Attack Hypotheses Generation Based on Threat Intelligence Knowledge Graph

Florian Klaus Kaiser, Uriel Dardik, Aviad Elitzur, Polina Zilberman, Nir Daniel, Marcus Wiens, Frank Schultmann, Yuval Elovici, and Rami Puzis

Abstract—Cyber threat intelligence on past attacks may help with attack reconstruction and the prediction of the course of an ongoing attack by providing deeper understanding of the tools and attack patterns used by attackers. Therefore, cyber security analysts employ threat intelligence, alert correlations, machine learning, and advanced visualizations in order to produce sound attack hypotheses. In this article, we present AttackDB, a multi-level threat knowledge base that combines data from multiple threat intelligence sources to associate high-level ATT&CK techniques with low-level telemetry found in behavioral malware reports. We also present the Attack Hypothesis Generator which relies on knowledge graph traversal algorithms and a variety of link prediction methods to automatically infer ATT&CK techniques from a set of observable artifacts. Results of experiments performed with 53K VirusTotal reports indicate that the proposed algorithms employed by the Attack Hypothesis Generator are able to produce accurate adversarial technique hypotheses with a mean average precision greater than 0.5 and area under the receiver operating characteristic curve of over 0.8 when it is implemented on the basis of AttackDB. The presented toolkit will help analysts to improve the accuracy of attack hypotheses and to automate the attack hypothesis generation process.

Index Terms—Attack hypotheses, cyber threat intelligence, data fusion, link prediction

1 Introduction

In the last years, the perpetrators of cyber attacks have been playing a dynamic cat and mouse game with those trying to stop them. In order to stay ahead of their opponents, cyber security analysts search for techniques that can assist them in threat hunting and intrusion detection, as well as in the forensic investigation of attacks, as they try to infer the attackers' objectives, trace back the attack vector used for initial penetration, and reconstruct the intermediate attack steps. The general workflow of these investigations starts with sensors that send monitored data to an organization's security information and event management system (SIEM). The SIEM aggregates and correlates the data from the sensors and generates alerts when a suspicious event is detected. On the basis of these alerts, security analysts derive hypotheses on the state of the system and draw conclusions about the attacker's goals, the methods he/she uses to achieve these goals, and further plans which is critical to providing a quick response to an attack and may thus result in reduced damage from an attack. Despite the great importance of these tasks, security analysts have little time to devote to them due to a shortage of experienced security analysts. Currently, even the best security operation centers are not fully automated and human analysts have primary responsibility for understanding, prioritizing, investigating, and responding to the alerts raised by the SIEM. Accumulated cyber threat intelligence (CTI) can help analysts to understand the goals and methods of relevant attack actors [1], [2].

Besides attack hypothesis generation, the automated use of CTI, such as indicators of compromise (*IoC*s), is prevalent throughout endpoint detection and response (EDR), extends beyond EDR as well as security orchestration automation and response solutions. Although there are many tools that utilize CTI (e.g., EDR), most of them concentrate on low-level constructs such as *IoC*s. Some automation employs cyber analytics, e.g., the MITRE Cyber Analytic Repository, to detect specific techniques. These tools

- *Florian Klaus Kaiser is with the Institute for Industrial Production (IIP), Competence Center for Applied Security Technology and Institute of Information Security and Dependability (KASTEL), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany. E mail: florian.kaiser2@kit.edu.*
- *Uriel Dardik, Aviad Elitzur, and Polina Zilberman are with the Cyber@BGU, Beersheba 84651, Israel. E mail: urield@post.bgu.ac.il, elitzura@post.bgu.com, polinaz@bgu.ac.il.*
- *Nir Daniel, Yuval Elovici, and Rami Puzis are with the Cyber@BGU, Beersheba 84651, Israel, and also with the Department of Software and Information Systems Engineering, Ben Gurion University of the Negev, Beer Sheva 8410501, Israel. E mail: nirdanie@post.bgu.ac.il, {elovici, puzis}@bgu.ac.il.*
- *Marcus Wiens is with the TU Bergakademie Freiberg, 09599 Freiberg, Germany, and also with the Competence Center for Applied Security Tech nology and Institute of Information Security and Dependability (KAS TEL), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany. E mail: marcus.wiens@kit.edu.*
- *Frank Schultmann is with the Institute for Industrial Production (IIP), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany, and also with the University of Adelaide, Adelaide, SA 5005, Australia. E mail: frank.schultmann@kit.edu.*

This research was partially supported in part by the Cyber Security Research Center at BGU , Helmholtz Information & Data Science Academy , NRF National Research Foundation under Grant NRF2016NCR NCR001 012, and U.S. Israel Energy Center managed by the Israel U.S. Binational Indus trial Research and Development (BIRD) Foundation.
(Corresponding author: Rami Puzis.)

operate in a top-down fashion as they hunt for the techniques supposedly employed by the adversary. The methods presented in this work differ from the state of the art by (1) focusing on the high-level indicators of attack (*IoA*s), i.e., collections of techniques, and (2) operating in a bottom-up manner, inferring collections of techniques from observable artifacts (not necessarily *IoC*s). This is an important step towards improving the cyber security of systems, as deriving high-level CTI from low-level CTI may help analysts with their tasks and contribute to increased efficiency among security analysts. However, deriving these high-level insights is challenging. Currently, except for cyber analytics, there is a lack of analysis tools and algorithms that utilize both high- and low-level CTI.

In this article, we build and extend AttackDB [3], a multi-level threat knowledge base that fuses data from the MITRE ATT&CK Enterprise knowledge base, the AlienVault Open Threat Exchange (OTX), the IBM X-Force Exchange (X-Force), and VirusTotal. We also introduce the Attack Hypothesis Generator (*AHG*), a toolkit that (1) infers adversarial techniques from low-level telemetry data using multiple techniques for information retrieval; and (2) refines a given hypothesis using various link prediction techniques.

The main contributions of this paper are as follows:

i) We contribute a comprehensive multi-level threat knowledge base that fuses multiple open-source threat intelligence sources [4].
ii) We utilize the proposed multi-level threat intelligence knowledge base to generate and attack technique hypotheses using graph analytics.
iii) We provide a heuristic, based on the expected number of techniques, which suggests when hypothesis refinement from our previous work [3] should be applied.
iv) Finally we compare the proposed techniques to a supervised machine learning learning approach.

All experiments are performed under the assumption that *AHG* is used to generate attack hypotheses when attack indicators are not known (e.g., a novel malware family utilizing a zero-day exploit).

AHG's ability to infer and refine the set of adversarial techniques used by an attacker is a critical step toward increasing the automation level of threat hunting and forensic investigations, both of which will contribute to organizations' cyber defense and enable them to gain insight regarding the state of their system (system monitoring). *AHG* can improve an analyst's perception of an attack under investigation and result in actionable insights pertinent to an attack. Furthermore, *AHG* addresses the "lack of published or accessible methodologies" [5] for threat hunting based on high-level attack patterns.

The rest of the paper is structured as follows. Relevant background and related work are described in Section 2. Section 3 presents the schema, insight into the construction process, and statistics regarding AttackDB. In Section 4, we present the proposed algorithms, both for hypothesis inference and hypothesis refinement; the various algorithms are evaluated in Section 5. Section 6 contains a summary, our conclusions, and plans for future research. Table 1 lists the notations and abbreviations used in this paper.

2 Background and Related Work

2.1 Cyber Threat Intelligence

CTI is structured, actionable information for identifying adversaries and their motives, goals, capabilities, resources, and tactics. It includes evidence-based knowledge in the form of measurable events and the context for the events' interpretation. This information can be clustered into four categories: (i) technical, (ii) tactical, (iii) operational, and (iv) strategic CTI [6]. Information extracted from CTI improves an analyst's ability to recognize relevant threats and respond to them in a timely manner [7], [8]. This is, methods for CTI analysis can provide analysts with a list of related information, supporting their decision-making as they handle cyber incidents [9]. Hence, CTI is a powerful means of increasing the efficiency of various security solutions, such as intrusion detection, incident response, real-time analytics, forensic investigation, and threat hunting. The practical use is validated within a survey of various cyber security and information technology management professionals presented by Shackleford [10]. According to the study, 48% of the respondents said their use of CTI has reduced incidents through early prevention, and 51% said they are able to respond more quickly to incidents.

CTI can be acquired by a victim organization that records attack investigation artifacts (such as *IoC*s), e.g., through anomaly detection systems respectively through intrusion detection systems. However, differentiating a benign anomaly (e.g., caused by irregular user behavior or the implementation of a new device) from an attack is oftentimes challenging leading to high false positive rates (fpr) and false negative rates (fnr) [11]. Therefore, assessing the relevance of observables (i.e., *IoC*s) is crucial for effective attack hypothesis generation. Capturing CTI is also critical and has been the subject of much research. Wheelus et al. [12] proposed a tiered Big Data architecture for the automated capturing and handling of network traffic; this enables the generation of features and artifacts for machine learning algorithms and anomaly detectors. Samtani et al. [13] suggested collecting CTI proactively from large international underground hacker communities without waiting for attacks to happen. They developed a framework for storing and analyzing malicious assets, such as crypters, keyloggers, and web and database exploits collected from the dark web. Landauer et al. [11] presented a methodology for automatically or semi-automatically transforming raw log data to actionable CTI. By doing so, they identified relevant information for threat hunting based on a continuous flow of raw log data using a parser tree and anomaly detection algorithms.

Furthermore, these low-level attack artefacts (technical CTI) are quickly actionable; however, very likely to become obsolete in a short time. Therefore, the value of low-level CTI (e.g., *IoC*s) for crafting attack hypotheses is debatable [11]. Hence, it is necessary to combine the benefits of more abstract and thus more robust (in terms of obsolescence) tactics and techniques (high-level CTI) with actionable *IoC*s (low-level CTI).

Since no organization possesses complete understanding of the threat landscape from recording attack artefacts, the importance of CTI lies in its ability be to shared among partners in a machine-to-machine manner. By sharing the who, what, where, how, and when of malicious activities, organizations

TABLE 1
Summary of Notations and Abbreviations

α	A smoothing parameter
Γ	Neighbors of an attack in $KG_[A]$
a	An attack
$\mathbb{A}$	Set of all attacks
AA	Adamic Adar
AD	Attack description
AP	Average precision
arh	Adaptive hypothesis refinement
AT	A set of attack techniques
CKC	Cyber Kill Chain
COD	Currently observed data
CTI	Cyber Threat Intelligence
$\mathbb{D}$	Set of all attack descriptions
E	Expectation
EDR	Endpoint Detection and Response
fnr	False negative rate
fpr	False positive rate
h	A hash value
H	Simulated human analyst (an ih strategy)
i	Specific Indicator of an Attack
ih	Initial hypothesis generation
IoA	Indicator of Attack
IoC	Indicator of Compromise
IP	Internet Protocol
J	Jaccard's/Tanimoto coefficient
K	Katz measure
KG	The AttackDB knowledge graph
$LOOCV$	Leave-one-out cross-validation
LP	Link prediction
$LPProjA$	Link prediction on projected attacks
$LPProjT$	Link prediction on projected techniques
$MLNB$-C	Multi-layer naive Bayesian classifier
N, k, j	Various counts
NB-C	Naive Bayesian Classifier
$\mathbb{OBS}$	Set of all observables
obs	An observable
OD	Observed data
P	Probability
PA	Preferential attachment
$ProjA$	Projected attack
$ProjAT$	Projected attack techniques
$ProjT$	Projected technique
R	Set of directed links connecting related SDOs
rh	Hypothesis refinement
ROC-AUC	Area under the receiver operating characteristic curve
SDO	STIX Domain Object
SIEM	Security Information and Event Management
STIX	Structured Threat Information eXpression
$SupLP$	Supervised link prediction
$\mathbb{T}$	Set of all techniques
T	A set of techniques
t	A technique
Tel	Telemetry
TF	Number of paths connecting two elements in KG
TTP	Tactics, Techniques, and Procedures
$TFIDF$	Term Frequency-Inverse Document Frequency (an ih strategy)
URL	Uniform Resource Locator
v	A component of an attack description
w	Ordering function on the basis of similarity metrics

obtain a holistic view of the threat landscape thus increasing their cyber security readiness [7]. However sharing CTI introduces novel risks for the trustworthiness of CTI. Thus, assessing the correctness and reliability of CTI is essential, as there may be untrusted data sources. Furthermore, a major challenge in sharing CTI is that it is shared in various different formats from a variety of sources. Therefore, information sharing needs to be streamlined and structured [14]. In an effort to formalize a standard language for sharing CTI, the US Department of Homeland Security's Office of Cybersecurity and Communications provided funding to MITRE to develop the Structured Threat Information eXpression (STIX) language. STIX covers the entire range of cyber security concepts, including observables, IoCs, attack patterns, tools, malware, threat actors, courses of action, and more. A STIX element is denoted as a STIX Domain Object (SDO). An IoC is an artifact or pattern which, if found, indicates that malicious activity is being performed. SDOs, such as observables and IoCs, are considered low-level CTI, while SDOs, such as attack patterns, tools, and threat actors, are considered high-level CTI. In the literature the term IoA is defined as the entirety of CTI available on an attack, including high-level descriptions of tactics, techniques, and procedures (TTP) [15].

