seconds, while evaluating the 2 million hypotheses for "`Valid Accounts`" takes up to 8 minutes. Moreover, we notice that the matching-based evaluator takes a larger execution time compared with the hypotheses evaluator since it has to check each path in GoPS and calculate its similarity with possible attacks in KB. The larger GoPS and KB are, the more execution time is required to perform similarity score calculation. In contrast, the hypothesis evaluator has a very small execution time since it performs only a linear computation with reduced overhead. Finally, it is important to mention that AUTOMA is executed as an offline defense line, which means the execution time will not impact the overall hunting performance. Overall, the execution time depends on how often a technique appears in past activities and in different attacks in KB.

2) *Efficiency of Variant Evaluation:*

a) Reduction rate: As shown in Fig. 11 and Table III, AUTOMA generates a small number of variants compared to the vast number of potential hypotheses received from the matching evaluator. This results in to an average reduction rate of 97% due to the efficiency of sequence alignment and



(a) Average hypothesis ranking. (b) Average reduction rate. (c) Average execution time.

Fig. 14: AUTOMA's results for 57 APTs with 284 attack campaigns using real dataset [16].

hierarchical similarity approach.

b) Execution time: Fig. 13 shows the execution time for variants generation per technique. Although the execution time depends on the number of hypotheses received from the matching evaluator, it remains very small for all tested techniques. For instance, AUTOMA needs 0.08 seconds to evaluate 1593 possible hypotheses for "`Supply Chain Compromise`", which leads to generating 4 potential variants. Similarly, 10 seconds were required to evaluate 2 million hypotheses for "`Valid Accounts`" with a result of 6 variants. Overall, our choice of hierarchical similarity (see section IV-D3c) is justified by the small execution time.

E. Validation with multiple APTs

To further validate the effectiveness and efficiency of AUTOMA (further confirm the findings in sections VI-C and VI-D), we validated it with 284 attack campaigns of 57 APTs (an average of 4 campaigns per APT) extracted from a real dataset [16].

Fig. 14 shows the average results for all APTs' attack campaigns for hypothesis ranking, reduction rate, and execution time. As mentioned earlier, AUTOMA is able to generate relevant hypotheses that will be sent to the SOC team. For instance, the majority of the relevant hypotheses are ranked within the top three (Fig. 14(a)), with an average relevant hypothesis ranking of 3 and a cumulative probability of 73%. This is due to the efficient evaluation performed by both matching and hypothesis evaluators. However, few hypotheses had a larger ranking, about 10% of relevant hypotheses are ranked above 5 (*e.g.,* CopyKittens, APT18, and Deep Panda) due to the fact that related APTs share the same similarity score and they have the same attack advance because of the nature of telemetry dataset [16].

In addition, we executed two loops: the first one for each APTs and the second one for each technique that constitutes the hypothesis in place. Still, AUTOMA is able to achieve an average reduction rate of 94.51% among all the evaluated APTs (Fig. 14(b)) – which drastically helps in addressing the alarm fatigue, and an average execution time of 1 minute (Fig. 14(c)). The longest execution time measured is 9 minutes for APT FIN5, which still does not have an impact on the overall performance since threat hunting is conducted as an offline defense line.

F. Discussion

We evaluated AUTOMA using a KB built from MITRE ATT&CK enterprise [39], which is widely used across the industry for classifying attack behaviors and understanding their life cycle. However, this does not limit the scope of the KB that could be used in AUTOMA. For instance, the SOC could integrate other MITRE frameworks, such as FiGHT (a 5G hierarchy of Threats) [56] or even augment existing KB using their definitions (*e.g.,* non publicly defined threat). Indeed, the attack behaviors need to be carefully mapped with the right technique. For instance, if a behavior is mapped to a broad technique (*e.g.,* lateral movement), it becomes challenging to distinguish the distinct threat actor, potentially leading to irrelevant ranking. Conversely, if mapped to a specific technique (*e.g.,* pass the hash), the threat infrastructure and steps might be missed. During the evaluation, we considered e_0 as only one technique, and then we changed the search window (r). However, AUTOMA can be extended to cover more than one technique as an IoC (*e.g.,* the appearance of 3 techniques in a row refers to one IoC), which in return grasp more specific APTs. Other design parameters (*e.g.,* similarity threshold) can be easily defined by the security expert based on the granularity of the desired attack variants they want to generate. Additionally, and due to the constant barrage of security threats and alerts received by the SOC team, AUTOMA helps in proactive generating attack hypotheses and variants of potential stealth attacks by using a unique observed security technique.

