MABAT: A Multi-Armed Bandit Approach for Threat-Hunting

Liad Dekel, Ilia Leybovich, Polina Zilberman, and Rami Puzis

***Abstract*— Threat hunting relies on cyber threat intelligence to perform active hunting of prospective attacks instead of waiting for an attack to trigger some pre-configured alerts. One of the most important aspects of threat hunting is automation, especially when it concerns targeted data collection. Multi-armed bandits (MAB) is a family of problems that can be used to optimize the targeted data collection and balance between exploration and exploitation of the collected data. Unfortunately, state-of-the-art policies for solving MAB with dependent arms do not utilize the detailed interrelationships between attacks such as telemetry or artifacts shared by multiple attacks. We propose new policies, one of which is theoretically proven, to prioritize the investigated attacks during targeted data collection. Experiments with real data extracted from VirusTotal behavior reports show the superiority of the proposed techniques and their robustness in presence of noise.**

***Index Terms*— Digital forensics, computer security, threat hunting, threat intelligence, reinforcement learning.**

I. Introduction

THREAT hunting is one of the most important activities performed by security operation centers today [1]. Relying on constant feed of cyber threat intelligence (CTI), hunting focuses on proactive collection of forensic evidence in order to investigate latent attacks [2], [3], [4]. Actively searching for evidence of malicious activity is preferred over the traditional "sit and wait" approaches. Analysts use variety of security analytics and rely on their experience and intuition to make the right hypotheses and perform targeted search for evidence to support or refute these hypotheses.

The decision which evidence to search for is not a trivial one. Feeding the security analytics with irrelevant information decreases their accuracy and increases the number of irrelevant alerts. This in turn, makes the analyst work harder to "find a needle in a haystack". Focused, targeted data collection decreases the amount of irrelevant information eventually displayed to the analyst. It also significantly reduces resources spent on data collection.

eXtended Detection and Response (XDR) solutions and Security Orchestration Automation and Response (SOAR) systems can partially automate the data collection process [5]. But some artificial intelligence is required to automatically decide when to investigate a promising lead, and when to look around exploring seemingly unrelated artifacts. A classical model for studying exploration vs exploitation trade-offs is the multi-armed bandit (MAB) problem [6], [7]. MAB is a theoretically solid framework but it requires adjustments to fully utilize the power of CTI.

In this paper we model targeted data collection as a variant of MAB. We augment MAB policies to take into account the variety of artifacts shared among different attacks and use this information to significantly improve targeted data collection. The contributions of this paper are twofold:

1. Theoretical: We formulate the task of targeted data collection as the Multi Shared-Arms Bandits problem (MSAB), a special case of Combinatorial MAB [8]. We propose a shared-arms (SA) adaptation of classic MAB policies and demonstrate it using ϵ-Greedy, UCB1, Thompson sampling [9] and KL-UCB [10]. We show that the SA variant of the classic UCB1 policy [11], [12], termed SAUCB, is in fact a degenerated policy for Combinatorial MAB with a slightly different constant.

2. Empirical: Evaluated based on multi-level CTI knowledge base, containing more than 50K behavioral reports from VirusTotal, the shared-arms (SA) policies are superior to benchmark policies both in the relevance of the collected artifacts and in the time it takes to home on the most relevant attack. We also show that prioritizing artifacts shared by most of the attacks increases performance.

These contributions are significant because they (1) advance the level of automation during threat hunting, (2) increase the effectiveness of data collection under constrained resources, and (3) help focusing on the most relevant artifacts even when the CTI about a novel attack is missing.

The rest of the paper is structured as follows: In § II we provide the Background on CTI, hunting and MAB. In § III we define threat model and the MSAB problem. We present collection policies and the regret bound of SAUCB in § IV. The CTI knowledge base is described in § V and in § VI we evaluate the collection policies using this knowledge base

Manuscript received 11 December 2021; revised 23 July 2022 and 28 August 2022; accepted 27 September 2022. Date of publication 17 October 2022; date of current version 9 December 2022. This work was supported in part by the National Research Foundation Singapore (NRF) under Grant NRF2016NCR-NCR001-006 and in part by the U.S.-Israel Energy Center managed by the Israel-U.S. Binational Industrial Research and Development (BIRD) Foundation. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kai Zeng. *(Corresponding author: Rami Puzis.)*

Liad Dekel, Ilia Leybovich, and Rami Puzis are with Cyber@BGU Laboratories, Beer-Sheva 8410501, Israel, and also with the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel (e-mail: liadd@post.bgu.ac.il; puzis@bgu.ac.il).

Polina Zilberman is with Cyber@BGU Laboratories, Beer-Sheva 8410501, Israel.

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2022.3215010, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2022.3215010

along with adversarial analysis of the presented methods. We compare and contrast to similar MAB solutions in § VII. The findings are summarized in § VIII together with future work.

II. Background

A. Cyber Threat Intelligence

Cyber Threat Intelligence (CTI) is structured actionable information describing adversaries, their motives, goals, capabilities, resources, and tactics. CTI includes evidence-based knowledge in the form of measurable events and the context for their interpretation. CTI increases the ability of the analyst to recognize relevant threats and respond to them in a timely manner [2], [13]. It is a powerful mean to increase efficiency of various security solutions, such as intrusion detection, response, real time analytics, forensic investigation, and threat hunting.

Since no organization possesses a complete understanding of the threat landscape, the importance of CTI lies with the ability to share threat information among partners in a machine-to-machine manner. By sharing the who, what, where, how, and when of malicious activities, targeted organizations obtain a holistic view of the threat landscape, thus increasing their cyber security readiness [13].

According to a survey among various cyber security and IT management roles presented by Shackleford [14], 48% of the respondents say their use of CTI has reduced incidents through early prevention, and 51% said they are able to respond more quickly to incidents. Methods for CTI analysis can be applied to provide SOC analysts with a list of related information, supporting them in the decision making process while handling cyber incidents [15], [16].

In an effort to formalize a standard language for sharing CTI, DHS Office of Cybersecurity and Communications funded MITRE to develop the Structured Threat Information eXpression (STIX) language.[1] STIX covers the entire range of cyber security concepts, including observables, indicators of compromise (IOC), attack patterns, tools, malware, threat actors, course of action, and other. A STIX element is denoted as STIX Domain Object (SDO). SDOs such as observable and IOCs are considered low-level SDOs, while SDOs such as attack patterns, tools and threat actors are considered high-level SDOs. Additional CTI languages include Malware Information Sharing Platform (MISP),[2] as well as proprietary languages and ontologies developed by McAfee [17] and IntelGraph by Accenture [18].