Additional CTI languages include OpenIOC, Trusted Automated eXchange of Indicator Information, and the Incident Object Description Exchange Format, as well as proprietary languages and ontologies developed, e.g., Global Threat Intelligence by McAfee [16] or IntelGraph by Accenture [17]. Yet, according to a study provided by Sauerwein et al. [18], STIX is the de facto standard language for sharing CTI. Zhao et al. [19] presented an unified cyber threat ontology integrating heterogeneous CTI languages. An ontology can thereby be considered a meta-model of the knowledge graph which presents general domain concepts (Iqbal & Anwar [20]. There are many different sharing platforms with practical relevance, including the Malware Information Sharing Platform, OpenCTI, the Collective Intelligence Framework, Anomali STAXX, and the hrefhttps://otx.alienvault.com/OTX platform. De Melo e Silva et al. [21] compared and investigated a large variety of CTI languages and platforms and evaluated their strengths and weaknesses. For a review on the various languages we refer to their work.

While existing CTI ontologies, languages, and repositories are integral to the approach presented in this study, they are not sufficient for the effective generation of attack hypotheses. Mavroeidis and Bromander [2] reviewed existing ontologies and concluded that there is ”not any (...) ontology readily available for use” [2]. According to their study, existing ontologies need to be criticized for their ”lack of expressiveness” and missing holistic view. Motivated by their study, we partially bridge this gap using the proposed AttackDB in Section 3. AttackDB facilitates a holistic view of the different techniques used by a variety of malware, effectively connecting them with low-level CTI.

2.2 Threat Hunting

In this subsection, we provide background on threat hunting. Cyber security experts are divided regarding the exact stages of the threat hunting cycle and its reactive or proactive nature. On the one hand, some experts define threat hunting as proactively looking for early indications of presumably ongoing attacks without waiting for alerts to indicate suspicious activity [22]. On the other hand, threat hunting may refer to an investigative process initiated in response to an alert. This process may include advanced analytics, forensic investigations, targeted data collection, or policy updating [23], [24]. The main difference between proactive and reactive threat hunting is the trigger for the investigation. Proactive threat hunting relies on CTI to actively search for potentially malicious behavior. Reactive threat hunting involves forensic investigation and attack hypothesis testing in response to alerts indicating such behavior.

A significant amount of effort has been invested in the seamless integration of machines and human analysts within the threat hunting cycle [8], [24], [25], [26]. A noteworthy product that provides human-machine teaming capabilities is the McAfee Investigator [25]. This product can be considered a reactive threat hunting product, because it starts with choosing an incident for detailed investigation. Advanced machine learning algorithms choose the most relevant insights for the human analysts who can then determine the risk and urgency of the incident. After the analyst has chosen an incident for detailed investigation, the machinery uses human input to gather relevant information and provides a summary to the analyst.

Mavroeidis and Jøsang [27] presented an ontological approach for automating threat hunting using system monitoring logs. The authors discussed the potential benefits of CTI in investigating attack events and anticipating the next attack steps. However in contrast to our approach, they did not present hypothesis generation at the level of adversarial techniques.

Homayoun et al. [28] used sequential pattern mining techniques to identify features (i.e., activity logs) that are used for classification relying on J48, random forest, bagging, and multi-layer perceptron to detect and hunt ransomware. The authors showed that different types of ransomware can be identified by their frequent patterns and that this differentiation can be used to build CTI from log data. However, their work mainly focused on the classification task and does not consider the possibility of crafting high-level CTI.

Ranveer and Hiray [29] presented an overview of methods that can be used in different stages of malware detection. They focused on the phase of feature extraction and performed a comparative analysis of feature extraction methods for malware detection.

While most prior research on threat hunting has focused on automated detection and response, the current work focuses on the hypothesis generation phase of the threat hunting process.

2.3 Hypothesis Generation

2.3.1 Reasoning With Operational and Strategic CTI

Attack reconstruction, which is often the output of a successful threat hunting procedure, refers to describing a threat by presenting the different steps the attacker successfully executed. The security analyst should be able to explain how each step was achieved by pointing to the relevant events based on the evidence collected and its analysis [30].

Causal attack graphs. Polatidis et al. [31] proposed an approach for cyber attack (path) prediction using visualized attack graphs and recommender systems, They used naive Bayesian and random forest classifiers for attack prediction and reasoning with CTI. The victim organization structure and situational awareness are out of the scope of this paper. However, we employ Bayesian inference for the analysis of the CTI knowledge graph.

Milajerdi et al. [32] employed causal provenance graphs to model the organization structure and processes alongside the attacker activities. Their objective was to infer the high-level TTPs from system logs. The inference of the causal relationships for the construction of the provenance graph from logs requires extensive threat emulation within the target environment. Such information is usually not provided by major CTI sources. In contrast, the TTP inference proposed in this article relies on readily available general-purpose CTI published by a variety of sources.

Attack detection. AlEroud et al. [33] used domain knowledge to improve initial predictions and create an accurate attack profile. Attacks were represented as nodes on semantic link networks. First, the authors ranked predictions according to the networks, and then, with domain knowledge and a taxonomy, they adjusted their predictions according to the predictions' correlation to the taxonomy. Fard et al. [34] presented a multi-view ensemble threat hunting model. For threat hunting they relied on weighted majority voting using sparse representation-based classifiers. Thereby, each classifier detects malware, using one feature set as a single modality of available CTI. In an experimental setting using three different data sets (e.g., extracted from the VirusTotal Threat Intelligence platform), they showed that the proposed approach delivers high accuracy with little computational burden. Bhatt et al. [35] also suggested a threat detection model. The proposed model is used to improve hypothesizing on ongoing attacks given correlated events, and knowledge on the Cyber Kill Chain[1] (CKC) [36]. In contrast to works in this category, AHG is not intended for attack detection. Rather, we aim to infer the most probable attack techniques given a set of suspicious artifacts under the assumption that an organization is already under attack.

Inferring TTPs/CKC phases from artifacts. Wafula and Wang [37] suggested a threat hunting hypothesis development methodology for identifying the threat actor, target assets, relevant vulnerabilities, and artifacts. They suggested using exploratory data analysis of CTI for both the generation of an initial hypothesis and validation of the hypothesis. Giura et al. [38] proposed an attack pyramid that aims to capture the movements of an attacker through CKC phases (represented by the levels of the pyramid) and the organization's environments (represented by the planes of the pyramid), e.g., physical, network, user, and application. The attacker's goal is located at the top of the pyramid, and it is reachable by stepping from one event to another. The pyramid enables the detection of the attack path and the attacker's goal. Iqbal et al. [20] built a unified graph representation for CKC and Pyramid of Pain models by extracting entities from textual reports. They compared two variants of the same attack and showed that they could map TTPs for each stage of the attack using the graph which would allow analysts relying on the methods presented by the authors to derive and predict missing TTPs from the graph. However, the approach presented does not provide automation and demonstrated the predictive power for only two malwares. Rubinshtein et al. [39] built an attack ontology from logs and reconstructed attack steps based on this ontology. In addition to generating actionable CTI from logs, Landauer et al. [11], processed the CTI generated using advanced unsupervised machine learning methods to transform the anomalies detected into hypotheses about abstract attack patterns.

Taken together, these pioneering works form the foundation upon which we build our hypothesis generation approach. Most prior state-of-the-art methods demonstrate inference of TTPs using expert based rules on just a few attacks that exhibit a similar attack flow. We build AttackDB – a comprehensive knowledge graph constructed top down, relying on major CTI sources (MITRE ATT&CK, AlienVault, X-Force Exchange, and VirusTotal). AttackDB enables to

1. CKC was developed by Martin Lockheed to model the diverse threat landscape. The CKC describes seven steps that an attacker must perform in order to accomplish the goal of an attack.

streamline the process of TTP inference by relying on network based inference instead of manually defined sets of rules. In addition, as proposed by Rubinstein et al. [39], we add similarity scores and hypothesis generation algorithms to increase the attack reconstruction and threat hunting capabilities. In this paper we investigate a wider set of approaches for crafting robust hypotheses.

2.3.2 *Reasoning Using CTI Knowledge Graphs*

In the field of cyber security, the use of knowledge graphs, which are systematics representing CTI with the help of directed labeled graphs, is in its early stages and is mainly used for visualization and less for prediction. One of the first studies on the use of knowledge graphs for analysis was performed by Lee et al. [40] who built a knowledge graph from open-source intelligence. Based on established graph algorithms, they improved the identification of malicious nodes and attack infrastructures, as well as the relationship among attack groups and their similarity. In particular, they used page rank and betweenness metrics to detect relevant information in the graph. Gao et al. [41] presented a trust evaluation mechanism to assess which information is relevant (malicious) and which is not. They trained supervised classification algorithms based on a random forest classifier to distinguish between trusted and untrusted information. They found that graph-based features increase the accuracy of the trained models. Link prediction techniques have shown to provide valuable insights from CTI-based knowledge graphs [42], [43]. Inspired by these works, we employ supervised machine learning and link prediction methods for the inference of TTPs from artifacts through knowledge graph analysis.

Najafi et al. [44] proposed a novel graph-based inference algorithm to evaluate the maliciousness of *IoC*s. The proposed algorithm outperformed established prior state-of-the-art algorithms, such as belief propagation and SimRank, and showed that the proposed algorithm was particularly effective in identifying previously unknown *IoC*s. Milajerdi et al. [45] modeled threat hunting as an inexact graph pattern matching problem based on kernel audits, with relationships between CTI used as reliable artifacts. However, the query graphs are manually constructed by the researchers for each instance of the attack.

Qamar et al. [46] and Riesco and Villagrá [47] proposed data-driven analytics based on the STIX ontology. They used logic-based deductive inference rules (defined in Semantic Web Rule Language [48]) and defined queries for evaluating threat likelihood and managing cyber threat response activities. Although logic rules may detect complex patterns in CTI, a domain expert is required to define the rules, which is the main drawback of this method.

Ulicny et al. [49] highlighted the need for inferences to be automated to cope with the high dynamic developments in the field of cyber security and motivate the automation of threat hunting relying on CTI. The authors showed the possibility of automated threat detection based on Web Ontology Language [50]. They thereby introduced an approach which mimics the work of a human analyst.

Elitzur et al. [3]present algorithms for refining *IoA*s crafted by a human analyst. These algorithms can be used within a decision support system but hardly automate the investigation process. We aim at contributing to both the automated inference of *IoA*s and the refinement of *IoA*s inferred by a human analyst. In contrast, to graph alignment and logical rules mentioned above, the methods proposed in this paper allow full automation of TTP inference. We rely on CTI sources that include many attack variants and are readily available to security teams. The threat intelligence knowledge base, AttackDB, that we present [4] in this paper is unique comprising all malware families from MITRE ATT&CK and connecting high-level TTPs with low-level observable artifacts.

3 Multi-Level Threat Knowledge Base

3.1 Schema

In this section we describe AttackDB - a multi-level threat knowledge base. AttackDB contains SDOs at all levels of the Pyramid of Pain [51], from abstract concepts, such as tactics and top-level techniques, down to *IoC*s and specific observables, such as hashes, Internet protocol (IP) addresses, and domain names.

Fig. 1 depicts AttackDB's schematic structure. We utilize the definition of SDOs and relationships between them with a few exceptions; for example, we do not take advantage of all SDOs defined in STIX (e.g., location). We also use a single concept of attack instead of the intrusion set, campaign, and malware defined in STIX.