Finally, AUTOMA can be integrated as a proactive hunting tool into existing SOC defense lines and will not replace SIEM/SOAR. For instance, SOC can be selective on the list of interested APTs (targeted threat actors) and IoC/IoA that AUTOMA should proactively hunt based on the semantics of their attack kill chain.

VII. Conclusion & Future Work

In this paper, we considered the challenge of automating threat hunting by generating a set of relevant attack hypotheses and their variants to help SOC's security experts. We designed, AUTOMA, a solution that uses knowledge discovery by connecting ST with a KB to automate attack hypothesis and variants generation. For a received technique, we first generated all possible hypotheses and then applied multi-level evaluation to minimize the number of generated hypotheses.

Specifically, we applied matching-based evaluation to split hypotheses based on their similarities to attack sequences defined in the KB, and then employed hypothesis evaluation using the probability of presence, success, and conditional probability to preserve only the most relevant hypotheses. We further used sequence alignment and hierarchical similarity approach to revise hypotheses with low-similarity scores and identify potential attack variants. Extensive experiments, validated with real dataset and multiple APTs, showed that AUTOMA not only generates the relevant hypotheses, but our subsequent evaluation also provides a valuable ranking for the most likely and impactful attacks and their variants in the network. The next phase of our research involves integrating machine learning and machine reasoning to design an automated hypotheses testing solution, which takes as input the hypotheses generated by AUTOMA to further refine the relevant hypotheses. We believe this approach could lead to uncovering new attacks based on the KB.

ACKNOWLEDGMENT

The authors would like to thank Jan Willekens and Jesus Alatorre from Ericsson for their invaluable feedback.

REFERENCES

[1] "Technical Requirements for the ArcSight Platform," 2021, micro Focus ArcSight Platform.

[2] B. Gelman, S. Taoufiq *et al.*, "That Escalated Quickly: An ML Framework for Alert Prioritization," *arXiv preprint arXiv:2302.06648*, 2023.

[3] B. Nour, M. Pourzandi *et al.*, "A Survey on Threat Hunting in Enterprise Networks," *IEEE Communications Surveys and Tutorials*, pp. 1–27, 2023.

[4] "A framework for cyber threat hunting," *Tech. rep., Sqrrl Data, Inc.*, 2018. [Online]. Available: https://www.threathunting.net/files/framework-for-threat-hunting-whitepaper.pdf

[5] S. M. Milajerdi, R. Gjomemo *et al.*, "HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1137–1152.

[6] T. Madi, H. A. Alameddine *et al.*, "AutoGuard: A Dual Intelligence Proactive Anomaly Detection at Application-Layer in 5G Networks," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 715–735.

[7] A. Alsaheel, Y. Nan *et al.*, "ATLAS: A Sequence-based Learning Approach for Attack Investigation," in *USENIX Security Symposium (USENIX Security)*, 2021, pp. 3005–3022.

[8] S. Qamar, Z. Anwar *et al.*, "Data-driven analytics for cyber-threat intelligence and information sharing," *Computers & Security*, vol. 67, pp. 35–58, 2017.

[9] X. Shu, F. Araujo *et al.*, "Threat intelligence computing," in *ACM Conference on Computer and Communications Security (SIGSAC)*, 2018, pp. 1883–1898.

[10] R. Puzis, P. Zilberman *et al.*, "ATHAFI: Agile Threat Hunting And Forensic Investigation," *arXiv preprint arXiv:2003.03663*, 2020.

[11] "APT41: A Dual Espionage and Cyber Crime Operation," Mandiant, Tech. Rep., 2022.

[12] G. C. A. Team, "Threat Horizons - April 2023 Threat Horizons Report," Google Cloud, Tech. Rep., 2023. [Online]. Available: https://services.google.com/fh/files/blogs/gcat_threathorizons_full_apr2023.pdf

[13] I. Alam, K. Sharif *et al.*, "IoT virtualization: a survey of software definition & function virtualization techniques for internet of things," *arXiv preprint arXiv:1902.10910*, 2019.