B. Threat Hunting

Generally speaking, threat hunting includes series of investigative steps for confirming or refuting attack hypotheses. This process may include forensic investigations and various analytics whose objective is inferring the attack steps and collecting artifacts. Cyber security experts are divided on the exact stages of the threat hunting cycle and on its *re*-active or *pro*-active nature. On 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 [19]. 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 [4], [20]. The main difference between the proactive and reactive threat hunting is the trigger for the investigation. Proactive threat hunting relies on *threat intelligence* to actively search for *potentially* malicious behavior. Reactive threat hunting involves forensic investigation of potentially malicious behavior *in response to* alerts.

Security Orchestration, Automation and Response (SOAR) systems perform autonomous response to security alerts or suspicious indicators relying on prescribed workflows, thus providing automation of hypotheses testing. The most sophisticated solutions utilize interactive guidebooks to help the analysts focus on what is important as they scope and assess the attack hypotheses [21], [22], [23], [24]. In this paper we present an automation of the transition between hypothesis generation and hypothesis testing, which is facilitated by the automatic generation of playbooks.

C. Multi-Armed Bandit

The multi-armed bandit (MAB) is a classic reinforcement learning problem that exemplifies the exploration-exploitation trade-off dilemma. Consider a gambler in front of k slot-machines who has to decide which machines to play, how many times and in which order. The goal of the gambler is to maximize his total-expected reward [6]. MAB problem consists of k arbitrary reward distributions, with expected rewards $\mu_1, \ldots, \mu_k$, each related to a slot-machine. MAB illustrates the fundamental difficulty of decision making in the face of uncertainty, since the reward distributions are unknown to the gambler. In the classical problem, each arm-pull may result in a gain of some reward [25]. We consider the binary case where each arm pull ends as a win or a loss, i.e., a gain of 1 or 0 (no gain), respectively.

In MAB a player tries to make successive arm-pulls in order to collect as much reward as possible. On each turn, the player selects a machine, pulls its arm and collects some reward according to the machine's reward distribution. MAB policy is a function that determines which machine/s the player should select on each turn to maximize the eventual reward.

If the reward distributions were known, the optimal policy would be to constantly choose the machine with the highest expected reward. Since the expected rewards are unknown, an exploration-exploitation trade-off is considered. The player has to choose between following what seems to be the best choice ("exploiting") or testing some alternative ("exploring"), hoping to discover a choice that beats the current best choice hence avoiding getting stuck in a local maxima [26].

MAB has a long history. Many variants of the problem were studied and many policies were proposed [27]. In § III we will define the Multi Shared-Arms Bandits problem (MSAB), which is closely related to MAB with dependent arms [28].

[1] https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/stix-20-finish-line

[2] https://www.misp-project.org/

A popular success criterion of MAB policies is the expected loss due to not choosing the optimal arm, referred to as the *regret* [11]. Theoretical guarantees on the regret of MAB policies allow managing the risk of making wrong decisions under uncertainty. The literature distinguishes between instance-dependent and instance-independent regret. We derive instance-dependent regret bounds utilizing the Combinatorial MAB framework [8].

Out of the wide range of MAB algorithms, we chose to focus on the following four policies:

1) ϵ-Greedy: The family of policies called ϵ-Greedy manifest trade-off between exploration and exploitation captured by the parameter ϵ [29]. At every iteration, the policy selects a random arm with the probability of ϵ or otherwise the arm with the highest average reward. We denote the average reward of arm j as $\overline{X}_j$.

2) Thompson Sampling (TS) [9]: This algorithm samples from the learned reward distributions of the arms and chooses the arm with the largest sampled reward. Arms with higher average rewards are chosen more frequently. We use TS policy for Bernoulli reward distributions which relies on the number of successful (s_j) and unsuccessful (f_j) pulls of arm j.

3) UCB1: A family of policies known as *upper confidence bound* (UCB) exemplify the principle of optimism under uncertainty to provide bounded regret [11]. UCB1 relies on the total number of arm pulls (n), the average reward of each arm ($\overline{X}_j$), and their pull counters (n_j) to select arms that maximize: $\overline{X}_j + \sqrt{\frac{2\ln n}{n_j}}$.

4) KL-UCB: Based on the Kullback-Leibler divergence, Garivier and Cappé [10] proposed a MAB algorithm with strictly better theoretical guarantees than UCB1. KL-UCB mainatins the pull counters (n_j) and the total reward (s_j) for each arm j.[3]

5) Combinatorial UCB (CUCB): Chen et al. [8] define a MAB problem where in every iteration a set S of arms, called a *super-arm*, are pulled altogether. The total aggregated reward $R(S)$ of pulling a super-arm S is derived from the rewards of the individual arms X_j. It may be a simple summation $R(S) = \sum_{j\in S} X_j$ or a different aggregation function. Let μ_j be the expected reward of arm j. The expected aggregated reward $r_\mu(S) = \mathbb{E}(R(S))$ should have two properties: *Monotonicity:* $\forall_j, \mu_j \le \mu'_j \implies r_\mu(S) \le r_{\mu'}(S)$; and *Bounded smoothness:* there exists strictly increasing fucntion $f(\cdot)$ such that $\max_{j\in S} |\mu_j - \mu'_j| \le \Lambda \implies |r_\mu(S) - r_{\mu'}(S)| \le f(\Lambda)$.

For many aggregation functions, finding the set S that maximizes $r_\mu(S)$ given the individual expected rewards $\{\mu_j\}$ is NP-hard. Thus Chen et al. assume an (α, β)-approximation oracle that takes the individual expected rewards $\{\mu_j\}$ and outputs a super-arm that with probability β generates an α fraction of the optimal expected aggregated reward. According to the CUCB policy, the beliefs regarding the individual expected rewards are updated as follows:

$$\hat{\mu}_{j,t} = \overline{X}_{j,t} + \sqrt{\frac{3\ln t}{2n_{j,t}}} \tag{1}$$

[3]We assume rewards of 0 (failure) and 1 (success).

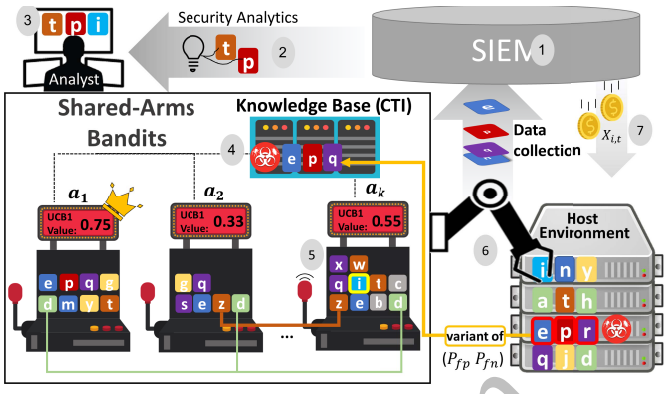


Fig. 1. Collection of artifacts during threat hunting modeled as MAB problem employed in a SOAR system.

where t is current round, $\overline{X}_{j,t}$ is the average reward obtained from arm j so far, $n_{j,t}$ is the number of times arm j has been played. In the rest of this paper we may add the subscript t to various quantities to indicate the relevant algorithm's iteration. According to CUCB whenever a super-arm S is played all the respective arms $j \in S$ are played.