The top-level SDOs in AttackDB are attack patterns (a.k. a. tactics and techniques). Tactics are the most abstract representations of attacks in AttackDB and represent tactical goals of attackers. Techniques denote the actions attackers take to achieve the tactical goal. SDOs include the malicious activities exhibited by malware, campaigns, or intrusion sets. Malware is software that exhibits a set of malicious activities. Malware can be a part of multiple campaigns. A campaign is a set of malicious activities performed for a specific period of time against specific targets. Campaigns that are believed to be orchestrated by the same threat actor may be grouped into intrusion sets. Despite the semantic differences between them, malware, campaign, and intrusion set SDOs can be used to represent an abstract attack that is

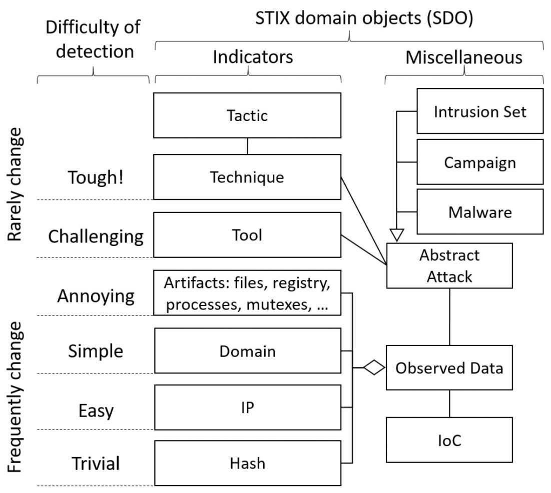

Fig. 1. AttackDB schema with detection difficulties according to the Pyramid of Pain.

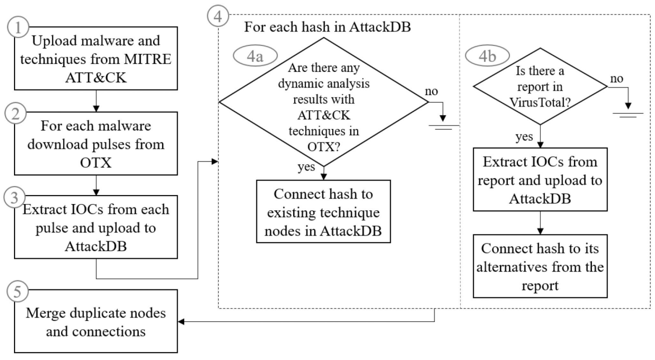
1
Upload malware and techniques from MITRE ATT&CK
2
For each malware download pulses from OTX
3
Extract IOCs from each pulse and upload to AttackDB
4
For each hash in AttackDB
4a
Are there any dynamic analysis results with ATT&CK techniques in OTX?
no
yes
Connect hash to existing technique nodes in AttackDB
4b
Is there a report in VirusTotal?
no
yes
Extract IOCs from report and upload to AttackDB
Connect hash to its alternatives from the report
5
Merge duplicate nodes and connections

Fig. 2. Flow chart of AttackDB's construction.

being hunted. A tool is software which can be used by an adversary.

AttackDB also contains observed data SDOs associated with an attack on the one hand and with a CTI report on the other. Report SDOs are included in AttackDB to trace back the CTI to its source (see Section 3.3) but are not used for hypothesis generation. Observed data SDOs may aggregate hashes, IP addresses, domains, network, and host artifacts (i.e., telemetry), such as process names, services, registry keys, and other artifacts. Artifacts may be grouped together in a pattern and tagged as an *IoC*. *IoC*s can be used to identify attacks observed in the past but are usually easily modified by the attacker. All of the observed data stored in AttackDB is processed and used to create attack hypotheses, rank the hypotheses according to their probability, and generate workflows for proper response to (and in this sense defending against) the hypothesized attack.

3.2 Data Fusion

In the current implementation, the AttackDB knowledge graph is stored in a Neo4j[2] database, however any graph database may be used for this purpose. With AttackDB, we constructed a rich knowledge base that consists of CTI from the MITRE ATT&CK Enterprise knowledge base, the OTX, the X-Force, and VirusTotal. Relationships between different objects are included to build the knowledge graph, as they are described in malware analysis reports extracted from the different CTI sources. Details regarding the data extraction process and data fusion are provided below. The process of constructing AttackDB is presented in Fig. 2.

MITRE's ATT&CK is an open CTI knowledge base that contains information on adversarial techniques and tactics, threat actors, mitigation, malware, and tools [30]. First, we populate AttackDB with malware and techniques and the relationships between them, extracted from MITRE ATT&CK.

OTX provides CTI in the form of pulses, which contain one or more *IoC*s, such as file hashes, Uniform Resource Locators (URLs), and IPs. Pulses can be tagged with malware names, threat actors, and additional information. In the second step, we search for pulses, using malware names from MITRE ATT&CK, via the OTX Application Programming Interface and link malware nodes in AttackDB to *IoC*s from the respective pulses. However, AlienVault allows anyone to post a set of *IoC*s as a pulse, which may lead to unreliable data. Therefore, we only use pulses posted by the top 20 publishers with the most subscriptions who posted pulses related to the malware searched for: AlienVault, MalwarePatrol, jnazario, niddel, Metadefender, cyberprotect, popularmalware, Malwaremustdie, Cyber_Hat, burberry, bartblaze, ESET-Spain, julsec, zer0daydan, rpsanch, erik, milind, BLUELIV, techhelplist, BotnetExposer, and nightingale.

IBM X-Force provides malware reports, which contain *IoC*s of various types, such as URLs, domain names, filenames, and processes. As with AlienVault, we search for reports using malware names and link malware nodes in AttackDB to *IoC*s from the respective reports. We use file hashes retrieved from AlienVault and X-Force to fuse the pulses with VirusTotal reports.

In the third step, we enrich AttackDB with malware telemetry, such as network and host artifacts. For this purpose, we retrieve behavioral analysis data from *VirusTotal* for all hashes obtained from OTX and X-Force. The behavioral data retrieved from VirusTotal includes file names (opened, created, searched, etc.), URLs, domains, IPs, process names, registry keys, mutual exclusions (mutexes), emails, and more. Note that during the population of AttackDB, *IoCs* with identical patterns should be represented by the same node, as well as observables with identical values.

Fig. 3 provides an illustration of AttackDB's structure containing two attack nodes (A1 and A2) and all of the relevant connections. At the top, we see malware and associated techniques extracted from MITRE ATT&CK. The three respective *IoC*s (two hash values and one URL) are extracted from OTX and X-Force. Following the STIX format, the relevant observed data nodes are connected with *IoC*s connected with the relevant attack (i.e., malware). Finally, the behavioral data extracted from VirusTotal is displayed at the bottom of the figure. This observed data is not connected with *IoC* nodes, because it is not necessarily a strong indication of the attack but is merely a collection of artifacts generated by the malware during dynamic analysis.

Note that a behavioral artifact may be connected to malware through multiple paths. This happens when there are multiple instances of the same malware analyzed by

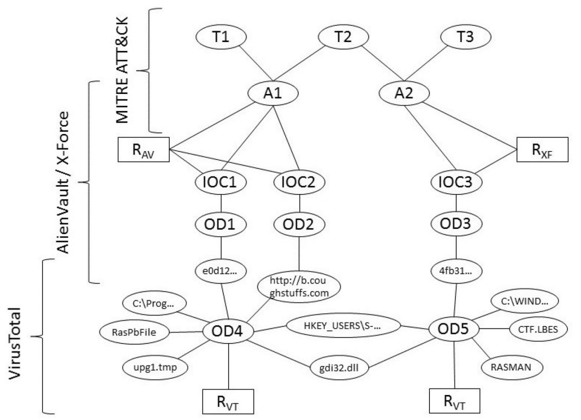

Fig. 3. An illustration of AttackDB's structure.

2. https://neo4j.com/

VirusTotal. Also note that observables are indirectly connected to techniques through the respective malware. We use this connection to build attack hypotheses, as described in Section 4.3.

3.3 Duplicate Data and Malware Aliases

In the resulting AttackDB there are hash nodes that are connected to two or more malware nodes. Since hash nodes represent specific malware instances, ambiguous connections to malware nodes require additional clarification. The possible reasons for this phenomenon are described below.

We analyze all occurrences of hash nodes shared by attack nodes. For each shared hash we analyze the relevant malware analysis reports in order to categorize the relationships between the attack nodes sharing the same hashes. The nature of these relationships is diverse: ambiguous naming (aliases), shared actor, shared campaign, belonging to the same malware family, shared infection mechanism, errors in the reports' parsing process (i.e., extraction error), etc.

Roughly, we divide reports into two types: (1) reports that describe some factor, such as actors, campaigns, families, infection mechanisms, or *IoC*s overlaps, common to multiple malware instances; and (2) reports that describe a single specific malware. Reports of the first type often contain references to multiple malware binaries. In such cases, an attack node is connected not only to its representative hash but to all the hashes provided in malware analysis reports. Reports of the second type describe a single malware containing a single representative hash, however there may be different reports describing the same malware with different names. In such cases a single hash is connected to all the malware aliases that appeared in the reports.

Ambiguous connections whose origin is in both types of reports provide meaningful information, and therefore, we include them in AttackDB. Connections caused by errors in the process of extracting data from the malware analysis reports have been removed.

We analyze the reports connected to hash nodes with ambiguous connections, searching for connections erroneously omitted and connections caused by extraction errors. This manual analysis resulted in the connection of 74 reports to 93 malwares and disconnecting 54 reports from 35 malwares. Connecting a report to a malware means connecting the malware to all observables, including *IoC*s, that appear in the report, while disconnecting a report from a malware means disconnecting the malware from all *IoC*s that appear in the report.

Example 1 (Adding connections). The XTunnel malware appears in a pulse in OTX and therefore is connected to the corresponding report in the graph. There are also *IoC*s from the CORESHELL and USBStealer malwares in the same report, because XTunnel, CORESHELL, and USBStealer belong to the APT28 group. However, during the data fusion process, when searching for CORESHELL and USBStealer, the report did not come up, resulting in missed connections.

Example 2 (Removing connection). We removed connections between the RTM malware and reports that contain the word "depa**rtm**ent.".

3.4 Knowledge Base Summary

The resulting AttackDB fuses data from 1,675 AlienVault pulses, 281 IBM X-Force reports, and 53,005 VirusTotal reports. It contains 253 malware nodes associated with 144,216 *IoC*s. Around 60,000 of the *IoC*s are file hashes, and the rest are domain names, IPs, etc. In total there are over half a million observables in AttackDB. All of the techniques in the graph (190) are connected to some malware. The average number of techniques per malware is 10.3. We only include techniques that are associated with a malware by MITRE within this study.

4 Attack Hypothesis Generation

4.1 High-Level Overview

Assume that suspicious events are taking place in an organization. The goal of *AHG* is to propose an hypothesis on the course of the possible attack. Fig. 4 provides an overview of the hypothesis generation process. The resulting hypothesis consists of a set of MITRE ATT&CK techniques that are closely related to: (1) the observed data, and (2) each other.

A hypothesis that consists of ATT&CK techniques that are closely related to the observed data can be obtained by an analyst who investigates the suspicious events and formulates an initial hypothesis based on alerts and various artifacts recorded by the organization's SIEM (see Fig. 4(1)). In addition, an initial hypothesis can be inferred by ranking adversarial techniques based on currently observed data (*COD*) stored in the system (see Fig. 4(2)). In Section 4.3, we describe two approaches for inferring an initial hypothesis: inferring techniques related to *IoC*s in AttackDB and inferring techniques related to telemetries in AttackDB. These two approaches are implemented by using three methods (initial hypothesis generation (ih) algorithms) that produce a relationship that maps the observed data to techniques: A method inspired by term frequency inverse-document-frequency (*TFIDF*) (see Section 4.3.1) a multi-nomial naïve Bayesian classifiers (*NB-C*) (see Section 4.3.2), and a multinomial multi-layer naïve Bayesian classifiers ($MLNB$-C) (see Section 4.3.3).