[14] Z. Li, J. Zeng *et al.*, "AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports," pp. 589–609, 2022.

[15] "SIEM Capabilities through FireEye Helix," FireEye, Tech. Rep., 2019.

[16] G. Di Tizio, M. Armellini *et al.*, "Software updates strategies: A quantitative evaluation against advanced persistent threats," *IEEE Transactions on Software Engineering*, vol. 49, no. 3, pp. 1359–1373, 2023.

[17] K. Alsubhi, E. Al-Shaer *et al.*, "Alert prioritization in intrusion detection systems," in *IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2008, pp. 33–40.

[18] E. Pelofske, L. M. Liebrock *et al.*, "Cybersecurity Threat Hunting and Vulnerability Analysis Using a Neo4j Graph Database of Open Source Intelligence," *arXiv preprint arXiv:2301.12013*, 2023.

[19] M. Villarreal-Vasquez, G. M. Howard *et al.*, "Hunting for Insider Threats Using LSTM-based Anomaly Detection," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[20] F. Araujo, D. Kirat *et al.*, "Evidential Cyber Threat Hunting," *arXiv preprint arXiv:2104.10319*, 2021.

[21] D. B. Rawat, R. Doku *et al.*, "Cybersecurity in big data era: From securing big data to data-driven security," *IEEE Transactions on Services Computing*, 2019.

[22] V. S. Sree, C. S. Koganti *et al.*, "Artificial Intelligence Based Predictive Threat Hunting In The Field of Cyber Security," in *Global Conference for Advancement in Technology (GCAT)*. IEEE, 2021, pp. 1–6.

[23] A. Yazdinejad, M. Kazemi *et al.*, "An ensemble deep learning model for cyber threat hunting in industrial internet of things," *Digital Communications and Networks*, 2022.

[24] K. Pei, Z. Gu *et al.*, "HERCULE: Attack story reconstruction via community discovery on correlated log graph," in *Annual Conference on Computer Security Applications (ACSAC)*, 2016, pp. 583–595.

[25] K. Satvat, R. Gjomemo *et al.*, "EXTRACTOR: Extracting attack behavior from threat reports," in *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 598–615.

[26] A. Berady, M. Jaume *et al.*, "From TTP to IoC: Advanced persistent graphs for threat hunting," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1321–1333, 2021.

[27] P. Parameshwarappa, Z. Chen *et al.*, "Analyzing attack strategies against rule-based intrusion detection systems," in *Proceedings of the Workshop Program of the International Conference on Distributed Computing and Networking (ICDCN)*, 2018, pp. 1–4.

[28] G. Ho, M. Dhiman *et al.*, "Hopper: Modeling and Detecting Lateral Movement," in *USENIX Security Symposium (USENIX Security)*, 2021, pp. 3093–3110.

[29] P. Radoglou-Grammatikis, A. Liatifis *et al.*, "TRUSTY: A Solution for Threat Hunting Using Data Analysis in Critical Infrastructures," in *IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2021, pp. 485–490.

[30] A. B. Ajmal, M. A. Shah *et al.*, "Offensive security: Towards proactive threat hunting via adversary emulation," *IEEE Access*, vol. 9, pp. 126 023–126 033, 2021.

[31] K. Subramanian and W. Meng, "Threat Hunting Using Elastic Stack: An Evaluation," in *IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. IEEE, 2021, pp. 1–6.

[32] H. Almohannadi, I. Awan *et al.*, "Cyber threat intelligence from honeypot data using elasticsearch," in *IEEE International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2018, pp. 900–906.

[33] N. Hurley, Z. Cheng *et al.*, "Statistical attack detection," in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 149–156.

[34] B. Bhattarai and H. Huang, "SteinerLog: Prize Collecting the Audit Logs for Threat Hunting on Enterprise Network," in *ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, 2022, pp. 97–108.

[35] M. Ahmed, A. N. Mahmood *et al.*, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[36] I. Ghafir, M. Hammoudeh *et al.*, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Generation Computer Systems*, vol. 89, pp. 349–359, 2018.

[37] R. Kryukov, V. Zima *et al.*, "Mapping the Security Events to the MITRE ATT &CK Attack Patterns to Forecast Attack Propagation," in *International Workshop on Attacks and Defenses for Internet-of-Things*. Springer, 2022, pp. 165–176.