We will show that the proposed targeted data collection approach for threat hunting is a spacial case of CMAB problem and show that a shared arms (SA) variant of UCB1 is in fact CUCB with a slightly different constant. Chen et al. provide a framework for deriving CUCB regret bounds for various applications. We elaborate on the regret bounds respective to our threat hunting application in § IV-D.

The main differences between CMAB and MSAB are threefold: (1) In MSAB every super-arm may be pulled only once. (2) The total aggregated reward $R(S)$ is the maximum of individual rewards. (3) It follows from (2) that there is no need in (α, β)-approximation since the optimal super-arm is easily selected according to the scores of the individual arms.

III. Preliminaries and Problem Definition

Consider an analyst investigating an incident as depicted in fig. 1. Sensor data (events, logs, and artifacts extracted from them) flows into the security information and event management (SIEM) (fig. 1.1). The various security analytic products operate using machine learning or expert rules to identify malicious behavior from artifacts stored in the SIEM. When malicious behavior is identified, the analyst receives an alert (fig. 1.2). Every alert is associated with a set of artifacts that caused it. After initial triage, when investigating a set of alerts the analyst is required to examine the set of artifacts associated with these alerts in order to form the complete picture of the attack (fig. 1.3). On one hand, we would like to collect the entire set of all available artifacts in order to form the complete picture. On the other hand, this is impossible due to constrained resources.

The latter is especially true when dealing with forensic data collection rather then real-time monitoring (e.g. hooking for file access or registry key usage). On many occasions the CTI about a malicious campaign arrives when the organization has already been penetrated. When it happens, the monitoring

policies are updated, but searching the entire IT infrastructure for suspicious artifacts (files, processes, registry keys, etc.) is too expensive and may disrupt operations. Such artifacts would be collected only as a result of a targeted hunt.

Next we discuss the assumptions regarding the attacker and the security tools in place.

A. Threat Model and Assumptions

1) Single Attack: It is a common assumption in threat hunting that an organization is under attack at any specific time. In order to perform efficient forensic investigation and correctly prioritize security alerts it is important to obtain as much information as possible about the currently investigated, yet unknown attack. In the scope of this work, we propose data collection policies focusing on one attack at a time. Nevertheless, the proposed policies are equally applicable for collecting data on multiple concurrent attacks as explained in § VI-D.

2) Forensic: The proposed policies are most applicable when searching for evidence of an attack post-factum or if the investigated adversarial activity is repetitive. "Racing" against the attacker is out of the scope of this work.

Definition 1 (Evidence): An artifact that was left by an attack on a host and can be found if specifically searched for.

Definition 2 (Search): The action of using a dedicated agent to attempt the collection of evidence.

3) CTI Is There: We assume that some intelligence about the investigated attack is found within the CTI knowledge base. For example, it may be executed by one of the well-known threat actors, but we do not know which. Or it may be any of the "we should have known that!" cases when relevant artifacts are within the CTI, but they are hard to pinpoint.

Definition 3 (Relevant Artifact): Artifact associated with the actual attack, promoting the forensic investigation of the attacked host when collected.

We assume two levels of the **novelty of an attack**:

(1) **A mild case**, when the investigated attack is known. For example, the attacker may use a variant of a malware that has been used against other victims during the same campaign. This case is not as trivial as searching for a malware in a database. The main challenge here is homing on the actual attack despite imperfect CTI and similarities between attacks (see fig. 13) in an efficient, resource hungry manner.

(2) **A tough case**, when the knowledge base does not contain information about the entire family of the malware being investigated. This may be the case of a new threat actor or a new campaign. The challenge here is homing on the attacks which are most similar to the investigated one in terms of artifacts stored in the knowledge base.

4) Erroneous Data Collection and Malware Variants: We assume that the attacked host is monitored and that the monitoring agent is capable of collecting all types of observable artifacts defined in the CTI. But we do not assume trusted monitoring. In either case, absence or presence of the actual attack in the knowledge base, we assume malware variants that may alter their behavior (see fig. 1) and any forensic evidence, e.g., a filename used for the dropper. Yet, we assume that such changes are limited and some artifacts are retained (see the adversarial analysis in § VI-E.2).

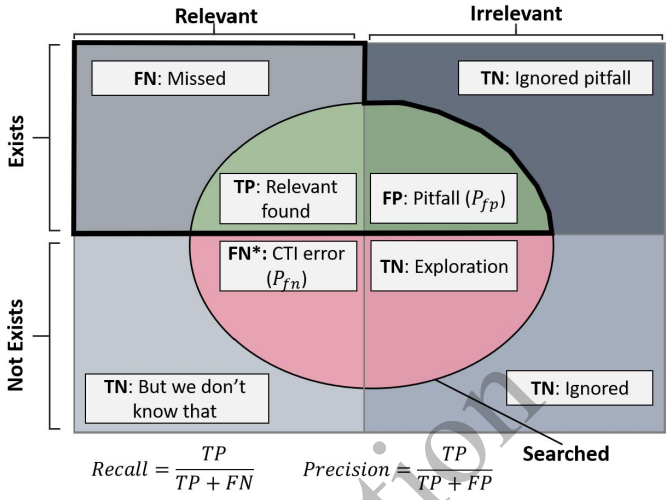


Fig. 2. Mapping between the 3 dimensions possible classes in our problem to the 2 dimensions possible classes. Each class is the intersection of the 3 dimensions. The classes marked with bold line are used for recall and precision calculation according to the defined formulas.

Definition 4 (Quality of CTI): An artifact associated with an attack within the knowledge base may not be found while attempting to collect it with the false negative probability P_{fn}. An artifact which is not related to the actual attack may be collected during the investigation with the false positive probability P_{fp}. The CTI may be considered informative when $1 - P_{fn} > P_{fp}$.

Performance of a classical information retrieval system is evaluated assuming that all items are retrievable and the search engine should only decide which are relevant (positive) and which are not (negative). In our case, an artifact may be relevant but irretrievable. Thus, we clarify what do *positive* and *negative* instances mean in our case to avoid confusion and define the targeted data collection objective. An artifact may be categorized along three dimensions: existence, relevance, and search attempt. We consider only artifacts that are both relevant to the actual unknown attack and exist on the host as *positive* (the left top quarter in fig. 2). Artifacts that are irrelevant to the actual attack or don't exist on the host are considered as *negative* and their search should be avoided.

The objective of the targeted data collection policies discussed in this paper is to maximize the number of collected artifacts associated with the actual attack using a limited number of search tasks.

B. Artifacts Collection as a Multi Shared-Arms Bandits Problem (MSAB)

In this section, we formalize the problem of efficient targeted data collection as a new variant of the MAB problem. Relying on CTI feeds or behavioral malware reports (fig. 1.4), we assume a finite set of attacks denoted by $A = \{a_1, \ldots, a_k\}$ and a finite set of artifacts D_j associated with each attack.