A hypothesis that consists of ATT&CK techniques that are closely related to each other can be crafted by refining the set of techniques (relying on hypothesis refinement (rh) algorithms) comprising the initial hypothesis using a recommender system-based technique (see Fig. 4(3)). Section 4.4 describes five recommender system techniques for refining the initial hypothesis: the projected technique (ProjT, Section 4.4.1), link prediction on projected technique (LPProjT, Section 4.4.2), link prediction on projected attack (LPProjA,

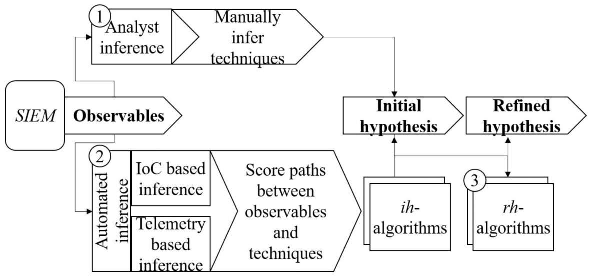

Fig. 4. Overview of the hypothesis generation process.

Section 4.4.3), projected hypothesis (ProjAT, Section 4.4.4), and supervised link prediction (SupLP, Section 4.4.5).

To make it easier for the analyst, all methods used to produce the initial and refined hypotheses aim to accurately rank the ATT&CK techniques so that the most probable techniques appear first. In Section 4.5, we propose a method for estimating the number of related techniques. The refinement mechanism selected is applied based on the results of the method for estimating the number of related techniques.

4.2 Problem Definition

Assume a knowledge graph as described in Section 3.2. Malware SDOs that are used to represent an attack are grouped in a super-set denoted as $\mathbb{A}$. An *attack descriptive* SDO contains information about an element that has been used or targeted by the attack. SDOs that can be used to describe an attack, specifically techniques (a.k.a. attack patterns), IoCs, observed data and specific observables, are grouped in a super-set denoted as $\mathbb{D}$.

Definition 1 (Cyber security knowledge graph). *A cyber security knowledge graph $KG = \langle \mathbb{A}, \mathbb{D}, R\rangle$ is a graph where $\mathbb{A}$ contains the nodes representing past attacks, and $\mathbb{D}$ contains the description nodes (specifically, techniques, IoCs, observed data, and observables). R is the set of directed links connecting related SDOs according the schema depicted in Fig. 1.*

Definition 2 (Attack descriptions). *Given an attack representation $a \in \mathbb{A}$, we refer to the set of $v \in \mathbb{D}$ that are at most five hops away from a as the* attack description *$AD_a = \{v | dist(a, v) \leq 5\} \subseteq \mathbb{D}$. We refer to all $v \in AD_a$, where v is a technique, as the* attack techniques *$AT_a = \{v | v \in AD_a \wedge type(v) = Technique\}$.*

Assume an unknown ongoing attack a_{new} currently being investigated by the analyst. If an analyst constructs the initial hypothesis, then he/she adds a_{new} to the KG and begins a preliminary investigation by concentrating on the recent alerts and related telemetry.

Problem 1 (Initial hypothesis generation problem). *Given a knowledge graph KG and COD in the SIEM, generate an initial hypothesis of an attack, denoted as $AT^{init}_{a_{new}}$, consisting of techniques closely related to COD.*

According to a preliminary investigation, the analyst connects a_{new} to technique SDOs, denoted as $AT^{init}_{a_{new}}$. $AT^{init}_{a_{new}}$ may also be referred to as an attack hypothesis consisting of a set of techniques.

The initial hypothesis can also be inferred automatically by ranking techniques based on COD, as will be elaborated on in Section 4.3.

AHG consists of a module that refines $AT^{init}_{a_{new}}$ by recommending more relevant techniques and ignoring or omitting techniques that are not relevant.

Problem 2 (Hypothesis refinement problem). *Given an initial hypothesis $AT^{init}_{a_{new}}$ and a knowledge graph KG, generate a new hypothesis, denoted as $AT^{ref}_{a_{new}}$, that is more accurate than $AT^{init}_{a_{new}}$ with respect to the correct description of the real attack $AT^{*}_{a_{new}}$.*

4.3 Initial Hypothesis Generation

$AT^{init}_{a_{new}}$ is constructed by selecting the techniques most relevant to COD. For this purpose we employ different methods that produce relationships mapping COD to techniques. The initial hypothesis generation is either based on a simulated human analyst (a strategy denoted as H) or automated, relying on one of the ih algorithms presented below. Note that COD may be either IoCs or telemetries, depending on the inferring technique approach employed.

Algorithm 1. CountPaths

```
Input: $observables, AttackDB, ih_{strategy}$
Output: $num\_techs\_per\_obs, related\_techs$
1:  $num\_techs\_per\_obs \leftarrow \{\}$
2:  $related\_techs \leftarrow Set([])$
3:  for $obs \in observables$ do
4:    $num\_techs\_per\_obs[obs] \leftarrow \{\}$
5:    if $ih_{strategy} == IoC$ then
6:      $techs_{obs} \leftarrow GetTechsConnectedToIoC(obs)$
7:    else
8:      $techs_{obs} \leftarrow GetTechsConnectedToTel(obs)$
9:    for $tech \in techs_{obs}$ do
10:     add $tech$ to $related\_techs$
11:     $num\_techs\_per\_obs[obs][tech] \leftarrow CountPaths(obs, tech)$
12: return $num\_techs\_per\_obs, related\_techs$
```

4.3.1 Term Frequency-Inverse Document Frequency

We propose the use of a technique scoring mechanism based on $TFIDF$ common in information retrieval. In this case, techniques are analogous to documents, and observables are analogous to search terms for the purpose of $TFIDF$ computation. A technique t is relevant to an observable obs if obs appears in a report on attack a that uses t. An observable may appear in several reports and be connected to a technique through multiple attacks.

Definition 3 (Connected observables to a technique). *$TF(obs, t)$ is the number of paths from obs to t in the knowledge base.*

In the discussions that follow, we use $x : Y$ notation to indicate $type(x) = X$, and we use dot $x : X - y : Y$ to indicate that x and y are connected in KG. When a technique's scoring is based on IoCs (obs is an IoC), then the set of paths between obs and t is defined as follows:

$$TF(obs, t) = |\{obs - (od : OD) - (i : IoC) - (a : A) - t\}|,$$

When a technique's scoring is based on telemetries (obs is a telemetry, $type(obs) = OD \wedge type(obs) \neq IoC$), then the set of paths between obs and t is defined as follows:

$$\begin{aligned} TF(obs, t) = |\{&obs - (od : OD) - (h : hash) \\ &- (od : OD) - (i : IoC) - (a : A) - t\}|. \end{aligned}$$

Note that the hash is a subset of IoC ($hash \subseteq IoC \subseteq OD \subseteq \mathbb{D}$). For example, in Fig. 3 there are three 6-hop paths connecting the *gdi32.dll* observable at the bottom of the figure (in the middle) with the technique T2.

There are several different versions of inverse document fequency (IDF) assessment available in the literature. We use the simplest one which is the logarithm of the total number of techniques divided by the number of relevant techniques in AttackDB.

Definition 4 (Term frequency-inverse document frequency of techniques). *Let $\mathbb{T} = \{t\}$ be the set of techniques in AttackDB. Let be T a subset of $\mathbb{T}$ denoting the relevant t given obs; then IDF can be defined as the following*

$$IDF(obs, t) = Log_e\left(\frac{|\mathbb{T}|}{|T|}\right). \quad (1)$$

The score of t is the sum of the $TFIDF(obs, t)$ values for all COD.

$$TFIDF(obs, t) = TF(obs, t) \cdot IDF(obs, t) \quad (2)$$

$$TFIDF(t) = \sum_{obs \in COD} TFIDF(obs, t). \quad (3)$$

4.3.2 Naïve Bayesian Inference

We implement a multi-nomial *NB-C*. Multi-nomial *NB-C*s can be based on word vector counts as well as on *TFIDF* [52]. Here, we present a *NB-C* based on vector counts. Therefore, the prior probabilities are extracted from the *KG* referring to the number of relevant observables. Hence, a priori probabilities represent prior knowledge that can be extracted from the *KG*.

Definition 5 (Prior probabilities). *We assume the prior probability of the utilization of a technique $P(t)$ to be a priori equal for each t. Consequently, $P(t)$ can be defined as follows:*

$$P(t) = \frac{1}{|\mathbb{T}|}. \quad (4)$$

We furthermore assume, $P(a)$ the prior probability of the utilization of an attack to be equal for each a.

$$P(a) = \frac{1}{|\mathbb{A}|}. \quad (5)$$

We assume the occurrence of an obs to be deterministic in the occurrence of the related attack. Let A denote all relevant (related to the specific obs) attacks. Then, the prior probability of the observation of a specific observable $P(obs)$ can be defined as follows:

$$P(obs) = \sum_{a \in A} P(a). \quad (6)$$

Based on these prior probabilities, the posterior probability that attack technique t is relevant given COD is defined by Equation (7)

$$P(t|COD) = \frac{P(t) \cdot \prod_{obs \in COD} P(obs|t)}{P(COD)} \quad (7)$$

$$P(obs|t) = \frac{TF(obs, t) + \alpha}{\sum_{t \in T} TF(obs, t) + \alpha} \quad (8)$$

$$P(COD) = \prod_{obs \in COD} P(obs), \quad (9)$$

where α is a smoothing prior. This smoothing prior can be used for Laplace smoothing ($\alpha = 1$) or Lidstone smoothing

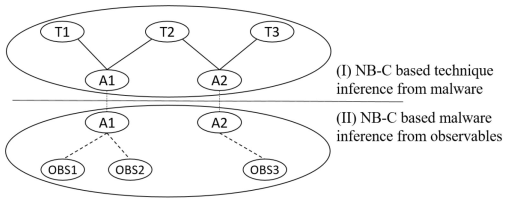
T1
T2
T3
A1
A2
(I) NB-C based technique inference from malware
A1
A2
(II) NB-C based malware inference from observables
OBS1
OBS2
OBS3

Fig. 5. Framework of multi-layer naïve Bayesian inference of initial hypotheses within AttackDB.

($\alpha < 1$). With the smoothing prior it is possible to account for *obs* and connections that are not present in AttackDB (e.g., unknown). Furthermore, it prevents probabilities of zero. Therefore, the use of this additive smoothing parameter may hence increase the accuracy of classification.

4.3.3 Multi-Layer Naïve Bayesian Inference

Leveraging on the ontology of the *KG*, we use *MLNB-C*. This inference algorithm allows to use causal relationships presented in AttackDB and therefore step-wise reduces the assumption of conditional independence of *NB-C*. The proposed method consists of two tiers (see Fig. 5): (I) *NB-C* for technique inference from malware; (II) *NB-C* for malware inference from *COD*.

According to the law of total probability, the probability of a technique t given COD can be defined as described in Equation (10).

Definition 6 (Connected attacks to a technique). *$TF(a, t)$ is the number of paths from a to t in KG. Equivalently, $TF(obs, a)$ describes the number of paths from obs to a in KG, where $TF(a, t)$ and $TF(obs, a)$ are calculated symmetrically to $TF(obs, t)$*

$$P(t|COD) = \sum_{a \in \mathbb{A}} \frac{P(t) \cdot P(a|t) \cdot P(a|COD)}{P(a)} \quad (10)$$

$$P(a|t) = \frac{TF(a, t) + \alpha}{\sum_{t \in T} TF(a, t) + \alpha} \quad (11)$$

$$P(a|COD) = \frac{P(a) \cdot \prod_{obs \in COD} P(obs|a)}{P(COD)} \quad (12)$$

$$P(obs|a) = \frac{TF(obs, a) + \alpha}{\sum_{a \in \mathbb{A}} TF(obs, a) + \alpha}. \quad (13)$$

4.4 Hypothesis Refinement

Next, we refine $AT^{init}_{a_{new}}$, relying on *rh* algorithms. This stage accounts for interdependence between techniques by increasing the score of techniques which are often used together in the same attacks.

For hypothesis refinement, we rely on a set of link prediction (LP) techniques and similarity metrics, namely Jaccard's/Tanimoto coefficient [53], [54], Adamic Adar [55], Friends measure/ Katz measure [56], and Preferential Attachment [57]. Here we define these algorithms, applying them to a, although the algorithms can analogously be used for AT and t.