[38] B. Nour, M. Pourzandi *et al.*, "Accurify: Automated New Testflows Generation for Attack Variants in Threat Hunting," in *Foundations and Practice of Security (FPS)*, Dec 2023, pp. 1–18.

[39] B. E. Strom, A. Applebaum *et al.*, "MITRE ATT&CK: Design and philosophy," *Technical report*, 2020.

[40] F. K. Kaiser, U. Dardik *et al.*, "Attack Hypotheses Generation Based on Threat Intelligence Knowledge Graph," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[41] W. J. Frawley, G. Piatetsky-Shapiro *et al.*, "Knowledge discovery in databases: An overview," *AI magazine*, vol. 13, no. 3, pp. 57–57, 1992.

[42] H. Cai, V. W. Zheng *et al.*, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.
[43] L. Tang and H. Liu, "Graph mining applications to social network analysis," *Managing and mining graph data*, pp. 487–513, 2010.
[44] D. Conte, P. Foggia *et al.*, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence*, vol. 18, no. 03, pp. 265–298, 2004.
[45] L. A. Zager and G. C. Verghese, "Graph similarity scoring and matching," *Applied mathematics letters*, vol. 21, no. 1, pp. 86–94, 2008.
[46] P. Xia, L. Zhang *et al.*, "Learning similarity with cosine similarity ensemble," *Information sciences*, vol. 307, pp. 39–52, 2015.
[47] P. Mell, K. Scarfone *et al.*, "Common vulnerability scoring system," *IEEE Security & Privacy*, vol. 4, no. 6, pp. 85–89, 2006.
[48] C. Ducruet and J.-P. Rodrigue, "Graph theory: Measures and indices," *The geography of transport systems*, 2013.
[49] T. Haveliwala, "Efficient computation of PageRank," Stanford, Tech. Rep., 1999.
[50] J. Gonsior, C. Falkenberg *et al.*, "To Softmax, or not to Softmax: that is the question when applying Active Learning for Transformer Models," *arXiv preprint arXiv:2210.03005*, 2022.
[51] J. Joyce, "Bayes' theorem," 2003.
[52] D. Fensel, U. Simsek *et al.*, *Knowledge graphs*. Springer, 2020.
[53] M. Rautiainen and T. Marschall, "GraphAligner: rapid and versatile sequence-to-graph alignment," *Genome biology*, vol. 21, no. 1, p. 253, 2020.
[54] C. P. Kruskal and M. Snir, "A unified theory of interconnection network structure," *Theoretical Computer Science*, vol. 48, pp. 75–94, 1986.
[55] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.
[56] "FiGHT (5G Hierarchy of Threats)," 2023, the MITRE Corporation. [Online]. Available: https://fight.mitre.org/

APPENDIX A
GRAPH OF POSSIBLE STEPS

Fig. 15 shows a graphical representation of GoPS for the technique "`External Remote Services`" with a search window of 3 and depth-first search strategy, generated by the HG agent.

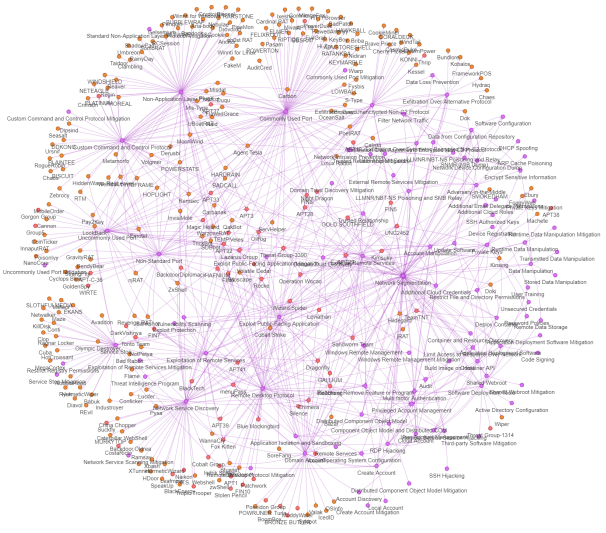


Fig. 15: Example of GoPS generated by the HG agent. (for "`External Remote Services`" using search window of 3 and depth-first search strategy - Techniques: purple, Group name: orange.).