The set of all observable artifacts in the CTI knowledge-base is denoted by $O = \bigcup D_j$. An artifact $o \in O$ may be associated with multiple attacks. We will refer to such artifacts as *shared*. The set of attacks sharing an artifact o is denoted by S_o.

For simplicity, we assume that artifacts are collected iteratively, one at a time, by the SOAR agents (fig. 1.6). We also assume a fixed collection cost for each artifact, but the proposed algorithms can accommodate arbitrary data collection costs. The maximal number of collection tasks is bounded by the budget B. In each iteration $t \in T = \{1, \ldots, B\}$ of the collection process an artifact $o_t \in O$ is selected according to some policy $P : T \rightarrow O$. Here and in the rest of the text we assume that the decisions of all policies depend also on the history of all previous data collection attempts in the current investigation. We skip the policy arguments defining the investigation history for brevity. The main focus of the current paper is devising an efficient collection policy that maximizes the number of successfully identified artifacts related to the attack being investigated.

Each one of the attacks ($a_i \in A$) in the knowledge-base is modeled as a single-armed slot-machine ($a_1, a_2, \ldots, a_k$ in fig. 1). Attack selection policy denoted P_a selects an arm to pull in each iteration.

Pulling machine's arm (fig. 1.5) means collecting some observable $o \in D_j$ associated with the attack on the investigated host (artifact i in fig. 1.6). An observable selection policy P_o selects an observable artifact associated with the selected attack $P_a(t)$.

Definition 5 (Composite Policy): Let $P_a : T \rightarrow A$ and $P_o : A \rightarrow O$ be the attack and observable selection policies, respectively. The composition $P = P_o \circ P_a$ results in selecting an observable to collect in each iteration $t \in T$.

$P = P_o \circ P_a$ is reminiscent of the two-level bandits policy by Pandey et al. [28]. Unfortunately, since we assume forensic analysis, in our case every artifact may be searched only once. There is no point in searching again for the same artifact in the same environment. As a consequence, there is no arm pull history to learn from at the second level. Any observable artifact selected by P that makes it to the analysts dashboard at time t results in a reward (fig. 1.7).

Definition 6 (Reward): An artifact o_t is searched for at time t grants a reward $R(o_t)$.

$$R(o_t) = \begin{cases} 1 & o_t \text{ is collected} \\ 0 & \text{otherwise} \end{cases}$$

If the artifact o_t was searched for due to investigating the attack a_j ($o_t \in D_j$) then the reward from playing arm j is $X_{j,t} = R(o_t)$. Expected reward μ_j from playing an arm a_j is two-fold: either we search for an artifact *shared with the actual attack* or not. We call the latter a *mistake pull*. The reward from searching for an artifact shared with the actual attack $Y_{j,t} \sim Bernoulli(1 - P_{fn})$ follows the Bernoulli distribution with probability $1 - P_{fn}$. The reward from a mistake pull is $Z_{j,t} \sim Bernoulli(P_{fp})$. We assume that all $Y_{j,t}$ and $Z_{j,t}$ are independent. Also, $Y_{j,t}$ are identically distributed for all j, t and the same is true for $Z_{j,t}$. We denote the expected reward from mistake pulls as $\mu = E(Z_{j,t}) = P_{fp}$. Let γ_j denote the fraction of a_j's artifacts shared with the actual attack. The expected reward from pulling arm a_j is

$$\mu_j = E(X_{j,t}) = \gamma_j(1 - P_{fn}) + (1 - \gamma_j)P_{fp}. \quad (2)$$

TABLE I
SUMMARY OF NOTATIONS

MAB term	Meaning	Notation
Arms	Attacks	$A = \{a_j\}$
	Observable artifacts	$O = \{o_i\}$
	Artifacts related to a_j	$D_j \subseteq O$
	Attacks sharing an artifact	$S_o \subseteq A$
Optimal arm	The actual attack	a^*
Iteration	Ordinal number of search attempts	$t \in T = \{1, 2, \ldots, B\}$
	Number of search attempts	B
Arm pull at iteration t	An artifact search attempt	$P(t), t \in T$
Reward in iteration t from pulling arm j	Success of a search attempt (regardless the relevance of the found artifact).	$X_{j,t} \in \{0, 1\}$
Policy	Artifact selection algorithm	$P_a : T \rightarrow A$ $P_o : A \rightarrow O$ $P = P_o \circ P_a$

In MAB problems, the *optimal arm* (a^*) is defined as the arm with the maximal expected gain. In our case, it is the actual attack in the mild case ($\gamma_{a^*} = 1$) or an attack that is the most similar to the actual attack in terms of artifacts (a_1 in fig. 1) in the tough case ($\gamma_{a^*} < 1$). Let $Y_t^* = X_{a^*,t}$ denote the reward from playing the optimal arm at time t (called an *optimal arm-pull*). Following the MAB notation, we denote the expected reward from an optimal arm-pull by μ^*. In the mild case when $\gamma_{a^*} = 1$, $\mu^* = E(Y_t^*) = 1 - P_{fn}$.

Definition 7 (Multi Shared-Arms Bandits Problem (MSAB)): Given set of attacks (arms) A, their artifacts $D_j \subseteq O$, and a budget B; find a policy $P : T \rightarrow O$ such that the total reward is maximized after at most B arm pulls.

IV. MSAB POLICIES

After formalizing efficient artifacts collection as MSAB problem, in this section we utilize the additional information about shared arms to significantly improve standard MAB policies. We start with definitions needed to explain the *shared-arms* (SA) technique. In § IV-A a general policy template for the MSAB problem is provided (alg. 1), allowing to easily create new SA variants of existing policies. Next, we discuss the SA attack selection policies (P_a), artifact selection policies (P_o), and their compositions ($P_o \circ P_a$) in § IV-B. Finally in § IV-D we show that the SA variant of UCB1 is a special case of the Combinatorial UCB (CUCB) framework [8] and derive its' regret bound.