Definition 7 (Jaccard's/Tanimoto coefficient). *The Jaccard's/Tanimoto coefficient $J_{\mathbb{A}}$ is a similarity metric* *describing the probability that a_i and a_j share the same AT for an AT that either a_i or a_j has. It is defined as follows.*

$$J_{\mathbb{A}}(a_i, a_j) = \frac{\left|AT_{a_i} \cap AT_{a_j}\right|}{\left|AT_{a_i} \cup AT_{a_j}\right|}. \tag{14}$$

Definition 8 (Adamic Adar). *Adamic Adar $AA_{\mathbb{A}}$ describes a frequency-weighted metric formalizing the notion that rarely observed techniques of a are more informative than frequently observed techniques. $AA_{\mathbb{A}}$ is defined as follows.*

$$AA_{\mathbb{A}}(a_i, a_j) = \sum_{t \in AT_{a_i} \cap AT_{a_j}} \frac{1}{log|AT(t)|}. \tag{15}$$

Definition 9 (Katz measure). *Katz measure $K_{\mathbb{A}}$ is a heuristic defined by the sum of the path length of connections. It is assumed that the shorter the paths connecting a_i and a_j are and the more paths that exist, the more similar the attacks. Therefore, it can be described as a variant of the shortest-path measure*

$$K_{\mathbb{A}}(a_i, a_j) = \sum_{l} \beta^l \cdot TF^l(a_i, a_j), \tag{16}$$

where β is a scaling parameter, and $TF^l(a_i, a_j)$ is the number of paths from a_i to a_j of length l.

For the undirected KG, the Katz measure is equal to the Friends measure, where $\beta = 1$ and $l_{max} = 2$.

Definition 10 (Preferential Attachment). *Preferential Attachment $PA_{\mathbb{A}}$ calculates the similarity without relying on relational information. $PA_{\mathbb{A}}$ can be computed as follows.*

$$PA_{\mathbb{A}}(a_i, a_j) = \left|AT_{a_i}\right| \cdot \left|AT_{a_j}\right|. \tag{17}$$

Given a snapshot of KG, LP techniques identify missing relationships within the knowledge graph that are likely to occur. LP techniques can also be used to identify relationships that are likely to be incorrect within the KG. We apply five techniques to predict the relationships between the SDO representing the new attack a_{new} and the relevant technique SDOs. We rank each technique t and select the top N to serve as $AT^{ref}_{a_{new}}$.

As described in previous sections, KG is a graph linking attacks to relevant techniques. Following the common practice in recommender systems [58], in some of the following approaches we rely on the one-mode projection of the KG on either the set of attacks ($\mathbb{A}$) or the set of techniques ($\mathbb{T} \subseteq \mathbb{D}$).

Definition 11 (Attack similarity graph $KG_{\mathbb{A}}$). *$KG_{\mathbb{A}} = \langle \mathbb{A}, E_{\mathbb{A}}, J_{\mathbb{A}} \rangle$ is a unipartite weighted graph, where $(a_i, a_j) \in E_{\mathbb{A}}$ if $AT_{a_i} \cap AT_{a_j} \neq \emptyset$, and the edge weight $J_{\mathbb{A}}(a_i, a_j)$ is the Jaccard coefficient of the respective attack descriptions.*

We define the weighted techniques similarity graph ($KG_{\mathbb{T}} = \langle \mathbb{T}, E_{\mathbb{T}}, J_{\mathbb{T}} \rangle$) symmetrically as the one-mode projection of KG on $\mathbb{T}$.

4.4.1 Projected Techniques

We begin with the simplest approach that aggregates the link weights between the hypothesis provided by the analyst or the initial hypothesis generation and each of the techniques $t \in \mathbb{T}$. The relevance score $ProjT$ of the techniques is calculated as follows:

$$ProjT(a_{new}, t) = \sum_{t' \in AT^{init}_{a_{new}}} J_{\mathbb{T}}(t,' t). \tag{18}$$

4.4.2 Link Prediction on Projected Techniques

Although $ProjT$ takes into account the similarity between techniques in terms of the attacks associated with them, the topology of KG_T is not considered. We utilize several common link prediction measures to improve $ProjT$ by taking the neighborhoods of the techniques suggested by the analyst into account. Given $KG_{\mathbb{T}}$, we apply known LP measures such as $J_{\mathbb{A}}, AA_{\mathbb{A}}, K_{\mathbb{A}}$, and $PA_{\mathbb{A}}$ [59] on $KG_{\mathbb{T}}$. The likelihood of each technique $t \in T$ is calculated as follows:

$$LPProjT(a_{new}, t) = \sum_{t' \in AT^{init}_{a_{new}}} lp^{KG_T}(t,' t). \tag{19}$$

4.4.3 Link Prediction on Projected Attack (LPProjA)

Equation (20) is used to rank each attack $a \in \mathbb{A}$ and select the top k attacks, denoted as $\mathbb{A}_{top}$.

$$score_a = lp^{KG_{\mathbb{A}}}(a_i, a_{new}). \tag{20}$$

Similar to Equation (19), we use a range of LP measures to find the attack's most similar a_{new}. The score of each technique $t \in \mathbb{T}$ is calculated as

$$LPProjA(a_{new}, t) = \sum_{a \in \mathbb{A}_{top}: a \neq a_{new} \wedge t \in AT_a} score_a. \tag{21}$$

In Section 5.2, we present the results obtained with the $J_{\mathbb{A}}$ as the LP measure, since in preliminary experiments this method obtained the best results. Let $\Gamma_a = \{a' : (a, a') \in E_{\mathbb{A}}\}$ denote the neighbors of an attack a in the attack similarity graph $KG_{\mathbb{A}}$. The hypothesis based on $J_{\mathbb{A}}$ is

$$LPProjA(a_{new}, t) = \sum_{a \in \mathbb{A}_{top}: a \neq a_{new} \wedge t \in AT_a} \frac{|\Gamma_{a_{new}} \cap \Gamma_a|}{|\Gamma_{a_{new}} \cup \Gamma_a|}. \tag{22}$$

4.4.4 Projected Attack Techniques (ProjAT)

Intuitively, the more an arbitrary attack a is similar to a_{new}, the higher the chance that a_{new} uses the same AT as a. In this approach, we rank the techniques $t \in \mathbb{T}$ according to $J_{\mathbb{A}}$ between $AT^{init}_{a_{new}}$ and all AT_a that include t ($\{AT_a | a \in \mathbb{A} \wedge t \in AT_a\}$). Equation (23) is used to calculate the likelihood score for each technique $t \in \mathbb{T}$

$$ProjAT(a_{new}, t) = \sum_{a \in \mathbb{A}_{top}: a \neq a_{new} \wedge t \in AT_a} J_{\mathbb{A}}(AT_a, AT_{a_{new}}), \tag{23}$$

where $a \in \mathbb{A}_{top}$ are the top k attacks that obtain the highest score when applying $J_{\mathbb{A}}(a, a_{new})$.

4.4.5 Supervised Link Prediction (SupLP)

Here, we describe how to formulate LP as a supervised learning problem and apply a random forest classifier to predict the probability of a link in KG. First, we build a training set that consists of pairs of nodes $\{(u,v)|u \in \mathbb{A} \wedge v \in \mathbb{T}\}$. Then, we extract a variety of features for each pair following the methodology presented by Fire et al. [59]:

i) the nodes' topological attributes (e.g., degree, page rank, hubs, and authorities),
ii) neighborhood-based metrics (e.g., $J_{\mathbb{A}}$, $PA_{\mathbb{A}}$), and
iii) distance-based measures (e.g., shortest path length, number of shortest paths).

Next, given the initial hypothesis $AT^{init}_{a_{new}}$ we want to estimate the likelihood of a link existing between a_{new} and $t_j \in \mathbb{T}$ $(1 \leq j \leq |\mathbb{T}|)$ in KG. As in the model's training phase, we (1) extract the topological attributes and distance-based metrics of a_{new} and t_j from KG, and (2) use the projections $KG_{\mathbb{A}}$ and $KG_{\mathbb{T}}$ to obtain neighborhood-based metrics for a_{new} and t_j, respectively. Finally, we feed the extracted features for all (a_{new}, t_j) tuples $(1 \leq j \leq |\mathbb{T}|)$ to the trained model. The probability of the positive class (link exists) is used to rank the techniques.

On the one hand, building the SupLP model requires a large number of examples in the learning phase, while on the other hand, SupLP makes it possible to combine several features and be aware of wider contexts than the other four algorithms, which only take into account one feature.

4.5 The Expected Number of Techniques and Adaptive Hypothesis Refinement

Preliminary experiments on the initial inference of attack hypotheses and their refinement show that in cases where the real attack uses a large number of techniques the refinement may reduce the quality of AT^{init} (Figs. 8 and 10). We therefore introduce an adaptive refinement procedure. A dynamic threshold which states when to refine and when to rely on the AT^{init} is established based on the expected number of techniques related to the investigated attack.

Similar to the ih algorithms, we estimate the probability of each attack being the investigated attack. For $NB\text{-}C$ and $MLNB\text{-}C$ the probability of an attack is given by $P(a|COD)$, as specified in Equation (12). Given $P(a|COD)$ and the number of techniques related to a specific attack $AT(a)$, the number of expected techniques related to the investigated attack (denoted as $|AT'|$) can be calculated for the naive Bayesian methods as follows:

$$|AT'_{NB\text{-}C}| = |AT'_{MLNB\text{-}C}| = \sum_{a \in \mathbb{A}} P(a|COD) \cdot |AT(a)|. \tag{24}$$

The decision on the refinement of $TFIDF$ based AT^{init} is based on $TFIDF(a)$, which is similar to $TFIDF(t)$ and is defined as follows:

Definition 12 (Term frequency inverse document frequency of attacks). *Let $\mathbb{A}$ be the set of attacks in AttackDB; let $A \subseteq \mathbb{A}$ be the subset of relevant attacks having a path to a given obs; and let $a \in A$ be some relevant attack. Then IDF can be defined as follows:*

$$IDF(obs, a) = Log_e\left(\frac{|\mathbb{A}|}{|A|}\right). \tag{25}$$

The score of a is the sum of the $TFIDF(obs, a)$ values for all COD.

$$TFIDF(obs, a) = TF(obs, a) \cdot IDF(obs, a) \tag{26}$$

$$TFIDF(a) = \sum_{obs \in COD} TFIDF(obs, a), \tag{27}$$

where $TF(obs, a)$ is as in Definition 6.

$|AT'|$ can then be calculated for the $TFIDF$ method as the product of the probability for a and the number of related techniques $AT(a)$

$$|AT'_{TFIDF}| = \sum_{a \in \mathbb{A}} \frac{TFIDF(a)}{\sum_{a \in \mathbb{A}} TFIDF(a)} \cdot |AT(a)|. \tag{28}$$

The decision whether to refine an initial hypothesis or not depends on the expected number of techniques $|AT'|$ and a configurable adaptive refinement threshold $(arth)$. The refinement is only performed when $AT' \geq arth$.

4.6 Machine Learning Based Hypothesis Generation

We conclude the attack hypotheses generation methods with a description of a supervised machine learning (ML) approach. We consider a multi-label learning where CODs are instances and techniques are the labels. Every COD is a bag of observables. We used one-hot encoding to extract features as follows: *Domain names:* every generic top level domain (.com,. cn, etc.) is a binary feature. *Email addresses:* the email domains (gmail. com, yahoo.es etc.) were used. *IPv4:* we used class A (/8) subnets as features. *Ports, mutexes, and files* were used as-is for the one-hot encoding. Every COD is represented with 1,677 binary features. In this research we used two state-of-the-art ML algorithms XGBoost and Random Forest (RF).

5 Evaluation

5.1 Experimental Setup

For every attack $a \in \mathbb{A}$, we use AT^{init}, AT^{ref}, and AT^{*} to respectively denote the initial hypothesis, the refined hypothesis, and the ground truth of the utilized techniques.

We consider three strategies for constructing an initial hypothesis: (1) by a simulated human analyst H, (2) by automatic inference from IoCs, and (3) by automatic inference from telemetry data Tel. The initial hypotheses constructed according to these strategies are denoted as AT^{init}_{H}, AT^{init}_{IoC}, and AT^{init}_{Tel} respectively.