GoPS is used to extract all possible hypotheses. For the sake of illustration, Table IV shows an example list of possible hypotheses based on GoPS shown in Fig. 2 consisting of different paths from past identified techniques and the possible next techniques. As we can see, the same attack a could be conducted using different sequences *e.g.*, $(h_a = \mathsf{P}_i \cdot \mathsf{F}_j, h'_a = \mathsf{P}_k \cdot \mathsf{F}_l)$. For instance, the two distinct sequences $(\mathsf{P}_1.e_0.\mathsf{F}_2)$ and $(\mathsf{P}_4.e_0.\mathsf{F}_8)$ lead to the same APT (a_2).

TABLE IV: Example of list of possible hypotheses with different sequences but the same APT (a_2).

Sequence	**Past Identified Techniques** (P)	**possible next techniques** (F)	**Hypothesis** (A)
h_1	P_1	F_1	$h_{a_1} \rightarrow a_1$
h_2	P_1	F_2	$h_{a_2} \rightarrow a_2$
...	...	...	...
h_{600}	P_4	F_8	$h_{a_600} \rightarrow a_2$

TABLE V: Qualitative comparison between different similarity functions/coefficients.

Similarity Coefficient	**Pros**	**Cons**
Pearson	• Standard measure of correlation • Works well with continuous variables and handles missing values • Sensitive to the scale of the variables	• Assumes a linear relationship between variables, which may not be true in some cases • Sensitive to outliers, which may affect similarity score • Requires variables to follow a normal distribution
Jaccard	• Easy to calculate and interpret • Useful for binary or categorical data • Works well with sparse data	• Omits the frequency or intensity of elements in the sets • May produce misleading results for sets of different sizes • May not be suitable for large datasets
Cosine	• Commonly used in text mining and information retrieval • Robust to the length of the vectors and effective with sparse data • Can handle multi-dimensional data	• Does not consider the magnitude or scale of the vectors • Assumes that the vectors are normalized, which may not always be the case • Sensitive to the presence of irrelevant or noisy features
Euclidean	• Commonly used in image recognition and clustering • Considers the magnitude and direction of the vectors • Intuitive and easy to understand	• Sensitive to the scale and units of the variables • Can be affected by outliers and noise in the data • May not be suitable for high-dimensional data

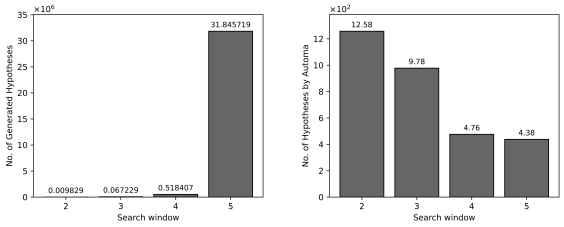
APPENDIX B
EMPIRICAL STUDY ON SIMILARITY FUNCTIONS/COEFFICIENTS

In the following, we study the impact of similarity functions on the performance of the overall system. In this study, we mainly focus on Pearson correlation coefficient and Jaccard, Cosine, and Euclidean similarity functions. Table V summarizes a qualitative comparison between these similarity functions/coefficients. However, AUTOMA is not limited to those functions and is able to use other similarity functions/coefficients [45].

TABLE VI: Qualitative comparison between different centrality functions.

Centrality function	Pros	Cons
Page-rank	• Considers the quality and quantity of incoming links to assess the influence and importance of nodes • Effective for large and complex networks • Can be used for personalized recommendations or ranking	• Can be manipulated by artificial creation of incoming links • May not be effective for smaller, densely connected networks
Closeness	• Identifies nodes with short path lengths to others, indicating quick reachability • Identifies key nodes for efficient information or resource dissemination • Easy to compute and suitable for networks of any size	• Less effective in identifying nodes crucial for overall network connectivity • Assumes equal node importance, which is not always true
Betweenness	• Identifies key bridging nodes connecting different network parts • Finds key nodes whose removal causes significant connectivity loss • Identifies bottlenecks and suggests efficient information/resource routing	• Computationally intensive for large networks • Ignores overall node connectivity or degree, limiting comprehensive measurement of node importance
Degree	• Easy-to-calculate metric, applicable to networks of all sizes • Measures node connectivity and identifies highly connected nodes • Identifies nodes important for overall network connectivity	• Ignores quality of connection and importance of highly connected nodes to other such nodes • May not identify nodes important for efficient information or resource routing