Consider an artifact $o_t = P(t)$ selected by some policy P at iteration t. Following the CMAB notation we refer to the set of attacks S_{o_t} sharing the artifact o_t as a super-arm. A super-arm

Algorithm 1 Multi Shared-Arms Bandits (MSAB) Template Algorithm

Input: $a_1, \ldots, a_k$ - attacks (bandit arms)
$D_1, \ldots, D_k$ - associated artifacts sets
O - observable artifacts set
B - investigation budget
P_a - attack selection policy
P_o - observable artifact selection policy
Output: F - found artifacts set

1 $F \leftarrow \{\};\ n \leftarrow 0;\ t \leftarrow 0$
2 $\forall_{o \in O}:\ S_o \leftarrow \{a_j : o \in D_j\}$
3 $\forall_{j \in [1,k]}:\ n_j \leftarrow 0,\ s_j \leftarrow 0,\ f_j \leftarrow 0$
4 **for** $t \in [1, B]$ **do**
5 **if** $t \leq k$ **then** // initial exploration
6 $a_j \leftarrow a_t$;
7 **else**
8 $a_j \leftarrow P_a(t)$ // select attack
9 $o_t \leftarrow P_o(a_j)$ // select artifact from D_j
 // search artifact and determine reward
10 $rwd \leftarrow SearchArtifact(o_t)$
11 **for** $a_i \in S_{o_t}$ **do** UpdateArm(i, rwd, o_t)
12 **if** $rwd > 0$ **then** $F \leftarrow F \cup \{o_t\}$;
13 **return** F
14 **Function** UpdateArm(j, rwd, o):
15 $n \leftarrow n + 1;\ n_j \leftarrow n_j + 1$
16 **if** $rwd = 1$ **then** $s_j \leftarrow s_j + 1$ **else** $f_j \leftarrow f_j + 1$
17 $D_j \leftarrow D_j \setminus \{o\}$

pull is thus the act of searching for o_t, paying the respective cost, and obtaining the respective reward:

Definition 8 (Super-Arm Pull): Let o_t be an artifact selected by policy P at iteration t. Super-arm pull $(R(o_t), o_t, t)$ is defined as searching o_t at iteration t, obtaining the reward $R(o_t)$, increasing the total number of pulls n and reducing the remaining budget B by 1.

Definition 9 (Arm Update): Assume a super-arm pull $(R(o_t), o_t, t)$ and a set of attacks $S_{o_t} = \{a_i : o_t \in D_i\}$ such that a_i share the artifact o_t. Arm update $(a_i, R(o_t), o_t, t)$ is defined as updating at iteration t a_i's average reward $(\overline{X}_{i,t})$ and increasing a_i's pulls counter $(n_{i,t})$ (without reducing the remaining budget B).

We say a collection policy (P) is a SA collection policy if on every iteration (t), it updates the properties of arms that share the observable artifact (o_t) selected by the policy. In this article we adapt existing MAB policies to be SA collection policies as described next.

A. The Shared-Arms Policy Template

The pseudo code shown in alg. 1 is a template of SA collection policy. Specific parts of the template (e.g. P_a, P_o, UdateArm) are implemented according to the chosen policies. Next, we will explain the pseudo code.

We start by initializing parameters and calculating S_o for each one of the artifacts $o \in O$ in the CTI knowledge-base (lines 1-3). Parameters maintained for each arm j include the number of related artifacts searched for (n_j), found (s_j), and not found (f_j). These parameters are sufficient to calculate arm parameters required by various MAB policies. For example, average reward is $\overline{X}_j = \frac{s_j}{n_j}$.

Next, we iteratively select artifacts to search until the budget B is exhausted (lines 4-12). The first iterations are devoted to pulling each attack once in order to obtain the attacks' initial reward distributions (lines 5-6). While many MAB algorithms, perform the initial exploration for the first k rounds, others, like Thompson Sampling, do not require such a phase. This initialization phase may be avoided by jumping from line 4 directly to line 8. Future work may utilize the distribution of artifacts across arms to optimize the initial exploration.

Following the initial exploration, attacks are selected according to the P_a policy (line 8). After selecting an attack a_j, we select an artifact $o_t \in D_j$ using the artifact selection policy P_o (line 9). o_t is searched for in the attacked host, and a reward, rwd, is obtained according to the search result (line 10). Next, we update the parameters of all arms sharing o_t (line 11). While updating the arms' parameters (lines 14-17), o_t is removed from the set of remaining artifacts because it makes no sense to repeatedly search for the same artifact. In case some D_j is empty after the removal of o_t, it's corresponding attack won't be available for selection on the next iterations. If the selected artifact was found, it is added to the set of collected artifacts (line 12).

Variety of MAB policies can be adapted to MSAB according to alg. 1. In § VI we demonstrate the adaptation of four policies. Such heuristic adaptation provides no regret guarantees, but empirical results show preferable performance of the SA variants over their non-adapted counterparts (§ VI fig. 4).

B. Most-Shared Artifact Selection Heuristic

Attack selection policies (P_a) need to be complemented with artifact selection policies (P_o) in order to provide a complete solution. While the trivial artifact selection policy is the random policy, we suggest a heuristic selecting the *most-shared* (MS) artifact, i.e., the artifact shared by highest number of attacks: $MS(a_j) = \text{argmax}_{o \in D_j} |S_o|$. By collecting the most-shared artifact and pulling all the shared-arms, we pull the maximal number of arms at each iteration. Each arm-pull means feeding the policies with more information. Consequently pulling all shared-arms maximizes the amount of extracted information in every collection task.

C. SA Variants of Common MAB Policies

All SA variants of regular MAB policies utilize the additional information gained from updating all arms in S_{o_t}. Successful search results provide additional evidence for multiple attacks. Failed search results are also useful because they decrease the likelihood of selecting all attacks sharing absent artifacts even if these attacks were not explicitly selected by the MAB policies themselves.

Note that MAB is a private case of MSAB where no artifacts are shared by different attacks, i.e., $\forall o \in O : |S_o| = 1$. Thus, a single arm update is performed corresponding to each selected observable. In addition, as with the standard

MAB, in such a case the rewards of the different arms are independent.

For a standard MAB policy with independent arms the expected average reward of arm j is identical to its expected reward: $\mathbb{E}(\overline{X}_j) = \mathbb{E}\left(\frac{s_j}{n_j}\right) = \gamma_j\mu^* + (1-\gamma_j)\mu = \mathbb{E}(X_j)$. Since all MAB policies strive to increase the gained reward they select arms with larger fraction (γ_j) of artifacts shared with the actual attack. Consequently, due to updates of the shared arms (def. 9, alg. 1 line 11) those arms that were not explicitly selected by the policies in line 8 may record superfluous rewards. Eventually, the arm parameters maintained by alg. 1 may reflect more optimistic beliefs than the expected reward of sub-optimal arms.

$$\mathbb{E}\left(\frac{s_j}{n_j}\right) \geq \gamma_j\mu^* + (1-\gamma_j)\mu \tag{3}$$

Such optimism may increase the amount of unnecessary exploration but it pays-off as we will see in § VI-D fig. 4. Next we analyze the behavior of the SA variant of UCB1 attack selection policy and derive its regret bound using the CMAB framework.

D. Shared Arms as Combinatorial MAB

In this section we show that the SA variant of UCB1 is reminiscent to the Combinatorial UCB policy introduced by Chen et al. [8] with a slightly different constant. We use their framework in order to infer a regret bound for MSAB. For simplicity, the following discussion refers only to the mild case. Nevertheless, the tough case can be simulated in the mild settings by increasing the probability P_{fn}.