For H based AT^{init}, we assume an analyst attempting to make the right decisions regarding the investigated attack. However, there may be errors in the analyst's hypothesis regarding the attack due to a lack of knowledge or insufficient forensic evidence. To challenge the robustness of the hypotheses refinement algorithms we simulate two types of errors: false positives – selection of unrelated techniques $(AT^{init}_{H} \setminus AT^{*})$ and false negatives – omission of related techniques $(AT^{*} \setminus AT^{init}_{H})$. The errors are captured by two configurable parameters false positive rate (fpr_H) and false negative rate (fnr_H).

$$fpr_H = \frac{|AT_H^{init} \setminus AT^*|}{|\mathbb{T} \setminus AT^*|} \quad fnr_H = \frac{|AT^* \setminus AT_H^{init}|}{|AT^*|}. \tag{29}$$

For IoC and Tel based inferences of AT^{init}, we rely on the ih algorithms presented in Section 4.3. ML algorithms (XGBoost and RF) were used with their default configurations. Similar to fpr_H and fnr_H used to challenge the hypotheses refinement algorithms we challenge the whole hypothesis generation pipeline based on IoCs and Tel by introducing errors into the currently observed log data (COD). We generate various input data samples using various false positive (fpr_{COD}) and false negative rates (fnr_{COD}): Let COD^* denote the optimal set of observations. Let $\mathbb{OBS}$ denote all obs in AttackDB. fpr is the fraction of erroneous SDOs associated with the attack out of the total number of irrelevant SDOs

$$fpr_{COD} = \frac{|COD \setminus COD^*|}{|\mathbb{OBS} \setminus COD^*|}, \tag{30}$$

fnr_{COD} is the fraction of SDOs used by the attacker, which were not observed.

$$fnr_{COD} = \frac{|COD \setminus COD^*|}{|COD^*|}. \tag{31}$$

We run the evaluation for a wide range of error rates

$$fpr_H, fpr_{COD} \in \{0.0, 0.01, 0.03, 0.05, 0.07, 0.1, 0.2, 0.5\}$$
$$fnr_H, fnr_{COD} \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}.$$

In the rest of the paper, we will omit the subscripts of fpr, fnr when their use is apparent from the context.

5.1.1 Evaluation Metrics

In the evaluation, for each $a \in \mathbb{A}$, we measure the gap between the actual techniques in AT^* and the initial hypothesis AT^{init}, examine whether the hypothesis refinement algorithms were able to decrease the gap and specify the extent to which they were able to do so, and improve AT^{init}.

Note that the proposed approaches described in Section 4.4 rank all $d \in \mathbb{D}$ according to their likelihood to be associated with the attack, similar to search results in a typical recommender system. Therefore, we use average precision (AP) to evaluate the TTP inference approaches.

Given the actual techniques in AT^*, the initial hypothesis AT^{init}, and the refined hypothesis AT^{ref}, we compute the AP score for AT^{ref} and AT^{init} by comparing each of the hypotheses to AT^*. Next, we evaluate the improvement of the refined hypothesis in comparison to the analyst's hypothesis by calculating the difference between the AP score of AT^{init} and the AP score of AT^{ref}, i.e., we compute $AP(AT^*, AT^{ref}) - AP(AT^*, AT^{init})$.

5.1.2 Outline of the Experiment

In order to evaluate the hypothesis generation methods, we perform a leave-one-out cross-validation procedure (LOOCV). In every iteration of LOOCV we choose one attack from AttackDB as the test set and rely on the remaining AttackDB content to reconstruct the attack's technique set.

For each attack $a \in \mathbb{A}$, we evaluate each combination of initial hypothesis inference strategy, hypothesis refinement algorithm, and fpr and fnr values. The hypothesis inference strategy is determined by two parameters: $ih\_approach \in \{H, IoC, Tel\}$ and $ih\_impl \in \{TFIDF, NB\text{-}C, MLNB\text{-}C\}$. The hypothesis refinement algorithm procedure is likewise controlled by two parameters: $rh\_approach \in \{arh, rh\}$ and $ih\_impl \in \{ProjT, ProjAT, LPProjT, LPProjA, SupLP\}$.

For each attack, we also investigate whether to apply the refinement or not (ahr). In doing so, we determine a threshold ($arth$) for each a_{new}. $arth$ thereby maximizes the AP for AHG for $a \in \mathbb{A} | a \neq a_{new}$.

Algorithm 2 delineates one iteration of the LOOCV and its evaluation. Each iteration of the LOOCV begins, in line 1, with sampling the AttackDB according to $AT*$, fpr and fnr. The type of relevant SDOs is determined by $ih\_approach$.

Algorithm 2. Evaluation Procedure

```
    Input: $fpr, fnr, ih\_approach, ih\_impl, rh\_approach, rh\_impl, a$
    Output: $\Delta AP$
1:  $input$    $sample(COD^*, 1 - fnr) \cup sample(\mathbb{OBS} \setminus COD^*, fpr)$
2:  $RemoveAttack(a)$
3:  $AT^{init}$    GenInitHyp$(input, ih\_approach, ih\_impl)$
4:  $AT^{ref}$    GenRefHyp$(AT^{init}, rh\_approach, rh\_impl)$
5:  $RestoreAttack(a)$
6:  $AP^{ref}$    $AP(AT^*, AT^{ref})$
7:  $AP^{init}$    $AP(AT^*, AT^{init})$
8:  $\Delta AP$    $AP^{ref} - AP^{init}$
9:  return $\Delta AP$
```

Next, in line 2, the investigated attack a is removed from AttackDB. We remove the attack family, all IoCs connected to it, and all related behavioral reports. This step allows the evaluation procedure to assess the performance of AHG as if was facing zero-day exploits. In line 3, the GenInitHyp 3 procedure is called to generate AT^{init} according to the given sample of SDOs and the initial hypothesis generation approach. For approach H (i.e., simulated analyst), the input is a sample of techniques that simulates AT_H^{init}, thus it is returned by GenInitHyp as is (see lines 1-2 in Algorithm 3). For IoC and Tel based approaches, the input is either a set of IoCs or Tel, respectively.

In line 4, one of the rh algorithms from Section 4.4 is used to improve AT^{init} and return AT^{ref}. If the threshold is not reached, the adaptive refinement mechanism prevents the use of refinement algorithms so that $AT^{ref} = AT^{init}$. Finally, we restore a to AttackDB (line 5) and evaluate the performance of the ih and rh algorithms (lines 6-8).

The experiment is performed for each strategy (H, IoC, and Tel), combination of ih and rh algorithm, and for each combination of fpr and fnr.

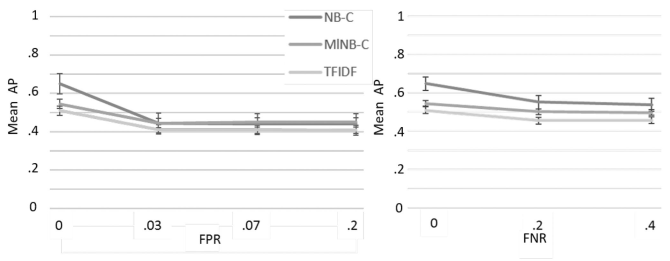

Fig. 6. Initial hypotheses from IoCs. (left) AP as a function of fpr_{COD} when $fnr_{COD} = 0$. (right) AP as a function of fnr_{COD} when $fpr_{COD} = 0$.

Algorithm 3. GenInitHyp

```
Input: $input, AttackDB, ih\_approach, ih\_impl$
Output: $ih\_impl(paths\_count, techs)$
1: if $ih\_approach == H$ then
2:     return $input$
3: else
4:     $paths\_count, techs \leftarrow CountPaths(input, AttackDB, ih\_approach)$
5:     return $ih\_impl(paths\_count, techs)$
```

Algorithm 4. GenRefHyp

```
Input: $AT^{init}, AttackDB, rh\_approach, rh\_impl$
Output: $rh\_impl(paths\_count, techs)$
1:  if $rh\_approach == arh$ then
2:    if $|AT'| \leq arth$ then
3:      $AT^{ref} \leftarrow rh(AttackDB, AT^{init})$
4:    else
5:      $AT^{ref} = AT^{init}$
6:    return $arh(paths\_count, techs)$
7:  else
8:    $AT^{ref} \leftarrow rh(AttackDB, AT^{init})$
9:    return $rh\_impl(paths\_count, techs)$
```

5.2 Results

5.2.1 Initial Hypothesis Generation Problem Results

Fig. 6 shows the performance of the initial hypotheses generation from IoCs as a function of the noise in the observed data. As expected the performance is the highest in absence of additional noise $fpr_{COD} = fnr_{COD} = 0$. Yet ih algorithms retain reasonable performance even for relatively high error rates in the observed data. In the following results we compare the performance of the algorithms averaged over the various error regimes.

Fig. 7 presents the mean AP for each ih algorithm (from left to right: simulated human analyst (H) based inference, automated inference of ih based on IoCs, automated inference of initial attack hypotheses AT^{init} based on telemetry and ih generation based on machine learning (ML)). For automated inference of hypotheses, the results for each algorithm described in Section 4.3 are provided. We also present a box-whisker plot to provide an overview of the behavior of the algorithms, highlighting the performance of the ih algorithms. As Fig. 7 presents the mean AP of refined attack hypotheses AT^{ref}, we use the identity function to represent the plain vanilla initial hypotheses (AT^{init}) and include their mean AP in the figure to show the effects of refinement.

The black line indicates the mean AP of random attack hypotheses. The expected AP for a baseline extracting 10 random techniques out of 190 is 0.077 (calculated according to Bestgen [60]). A mean AP of 0.08 was measured empirically. The $MLNB\text{-}C$ outperforms the other ih algorithms ($NB\text{-}C$ and $TFIDF$), providing AT^{init} with the highest AP of the ih algorithms.

Fig. 8 shows the dependence of the precision of the AT^{init} inference on the number of related techniques. In the upper part a histogram on the number of attacks is given; in the middle part of the figure, the mean AP is presented; while the bottom part presents the receiver operating characteristics area under the curve ($ROC-AUC$). As can be seen, the precision of attack hypotheses increases with an increasing number of related techniques, and the variance decreases with the number of related techniques. These effects can be seen for both AP and $ROC-AUC$. $MLNB-C$ shows superiority for attacks that are linked to a large number of techniques, although it is not as effective for generation of AT^{init} on attacks that are linked to a small number of techniques (compared to ML, $TFIDF$ and $NB-C$). The precision of AT^{init} does neither increase with an increasing number of related telemetries nor related IoCs. These results show that the number of IoCs and telemetries does not significantly influence the performance of AHG. Rather the specificity of COD is more important.

5.2.2 Hypothesis Refinement Problem Results

Fig. 9 presents the mean AP for H based AT^{init} (left) and AT^{ref} for each rh algorithm (top) and the improvements

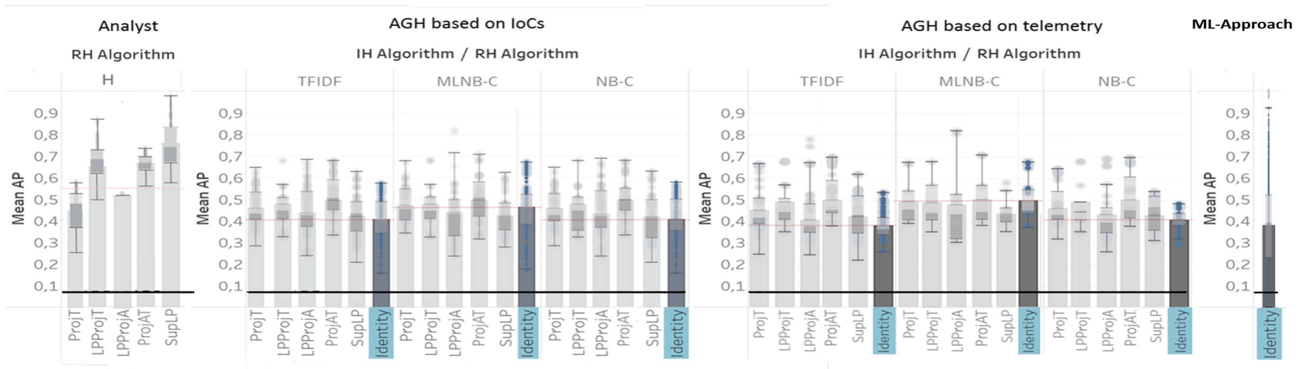

Fig. 7. Mean AP of analyst based (left,) automatic inference of attack hypotheses based on IoCs (middle-left) and Tel (middle-right), and ML based inference of attack hypotheses (right). Black line represents the AP of a random baseline.