Fig. 19 shows the obtained ranking of relevant hypothesis, reduction rate (%), and execution time (s) for each technique used in APT41, per similarity function. We can notice that Cosine performs better since it can handle multi-dimensional data compared with Jaccard, Euclidean, and Pearson. Specifically, the Cosine similarity performs well in dense graphs (which is the case in our work) due to the fact it measures the similarity between nodes based on their attribute values. Jaccard similarity is commonly used in natural language processing applications, such as text classification and clustering (which is not suitable for our scenarios) and it is not efficient when it comes to cases where the length of the sequences varies significantly (*e.g.,* hypotheses sequences and APT sequences vary a lot). Euclidean distance is sensitive to the scale of the data and is not suitable for our scenario since each APT has a different length and each hypothesis/technique has different attributes. Finally, Pearson is not performing well since it is sensitive to outliers, which can affect the similarity score. Overall, the choice of the similarity function should be carefully considered based on the nature of the system telemetry and the application requirements (nature of knowledge base and APT definition) to achieve optimal performance.



(a) by naive. (b) by AUTOMA.

Fig. 16: Number of hypotheses in function of search window.

APPENDIX C
EMPIRICAL STUDY ON CENTRALITY FUNCTIONS

In the following, we study the impact of centrality functions on the performance of the overall system. In this study, we mainly focus on page-rank, closeness, betweenness, and degree. Table VI summarizes a qualitative comparison between these centrality functions. However, AUTOMA is not limited to those functions and is able to use other centrality functions.

Fig. 20 shows the obtained ranking of relevant hypothesis, reduction rate (%), and execution time (s) for each technique used in the 11 APT41 campaigns, per centrality function. We can notice that page-rank performs better since it considers the quality and quantity of incoming links (influence and importance of nodes) compared with closeness, betweenness, and degree. This is due to the fact that: (a) page-rank assigns a score to each node (*i.e.,* technique) based on its connections to other important nodes in the graph (*i.e.,* hypothesis sequence), and this score can be used to rank nodes, which has an impact on the overall hypothesis; (b) although closeness helps in indicating a node's reachability, it struggles in identifying nodes crucial for overall network connectivity (*i.e.,* entire hypothesis) while considering all nodes equal importance (which is not always the case); (c) Betweenness helps in identifying connectivity loss causes, yet this is not suitable in hypothesis evaluation since it considers only bridges between different parts of the graph (*e.g.,* ignoring other nodes that are not bridge but still important); and finally (d) degree centrality ignores the quality of connection and importance of highly connected nodes to others (equal node importance). Overall, the choice of centrality function depends on the application and use case requirements, such as letting an attacker advance and collect more artifacts or hunting it right away and don't collect any telemetry.

APPENDIX D
EMPIRICAL STUDY ON SEARCH WINDOW

In the following, we study the impact of the search window on the performance of the overall system. As discussed in Table I, expanding the search window leads to an exponential explosion in the number of possible hypotheses when dealing with a large number of techniques. In the following, we assume the system receives the technique "`Registry Run Keys / Startup Folder`" and then we measure different metrics in function of the search window (r).

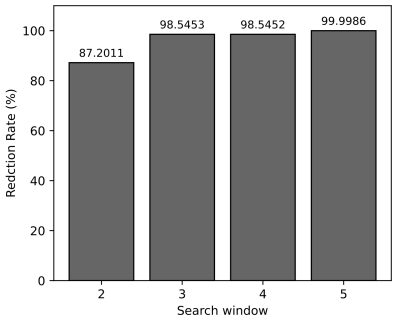
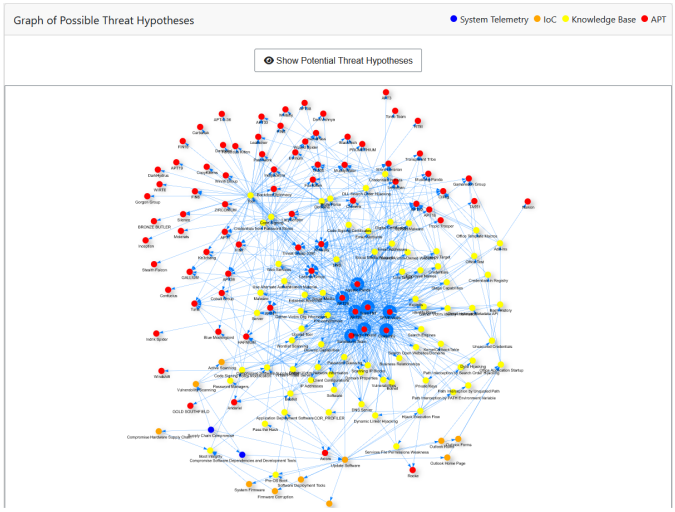


Fig. 17: Reduction rate in relation to the original number of hypotheses in function of the search window.