Assume an MSAB problem with k arms, and a single optimal arm with the expected reward $\mu^* = 1 - P_{fn}$. The reward μ^* is expected when searching for an artifact shared with the actual attack. Otherwise, the expected reward of a mistake pull is $\mu = P_{fp}$. Assume a composite policy **UCB1-Rnd** where $P_a = UCB1$ and $P_o(a_i)$ randomly selects an artifact from D_i. Let $o_t = P_o(P_a(t))$ be the artifact selected at iteration t. Following defs. 8 and 9 the rewards used to update all arms in S_{o_t} are always equal because they stem from the presence or absence of the same artifact in the investigated environment.

In terms of CMAB the set S_{o_t} of attacks sharing the artifact o_t is a super-arm. Assume two artifacts o_{t_1}, o_{t_2}. If $S_{o_{t_1}} = S_{o_{t_2}}$ then according to CMAB it is the same super-arm. Every artifact may be searched for only once. Thus, a super-arm S may be pulled at most $|\{o \in D : S_o = S\}|$ times. This is an important difference between MSAB and CMAB applications presented by Chen et al. [8]. Luckily, their framework does not require unlimited lifetime of the super-arms.

The total aggregated reward from pulling a super-arm within the CMAB framework $R(S_{o_t})$ is equal to the reward of searching an artifact o_t according to def. 6 ($R(S_{o_t}) \equiv R(o_t)$). Thus to derive $R(S_{o_t})$ from a set of (the identical) rewards of the individual arms we can assume any aggregation function that selects one value from the set (e.g., min or max). Assume $R(S_{o_t}) = \text{any}\{X_{i,t} : a_i \in S_{o_t}\}$ where $\forall_{i,j} X_{i,t} = X_{j,t}$ and 'any' chooses an arbitrary element.

Consider the vector of expected rewards $(\mu_1, \ldots, \mu_k)$ of all individual arms (eq. (2)). Assume a set of super-arms $\mathcal{S}$ corresponding to the set of remaining artifacts that were not searched for yet. Unlike in most CMAB applications, the problem of choosing a super arm $S \in \mathcal{S}$ that maximizes $\mathbb{E}(R(S))$ is trivial. We need to choose any artifact from the arm with the maximal expected reward.

$$\begin{aligned}\underset{S}{\text{argmax}}\{\mathbb{E}(R(S))\} &= \underset{S}{\text{argmax}}\left\{\max\{\mu_j : a_j \in S\}\right\} = \\ &= S_o \text{ where } o = P_o(\underset{a_j}{\text{argmax}}\{\mu_j\})\end{aligned}$$

Thus $r_\mu(S) = \mathbb{E}(R(S)) = \max_{a_j \in S}\{\mu_j\}$. In particular, the $(\alpha = 1, \beta = 1)$-approximation oracle in our case is $P_o(\text{argmax}_{a_j}\{\mu_j\})$.

CUCB applies the approximation oracle on the vector $(\hat{\mu}_{1,t}, \ldots, \hat{\mu}_{k,t})$ at each iteration t where $\hat{\mu}_{j,t}$ is the CUCB score according to eq. (1). Thus, if $P_a(t) = \text{argmax}_{a_j}\{\hat{\mu}_{j,t}\}$ alg. 1 behaves exactly as CUCB updating all arms associated with the super-arm S_{o_t} in every iteration. Note that when $P_a(t) = UCB1(t) = \text{argmax}_{a_j}\left\{\overline{X}_{j,t} + \sqrt{\frac{2\ln t}{n_{j,t}}}\right\}$ the only difference between alg. 1 and CUCB is the constant 2 instead of 3/2 within the square root.

Monotonicity requirement is satisfied because $\forall_j : \mu_j \leq \mu'_j \implies r_\mu(S) = \max_{a_j \in S}\{\mu_j\} \leq \max_{a_j \in S}\{\mu'_j\} = r_{\mu'}(S)$ Finally, the bounded smoothness function is $f(x) = x$.

Lemma 4.1 (MSAB Bounded Smoothness): $\forall S \subseteq A, \{\mu_j\}, \{\mu'_j\} : max_{a_j \in S}|\mu_j - \mu'_j| \leq \Lambda \Rightarrow |r_\mu(S) - r_{\mu'}(S)| \leq f(\Lambda)$

Proof: Let $S \subseteq A$ be an arbitrary super-arm. Let $a_j \in S$ and $\Lambda, \mu_j, \mu'_j \in [0,1]$ such that $\max_{a_j \in S}|\mu_j - \mu'_j| \leq \Lambda$. Let $a_i = \text{argmax}_{a_j}\{\mu_j\}$ and $a_{i'} = \text{argmax}_{a_j}\{\mu'_j\}$. Then $r_\mu(S) = \mu_i$ and $r_{\mu'}(S) = \mu'_{i'}$. Without loss of generality assume $\mu'_{i'} \geq \mu_i$. Then $|r_\mu(S) - r_{\mu'}(S)| = \mu'_{i'} - \mu_i = \mu'_{i'} - \max_{a_j \in S}\{\mu_j\} \leq \mu'_{i'} - \mu_{i'} \leq \max_{a_j \in S}|\mu'_j - \mu_j| \leq \Lambda = f(\Lambda)$ □

Let $\mathcal{S}_B$ denote a set of bad super-arms with $r_\mu(S) < \alpha\mu^*$ for each $S \in \mathcal{S}_B$. In our case, since $\alpha = 1$, $\mathcal{S}_B$ corresponds to the collection of artifacts not shared by the actual attack $\mathcal{S}_B = \{S_o : o \in O \setminus D^*\}$. Regret bounds of CUCB depend on the following two quantities adapted from [8]:

$$\Delta_{min} = \min_{a_j \in A}\left\{\mu^* - \max\left\{r_\mu(S) : a_j \in S \wedge S \in \mathcal{S}_B\right\}\right\}$$

$$\Delta_{max} = \max_{a_j \in A}\left\{\mu^* - \min\left\{r_\mu(S) : a_j \in S \wedge S \in \mathcal{S}_B\right\}\right\}$$

Intuitively Δ_{min} is the regret of the best arm (attack) that includes artifacts not related to the actual attack. In MSAB terms it is $\Delta_{min} = \mu^* - \max_{a_j \in A, \gamma_j < 1}\{\mu_i\}$. Similarly, Δ_{max} is the regret of the worst arm (attack) $\Delta_{max} = \mu^* - \min_{a_j \in A}\{\mu_i\}$. Typically Δ_{max} would be $\Delta = \mu^* - \mu$.