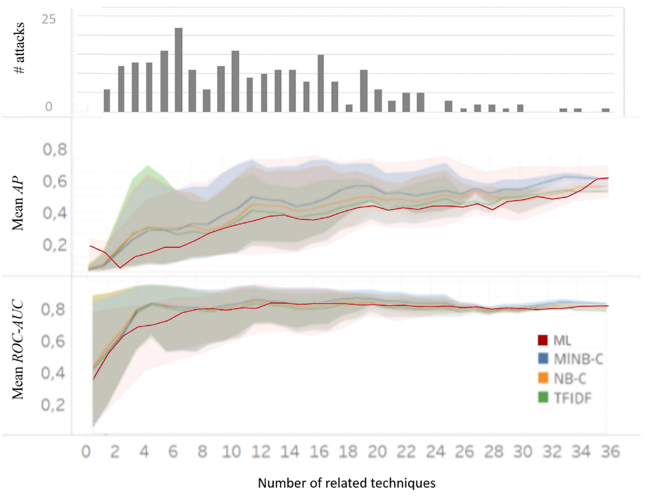

Fig. 8. The number of attacks (top), mean AP (middle) and $ROC\text{-}AUC$ (bottom) of ih algs. as a function of the number of related techniques.

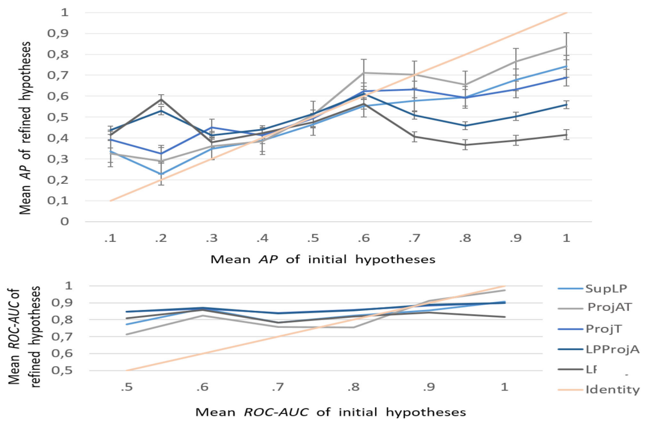

Fig. 10. Mean AP and $ROC \quad AUC$ of refined hypotheses depending on the precision of the initial hypotheses.

that can be reached through improving the analysts' hypotheses when employing AHG based refinement (bottom). Improvements of AT^{init} are highlighted in green, while deterioration is indicated in red. As can be seen, each rh algorithm has the potential for improving the AT^{init} for specific combinations of fnr and fpr. However, especially for combinations of both, low fnr and low fpr, where the simulated human analyst (H) is able to generate very precise AT^{init}, there may be a deterioration of the precision of attack hypotheses. The extent of the deterioration of AT^{init}' precision is highly dependent on the rh algorithm employed. $SupLP$ decreases the precision of AT^{init} the lowest for combinations of low fnr and low fpr (AT^{init} that is close to AT^{*}), while $LPProjT$ performs the poorest under these circumstances. Furthermore, $SupLP$ achieves the best results for refining H based inferences of AT^{init} for reasonable combinations of fnr and fpr. However, it is shown to work worse for combinations of high fnr and high fpr than $LPProjA$ and $ProjAT$.

Fig. 7 presents the results for automatic hypothesis generation for each rh algorithm (Section 4.4) applied to AT^{init} crafted relying on ih algorithms, based on IoCs (left) and Tel (right). We highlight the identity function (as a plain vanilla initial hypothesis (AT^{init}) without refinement) to disentangle the evaluation of the of hypothesis refinement.

Both ML algorithms achieved similar performance mean AP (std) of 0.386 (0.2) and 0.384 (0.2) for XGBoost and RF respectively. Hence we show only one box-plot in Fig. 7. The std. of AP of the ML models is two times higher than the std. of AP of the other approaches (TFIDF:0.09, MLNB-C: 0.11, NB-C:0.09). We attribute this behavior to the ability of ML algorithms to focus on indicative observables when they are available. In contrast, in the NB based approaches, the contribution of the most indicative observables may be diluted by the Laplace smoothing.

When the attack hypotheses are refined (if AT^{init} gets refined), $TFIDF$ and $NB-C$ can lead to comparable precise hypotheses than $MLNB-C$. What stands out is that when applying Tel based inference of attack hypotheses, the precision of $MLNB-C$ increases, while a slight deterioration (compared to IoC based inference of attack hypotheses) can be seen for both $TFIDF$ and $NB-C$. Furthermore, $ProjAT$ is shown to be the most efficient for refining automatically generated AT^{init}. Regarding the behavior of rh algorithms, it can be seen that $SupLP$ is ineffective for automatically generated AT^{init} (relying on ih algorithms) and obtains the worst precision of AT^{ref}.

Fig. 10 shows the efficiency of rh algorithms depending on the quality (precision) of AT^{init}. This figure presents the behavior for each rh algorithm. While it can be seen that the rh algorithms react differently to the precision of AT^{init}, rh algorithms prove to be most useful when the accuracy of AT^{init} is low.

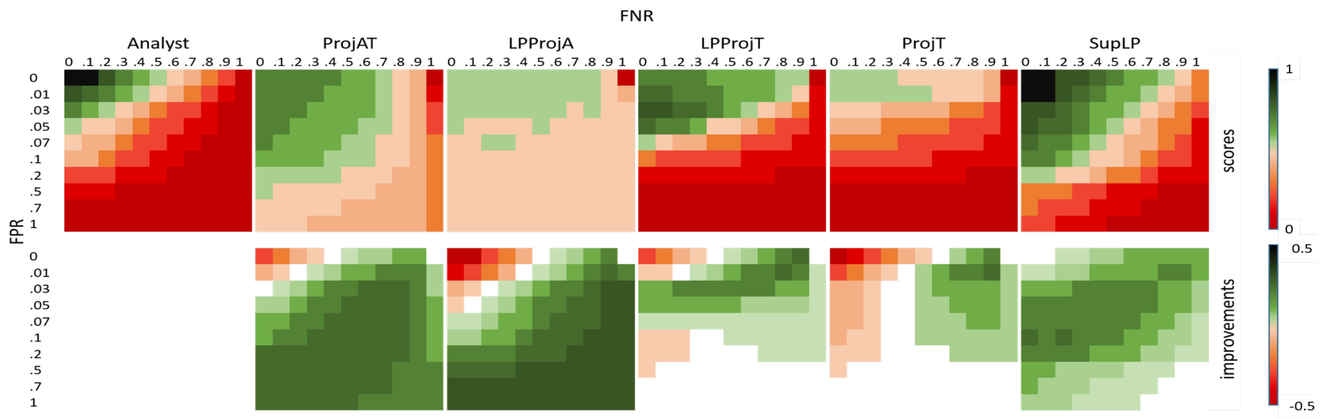

Fig. 9. Mean AP for initial hypothesis generation and refinement for analyst based initial inferences.

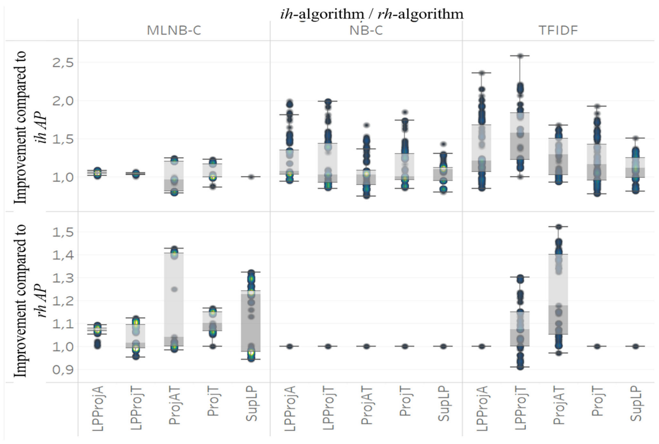

Fig. 11. Improvements of mean AP when applying the arh-procedure.

However, for an AP of AT^{init} of around 0.5 and higher, rh algorithms are more likely to deteriorate the accuracy of AT^{init}. This is consistent with the results obtained for analyst crafted hypotheses (see Fig. 9). It is also shown that, the behavior of $LPProjA$ and $LPProjT$ is comparable.

5.2.3 Adaptive Refinement

Finally, we address the limitations of link prediction in cases of accurate AT^{init} using the adaptive refinement method (arh algorithm) described in Section 4.5. Fig. 11 shows that the arh algorithm improves the precision of AT^{init} in the vast majority of cases. Moreover, arh based refinement does not lead to a deterioration of the mean AP for either combination of ih and rh algorithms). For each ih algorithm, the optimal adaptive refinement threshold ($arth$) is calculated on the basis of the remaining data (according to the LOOCV procedure), leading to a refinement in 40% of the cases for $MLNB-C$ and approximately 84% for $TFIDF$. In the case of the $NB-C$, the precision of AT^{ref} is always better than the AT^{init}; thus, the arh procedure activates the rh algorithm in every case.

5.3 Discussion

We evaluated AHG based on a LOOCV procedure. Therefore, we consider AHG to be applicable for the analysis of novel (yet unobserved) attacks (e.g., zero-day exploits).

For analyst (H) based initial hypotheses ($AT^{init}_{a_{new}}$), hypothesis refinement is able to increase the precision of attack hypotheses for a wide range of combinations of fnr and fpr. Experimental results show the superiority of $ProjAT$ and $SupLP$ for refining H based AT^{init}. The supremacy of $ProjAT$ for refining H based AT^{init} over the other rh algorithms indicates that (1) relying on similar attacks is a good approach for hypothesizing about new attacks, and (2) the Jaccard's/ Tanimoto coefficient between attack descriptions is a good measure for identifying similar attacks. Furthermore, $SupLP$ based hypothesis refinement can be integrated to the process of attack hypothesis generation within the threat hunting cycle for H generated AT^{init} without the risk of a significant deterioration of the precision of attack hypotheses. Evaluations of the results obtained for H based attack hypotheses need to consider that the real performance of H can differ from analyst to analyst. Furthermore, we simulated the accuracy of AT^{init} crafted by H rather than assessing the accuracy of H based attack hypotheses empirically. Also, we considered an analyst with infinite computational resources who produces a reasonable AT^{init} given a set of observables. If there are multiple reasonable AT^{init}, we assume that the analyst randomly selects one of those. In this study we did not account for the possibility that the structure of the database (AttackDB) could cause difficulties for a human analyst (e.g., due to the large volume of obs). Consequently, it is likely that we systematically overestimated the precision of H based AT^{init} when relying on the simulation approach.

For hypothesis refinement, our results show that $LPProjA$ provides mean AP of around 0.5, even for high fpr and high fnr. Since some of the attack patterns are very common, $LPProjA$'s performance (measured in terms of AP and $ROC-AUC$) is reasonably high. Thus, recommending popular techniques can have significant value, especially when there is a lot of uncertainty, i.e., when there is no context to the attack or observations (COD) are weak. Furthermore, $ProjAT$ and $SupLP$ show the best results for reasonable combinations of fnr and fpr, highlighting their potential use for refining hypotheses.

Additionally, the results raise some interesting insights for establishing fully automated hypothesis generation that combines the use of ih and rh algorithms. Since the ih algorithms seem to be robust, even for combinations of high fnr and high fpr, the effectiveness of using some rh algorithms seems to be questionable. For most observed cases, the use of $LPProjT$ and $LPProjA$ resulted in a deterioration of the precision of AT^{init}. This indicates that there is a need for an effective combination of rh and ih algorithms to craft attack hypotheses with high precision.

6 Conclusion and Future Work

In this work we proposed a comprehensive multi-level threat knowledge base derived from multiple open-source threat intelligence sources called AttackDB. AttackDB can be used to generate attack hypotheses which include high-level descriptions of the investigated attack. We focus on the automated inference of adversarial techniques from observable artifacts found within the attacked systems relying on multiple initial attack hypothesis generation algorithms. The inference is demonstrated using a large collection of VirusTotal behavioral reports. Further, we employ a variety of techniques inspired by recommender systems to refine initial attack hypotheses suggested by one of the algorithms or an analyst. We show that such refinement works best when the number of techniques used in the attack is sufficiently large. Based on this insight we developed an heuristic to decide whether to refine an initial hypothesis or not. Automated attack hypothesis generation relying on this adaptive hypothesis refinement procedure based on the expected number of techniques works the best, achieving a mean AP greater than 0.5 and a $ROC-AUC$ above 0.8.