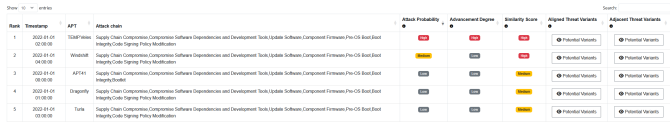
As we can see in Fig. 16(a), the number of possible hypotheses exponentially increases when we expand the search window value since there is no evaluation (*i.e.,* filtering) of the generated hypotheses. In contrast, Fig. 16(b) shows that the number of generated hypotheses by AUTOMA is minimized starting from $r = 3$ with a good tradeoff between the number of generated hypotheses, reduction rate, and ranking. This result is further proved by relevant hypothesis ranking (see Fig. 9) when starting from $r = 3$, the relevant hypothesis is always ranked almost the top 3 hypotheses. This result further leads to an average reduction of 99% in relation to all possible hypotheses generated by the naive algorithm (Fig. 17).

APPENDIX E
GUI TOOL FOR THREAT HYPOTHESES AND VARIANTS GENERATION

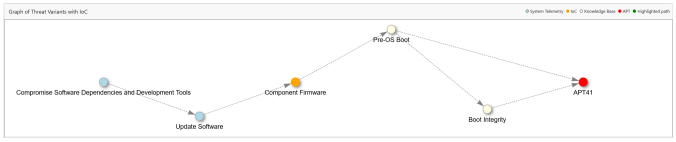
Fig. 18 shows different screenshots of our GUI tool for threat hypotheses and variants generation using AUTOMA. For a given security event, this tool assists the security experts during threat hunting by providing the list of most relevant threat hypotheses (Fig. 18(b)) and their variants (Figs. 18(c) and 18(d)) along with explainable metrics.



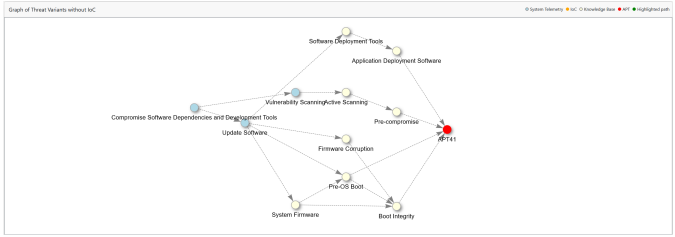
(a) List of all possible threat hypotheses.



(b) List of most relevant threat hypotheses.

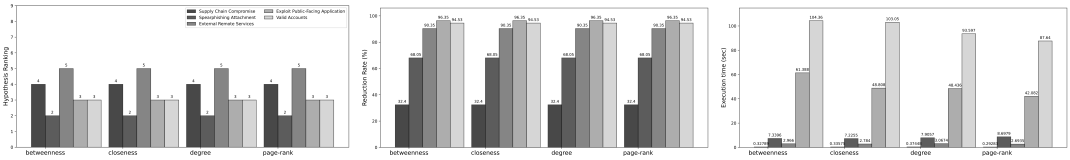


(c) Graphical representation of most relevant threat variants via received IoC.

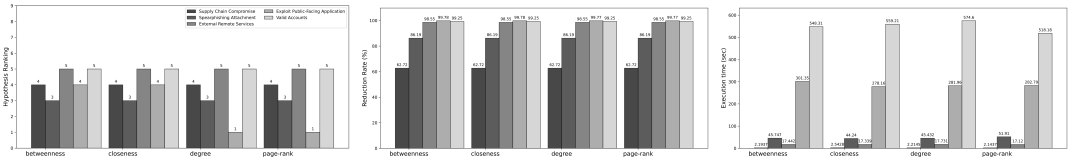


(d) Graphical representation of most relevant threat variants without using IoC.