The simplified form of the instance-dependent regret bound of CUCB for the targeted data collection application (adapted from [8] Equation 4) is:

$$\left(\frac{6\ln B}{\Delta^2_{min}} + \frac{\pi^2}{3} + 1\right) \cdot k \cdot \Delta \tag{4}$$

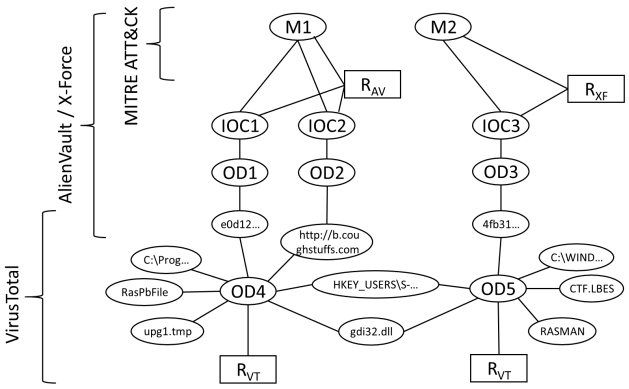


Fig. 3. An illustration of AttackDB's structure.

V. AttackDB

Here the knowledge base (AttackDB) fusing data from multiple CTI sources used in our experiments. We constructed a rich AttackDB that consists of CTI from the MITRE ATT&CK Enterprise knowledge base,[4] the AlienVault Open Threat Exchange,[5] the IBM X-Force Exchange,[6] and VirusTotal.[7] The details on the structure and the population of AttackDB can be found in Supplemental Materials II.A and II.B.

Following the CTI fusion we built a data set associating a wide variety of telemetry including file names (opened, created, searched, etc.), URLs, domains, IPs, process names, registry keys, mutexes, emails, and more with 253 attack families as depicted in fig. 3. The telemtry was extracted from 53,005 VirusTotal behavioral reports and contains about half a million observable artifacts in addition to 144,216 IOCs. More than 90% of the attacks are associated with less than 10K artifacts. Additional statistics on the data set can be found in Supplemental Materials II.C.

VI. Experiments

In this section we present the experiments conducted to evaluate the collection policies using AttackDB. We aim at answering the following research questions:

RQ 1 (SA or Not SA): To what extent do the SA policy variants outperform their non-SA counterparts? (fig. 4)

RQ 2 (MS or Rnd): Which artifact selection policy is better MS or random selection? (fig. 5)

RQ 3 (Amount of CTI): Which policies benefit the most from the amount of CTI on the actual attack? (figs. 4 to 6)

RQ 4 (Efficiency Through Time): Which data collection policy collects the largest number of relevant artifacts in a given time budget? (figs. 8 7 and 10)

RQ 5 (Attack Detection): How fast do the policies home on the actual attack? (fig. 8)

RQ 6 (Robustness to Noise): How robust is targeted data collection to attack variations? (figs. 9 and 10)

RQ 7 (Adversarial Analysis): How can the attacker delay the investigation? (fig. 10)

A. Composite Policies and the Oracle

According to def. 5, we evaluate MSAB policies composed from an attack selection policy and an observable selection policy. There are ten attack-selection policies (P_a): **Rnd** – a baseline that randomly selects an attack. **UCB1** – the classic UCB1 policy. **EG01** – the classic ϵ-Greedy policy. We use $\epsilon = 0.1$, which outperformed other ϵ values in preliminary experiments. **TS** – Thompson sampling policy [9]. **KL-UCB** – the KL-UCB policy [10]. **SAUCB**, **SAEG01**, **SATS**, **SAKLUCB** – the SA variants of UCB1, EG01, TS, and KL-UCB policies respectively. **CUCB** – the Combinatorial UCB policy [8] applied for MSAB. The following two observable selection policies (P_o) can be combined with each attack selection policy: **Rnd**(a_j) – randomly selects an artifact associated with a_j. **MS**(a_j) – selects the most-shared artifact $o \in D_j$.

In total there are twenty composite collection policies ($P_o \circ P_a$). We name the composite policies according to their two constituents. For example, **EG01-Rnd** selects an attack according to the ϵ-greedy policy, and then selects a random observable. **SAUCB-MS** selects an attack according to the SA variant of UCB1 policy. Then it selects the most-shared artifact associated with the attack.

To highlight the space for potential improvement we execute an **Oracle** that always selects the optimal attack.

B. Experimental Settings

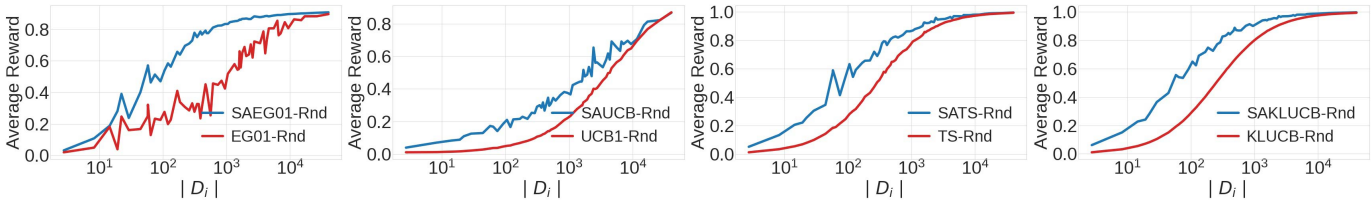
The independent variables in our experiments are the policies described earlier, the actual attack being investigated, quality of the CTI (P_{fp}, P_{fn}) to challenge the robustness of the policies, novelty of an attack, and the investigation budget.

1) Actual Attack: We run the experiment with each one of the 253 attacks in the AttackDB. Attacks differ by the number of artifacts associated with them, and the artifacts shared with other attacks. We mimic an attacked host by populating its logs with a subset of all artifacts in the AttackDB. Some of them are related to the actual attack and some are not.

2) Quality of the CTI: We assume errors in the CTI due to variety of reasons (see § III-A). The false positive (P_{fp}) and false negative (P_{fn}) error probabilities encapsulate both the errors due to imperfect data collection and errors due to malware variants. To test the robustness of the collection policies in extreme cases we experiment with $P_{fp} \in \{0, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64\}$ and $P_{fn} \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. We will refer to $P_{fp} = P_{fn} = 0$ as high CTI quality and to $P_{fp} = 0.01$, $P_{fn} = 0.1$ as medium CTI quality.

3) Novelty of an Attack: We experiment with the actual attack *present* in or *absent* in the AttackDB. The former (a mild case) mimics CTI containing information about the actual attack. The latter (a tough case) mimics a situation when AttackDB does not contain information about the whole malware family. Yet, some relevant artifacts that are shared with multiple malware families in the AttackDB help guiding the investigation.

[4]https://github.com/swimlane/pyattck
[5]https://otx.alienvault.com/dashboard/new
[6]https://exchange.xforce.ibmcloud.com/
[7]https://www.virustotal.com/gui/home/search



(a) EG01 (b) UCB1 (c) TS (d) KL-UCB

Fig. 4. The average reward as a function of the number artifacts associated with the actual attack for SA and non-SA variants of MAB policies (RQs 1 to 3) ($\gamma^* = 1$, $P_{fp} = P_{fn} = 0$, $P_o = Rnd$).

4) Budget: In our experiments we use both constant and non-constant budgets.