Future research on evaluating the performance of the presented algorithms empirically against professional security analysts or against rule-based TTP inference such as HOLMES [32] would highlight the pros and cons of automated CTI driven TTP inference. The former will assess the usefulness of the AHG algorithms in real security operations center settings with human in the loop. The latter will be possible once rule base for TTP inference will be expanded to cover a large fraction of ATT&CK techniques and allow inference based on VirusTotal behavioral reports.

MITRE ATT&CK does not include information about the order of adversarial techniques employed by malware. The same is true for VirusTotal behavioral reports. As a result, AttackDB does not include information about the sequences of attack steps. Once this limitation is removed by the major CTI sources, *AHG* can be augmented to consider the order of techniques in an *IoA*, similar to what has taken place in the area of expert based TTP inference.

While the high-level attack hypotheses provided in form of a collection of adversarial techniques are useful for analysis and reporting, future development is required to close the loop with data collection based on these hypotheses [61]. Such automation is possible using analytics such as those provided in the MITRE Cyber Analytic Repository.[3]

References

[1] M. Bromiley, "Threat intelligence: What it is, and how to use it effectively," *SANS Inst. InfoSec Reading Room*, vol. 15, 2016.

[2] V. Mavroeidis and S. Bromander, "Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence," in *Proc. Eur. Intell. Secur. Informat. Conf.*, 2017, pp. 91–98.

[3] A. Elitzur, R. Puzis, and P. Zilberman, "Attack hypothesis generation," in *Proc. Eur. Intell. Secur. Informat. Conf.*, 2019, pp. 40–47.

[4] L. Dekel, P. Zilberman, R. Puzis, U. Dardik, and A. Elitzur, "Attack DB OTX-XFORCE-VT," 2021. [Online]. Available: https://dx.doi.org/10.21227/f74t-gh08

[5] R. Daszczyszak, D. Ellis, S. Luke, and S. Whitley, "TTP-based hunting," MITRE Corp, McLean VA, Tech. Rep., 2019.

[6] D. Chismon and M. Ruks, "Threat intelligence: Collecting, analysing, evaluating," MWR InfoSecurity, Tech. Rep., 2015.

[7] S. E. Jasper, "U.S. cyber threat intelligence sharing frameworks," *Int. J. Intell. CounterIntelligence*, vol. 30, no. 1, pp. 53–65, 2017. [Online]. Available: https://doi.org/10.1080/08850607.2016.1230701

[8] I. Tyler Technologies, "A guide to cyber threat hunting," 2018. [Online]. Available: https://www.tylertech.com/services/ndiscovery/nDiscovery-Threat-Hunting.pdf

[9] G. Settanni, Y. Shovgenya, F. Skopik, R. Graf, M. Wurzenberger, and R. Fiedler, "Acquiring cyber threat intelligence through security information correlation," in *Proc. IEEE 3rd Int. Conf. Cybern.*, 2017, pp. 1–7.

[10] D. Shackleford, "Who's using cyberthreat intelligence and how," SANS Institute, 2015.

[11] M. Landauer, F. Skopnik, M. Wurzenberger, W. Hotwagner, and A. Rauber, "A framework for cyber threat intelligence extraction from raw log data," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 3200–3209.

[12] C. Wheelus, E. Bou-Harb, and X. Zhu, "Towards a Big Data architecture for facilitating cyber threat intelligence," in *Proc. 8th IFIP Int. Conf. New Technol., Mobility Secur.*, 2016, pp. 1–5.

[13] S. Samtani, R. Chinn, H. Chen, and J. F. Nunamaker Jr ., "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence," *J. Manage. Inf. Syst.*, vol. 34, no. 4, pp. 1023–1053, 2017. [Online]. Available: https://doi.org/10.1080/07421222.2017.1394049

[14] S. Bromander, M. Swimmer, M. Eian, G. Skjotskift, and F. Borg, "Modeling cyber threat intelligence," in *Proc. 6th Int. Conf. Inf. Syst. Secur. Privacy*, 2020, pp. 273–280.

[15] J. DeCianno, "Indicators of attack vs. indicators of compromise," CrowdStrike, 2014.

[16] McAfee, "Mcafee threat intelligence exchange (datasheet)," retrieved 2019. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/data-sheets/ds-threat-intelligence-exchange.pdf

[17] T. Plona, K. Maxwell, and B. Varni, "Threat intelligence in splunk," 2017. [Online]. Available: https://conf.splunk.com/files/2017/slides/do-you-really-know-my-adversaries-prove-it.pdf

[18] C. Sauerwein, C. Sillaber, A. Mussmann, and R. Breu, "Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives," in *Proc. der 13. Internationalen Tagung Wirtschaftsinformatik*, 2017, pp. 837–851.

[19] Y. Zhao, B. Lang, and M. Liu, "Ontology-based unified model for heterogeneous threat intelligence integration and sharing," in *Proc. IEEE 11th Int. Conf. Anti counterfeiting, Secur. Identification*, 2017, pp. 11–15.

[20] Z. Iqbal and Z. Anwar, "Ontology generation of advanced persistent threats and their automated analysis," *NUST J. Eng. Sci.*, vol. 9, no. 2, pp. 68–75, 2016.

[21] A. de Melo e Silva, J. J. Costa Gondim, R. de Oliveira Albuquerque, and L. J. García Villalba, "A methodology to evaluate standards and platforms within cyber threat intelligence," *Future Internet*, vol. 12, no. 108, pp. 1–23, 2020.

[22] S. Alonso, "Cyber threat hunting (1): Intro," Jan. 2016. [Online]. Available: https://cyber-ir.com/2016/01/21/cyber-threat-hunting-1-intro/

[23] H. Rasheed, A. Hadi, and M. Khader, "Threat hunting using grr rapid response," in *Proc. Int. Conf. New Trends Comput. Sci.*, 2017, pp. 155–160.

[24] I. Sqrrl Data, "A framework for cyber threat hunting," 2016. [Online]. Available: https://sqrrl.com/media/Framework-for-Threat-Hunting-Whitepaper.pdf,https://www.threathunting.net/files/framework-for-threat-hunting-whitepaper.pdf

[25] McAfee, "Mcafee investigator: Transform analysts into expert investigators," retrieved 2019. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/data-sheets/ds-investigator.pdf

[26] R. M. Lee and R. T. Lee, "SANS 2018 threat hunting survey results," SANS Institute, Sep. 2018.

[27] V. Mavroeidis and A. Jøsang, "Data-driven threat hunting using sysmon," 2018.

[28] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 341–351, Apr.–Jun. 2020.

[29] S. Ranveer and S. Hiray, "Comperative analysis of feature extraction methods of malware detection," *Int. J. Comput. Appl.*, vol. 120, no. 5, pp. 1–7, 2015.

[30] B. E. Strom et al., "Finding cyber threats with att&ck^{TM}-based analytics," The MITRE Corporation, McLean, VA, Tech. Rep. MTR170202, 2017.

[31] N. Polatidis, E. Pimenidis, M. Pavlidis, S. Papastergiou, and H. Mouratidis, "From product recommendation to cyber-attack prediction: Generating attack graphs and predicting future attacks," *Evolving Syst.*, vol. 11, pp. 479–490, 2020.

[32] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: Real-time apt detection through correlation of suspicious information flows," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 1137–1152.

[33] A. AlEroud and G. Karabatis, "Methods and techniques to identify security incidents using domain knowledge and contextual information," in *Proc. IEEE/IFIP Symp. Integr. Netw. Serv. Manage.*, 2017, pp. 1040–1045.

[34] S. M. H. Fard, H. Karimimpour, A. Dehghantanha, A. N. Jahromi, and G. Srivastava, "Ensemble sparse representation-based cyber threat hunting for security of smart cities," *Comput. Electronical Eng.*, vol. 88, no. 106825, pp. 1–13, 2020.

[35] P. Bhatt, E. T. Yano, and P. Gustavsson, "Towards a framework to detect multi-stage advanced persistent threats attacks," in *Proc. IEEE 8th Int. Symp. Service Oriented Syst. Eng.*, 2014, pp. 390–395.

[36] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues Inf. War fare Secur. Res.*, vol. 1, no. 1, 2011, Art. no. 80.

[37] K. Wafula and Y. Wang, "CARVE: A scientific method-based threat hunting hypothesis development model," in *Proc. IEEE Int. Conf. Electro Inf. Technol.*, 2019, pp. 1–6.

[38] P. Giura and W. Wang, "A context-based detection framework for advanced persistent threats," in *Proc. Int. Conf. Cyber Secur.*, 2012, pp. 69–74.

[39] S. Rubinshtein and R. Puzis, "Modeling and reconstruction of multi-stage attacks," in *Proc. IEEE Int. Conf. Softw. Sci., Technol. Eng.*, 2016, pp. 135–137.

[40] S. Lee, H. Cho, N. Kim, B. Kim, and J. Park, "Managing cyber threat intelligence in a graph database: Methods of analyzing intrusion sets, threat actors, and campaigns," in *Proc. Int. Conf. Platform Technol. Service*, 2018, pp. 1–6.

[41] Y. Gao, X. Li, J. Li, Y. Gao, and N. Guo, "Graph mining-based trust evaluation mechanism with multidimensional features for large-scale heterogeneous threat intelligence," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 1272–1277.

3. https://car.mitre.org/

[42] H. Xiao, Z. Xing, X. Li, and H. Guo, "Embedding and predicting software security entity relationships: A knowledge graph based approach," in *Proc. Int. Conf. Neural Inf. Process.*, 2019, pp. 50–63.

[43] J. S. Garrido, D. Dold, and J. Frank, "Machine learning on knowledge graphs for context-aware security monitoring," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience*, 2021, pp. 55–60.

[44] P. Najafi, A. Mühle, W. Pünter, F. Cheng, and C. Meinel, "MalRank: A measure of maliciousness in SIEM-based knowledge graphs," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, 2019, pp. 417–429.

[45] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. N. Venkatakrishnan, "POIROT: Aligning attack behavior with kernel audit records for cyber threat hunting," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1795–1812.

[46] S. Qamar, Z. Anwar, M. A. Rahman, E. Al-Shaer, and B.-T. Chu, "Data-driven analytics for cyber-threat intelligence and information sharing," *Comput. Secur.*, vol. 67, pp. 35–58, 2017.

[47] R. Riesco and V. Villagrá, "Leveraging cyber threat intelligence for a dynamic risk framework," *Int. J. Inf. Secur.*, vol. 18, pp. 715–739, 2019.

[48] I. Horrocks et al., "SWRL: A Semantic Web rule language combining OWL and RuleML," *W3C Submission*, vol. 21, no. 79, pp. 1–31, 2004.

[49] B. Ulicny, J. Moskal, M. M. Kokar, K. Abe, and J. K. Smith, *Infer ence and Ontologies*, vol. 62. Cham, Switzerland: Springer, 2014, pp. 167–199. [Online]. Available: https://doi.org/10.1007/978–3-319-11391-3_9

[50] G. Antoniou and F. Van Harmelen, "Web ontology language: OWL," in *Handbook on Ontologies*, Berlin, Germany: Springer, 2004, pp. 67–92.

[51] D. Bianco, "The pyramid of pain," 2014. [Online]. Available: http://detect-respond.blogspot.nl/2013/03/the-pyramid-of-pain.html

[52] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 616–623.

[53] T. T. Tanimoto, "Elementary mathematical theory of classification and prediction," 1958.

[54] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[55] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Netw.*, vol. 25, no. 3, pp. 211–230, 2003.

[56] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[57] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[58] B. M. Sarwar et al., "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[59] M. Fire, L. Tenenboim-Chekina, R. Puzis, O. Lesser, L. Rokach, and Y. Elovici, "Computationally efficient link prediction in a variety of social networks," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 1, pp. 1–25, 2014.

[60] Y. Bestgen, "Exact expected average precision of the random baseline for system evaluation," *Prague Bull. Math. Linguistics*, vol. 103, no. 1, 2015, Art. no. 131.

[61] L. Dekel, I. Leybovich, P. Zilberman, and R. Puzis, "MABAT: A multi-armed bandit approach for threat-hunting," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 477–490, Oct. 2022.