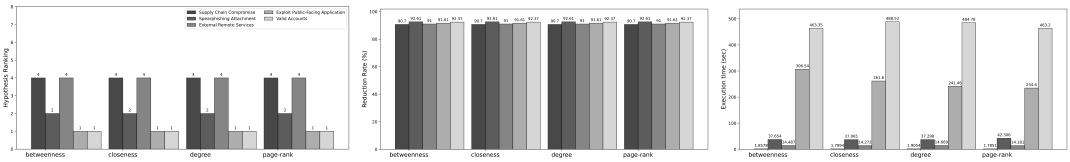
Fig. 18: Screenshots of our GUI tool for threat hypotheses and variants generation using AUTOMA.



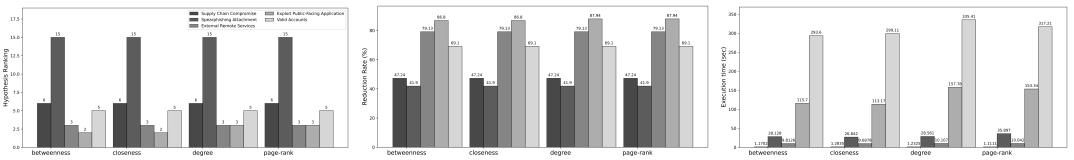
(a) Impact of Jaccard.



(b) Impact of Cosine.

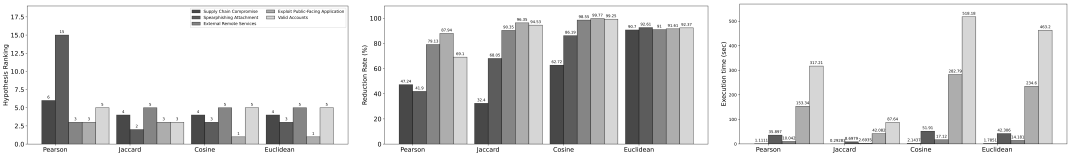


(c) Impact of Euclidean.

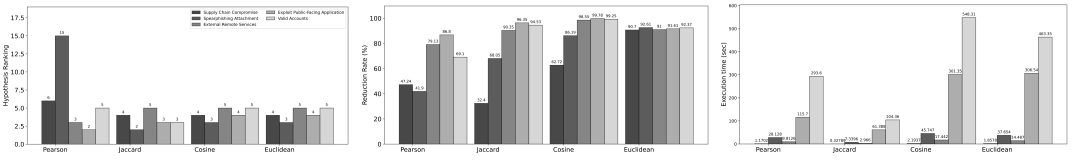


(d) Impact of Pearson.

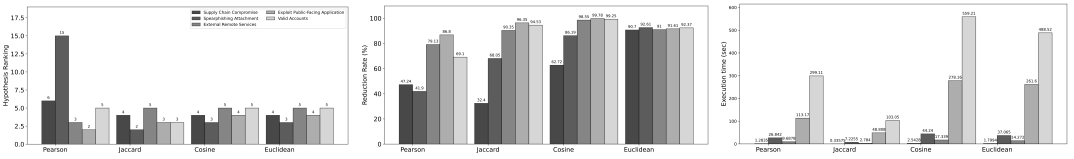
Fig. 19: Impact of similarity functions (from left to right: ranking of relevant hypothesis, reduction rate (%), and execution time (s)).



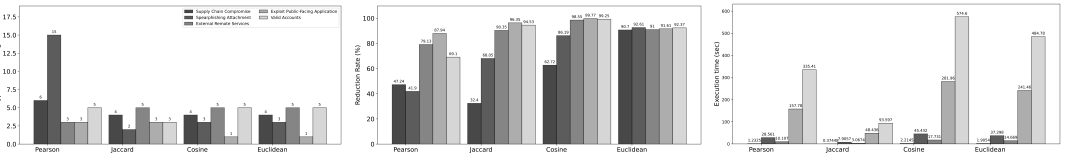
(a) Impact of page-rank.



(b) Impact of betweenness.



(c) Impact of closeness.



(d) Impact of degree.

Fig. 20: Impact of centrality functions (from left to right: ranking of relevant hypothesis, reduction rate (%), and execution time (s)).