C. Evaluation Metrics

To evaluate the effect of the collection policy on the artifacts collection process we use the following metrics:

1) Average Reward: MSAB policies maximize the total reward. The average reward ($\overline{X}_B$) of a policy after B iterations across all arms is the total number of highlighted artifacts until iteration B, divided by B: $\overline{X}_B = \left(\sum_{j=1}^{k}\sum_{t=1}^{B} X_{j,t}\right)/B$. High reward indicates that the policy makes a good use of the clues (found and not found artifacts) during the search.

2) Recall and Precision: We measure the ability of a policy to pinpoint existing relevant artifacts using precision and recall (see fig. 2). While the general intuition behind precision and recall is retained, we note here, that quality of the CTI, in particular P_{fp}, directly affects precision of the collected data. The policies have no means to differentiate between TP and FP. Thus it is important to evaluate the extent to which FP mislead the search process by measuring precision.

Different attacks have different numbers of relevant artifacts, D_j, associated with them. Thus, comparing or averaging precision across attacks at specific t, denoted as P@t,[8] is unfair toward attacks with less than t artifacts. To remedy the heterogeneity of $|D_j|$ we will use mean average precision.

3) Mean Average Precision (MAP): MAP is a metric commonly used to evaluate the performance of an entire information retrieval process [30], [31]. MAP is derived from P@t. Provided an investigation of the attack a_j, average precision, $AP(a_j)$, is defined as the P@t averaged over those arm pulls at which the searched artifact was relevant and successfully retrieved: $AP(a_j) = \left(\sum_{t=1:P(t)\in D_j}^{B} P@t \cdot X_{jt}\right)/(TP+FN)$, where $TP + FN$ is the total number of existing relevant artifacts (as in fig. 2). Mean AP is $MAP = \left(\sum_{a_j \in A} AP(a_j)\right)/k$, where k is the number of attacks.

4) Normalized Rank: The average reward of an arm $\overline{X}_j$ reflects the confidence of the policy that arm j is the optimal one (i.e. the actual attack). We order all attacks on a scale [0, 1] where 0 corresponds to the lowest average reward and 1 corresponds to the highest and refer to the position on an attack on this scale as its normalized rank.

[8]Commonly referred to as Precision at k (P@k) in literature.

D. Results

1) Mild Case With Perfect CTI: We execute the first experiments in mild settings ($\gamma^* = 1$) with high quality CTI ($P_{fp} = P_{fn} = 0$). The investigation ends when 100% of the relevant existing artifacts are found. Since all successfully collected artifacts are relevant (precision = 1), the main challenge is identifying the actual attack fast, despite shared artifacts. We elaborate on identifying the actual attack in presence of noise in § VI-E.

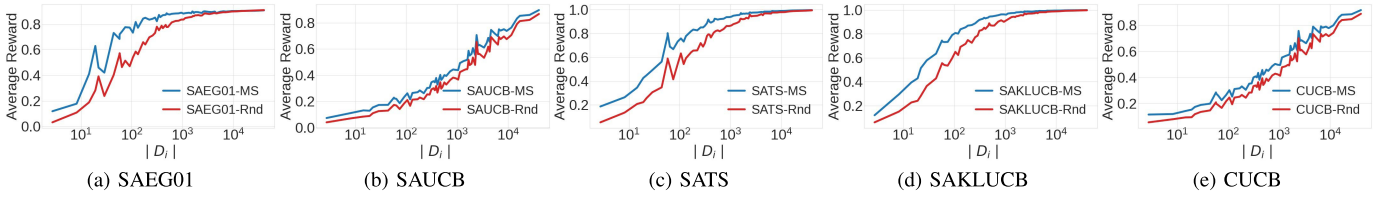
a) SA or not SA: The first and most important question (RQ 1) is whether or not SA variants of standard MAB policies implemented according to alg. 1 outperform their non-SA counterparts in our problem settings. We compare the SA and non-SA variants of attack selection policies

Figure 4 shows that SA variants of EG01, UCB1, TS, and KL-UCB outperform their non-SA counterparts. CUCB is omitted because it is inherently a SA policy (Figure IV-D). In the rest of this section we present only the SA variants except when other results exhibit a trade-off between the variants.

b) MS or Rnd: Figure 5 focuses on the artifact selection policies comparing the selection of the most shared artifact with random selection (RQ 2). The results show consistent benefit of MS over Rnd. Thus, in the rest of this section, we present only the MS results except for the random baseline.

c) Amount of relevant CTI: We conclude the first part of the results with fig. 6 that summarizes the best performing combinations of policies (SA*-MS). Figures 4 to 6 show that the amount of relevant CTI significantly affects the performance of automatic targeted data collection (RQ 3). The more CTI there is the higher is the average reward. It means that the targeted data collection becomes more efficient with additional CTI up to the level where SAKLUCB and SATS approach the efficiency of the oracle. This behavior is explained by the ability of the policies to learn the distributions of rewards. Although, these distributions are biased as explained in § IV-C, the policies effectively home on the actual attack as shown in fig. 8a With attacks having few (<100) known artifacts the policies do not have enough time for sufficient exploration before they exhaust the relevant CTI.

SAKLUCB-MS and SATS-MS are leading the way regardless the number of artifacts associated with the attack. The performance of the SAEG01-MS and EG01-MS policies is unstable. We display both in fig. 6 because neither one of them is strictly better than the other. SAEG01-MS performs



(a) SAEG01 (b) SAUCB (c) SATS (d) SAKLUCB (e) CUCB

Fig. 5. The average reward as a function of the number artifacts associated with the actual attack for Rnd vs. MS artifact selection policies (RQs 2 to 3) ($\gamma^* = 1$, $P_{fp} = P_{fn} = 0$).

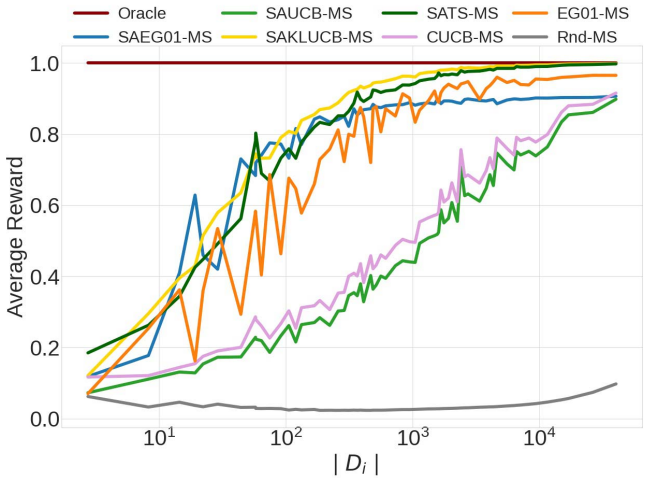


Fig. 6. The average reward as a function of the number artifacts associated with the actual attack (RQ 3) ($\gamma^* = 1$, $P_{fp} = P_{fn} = 0$).

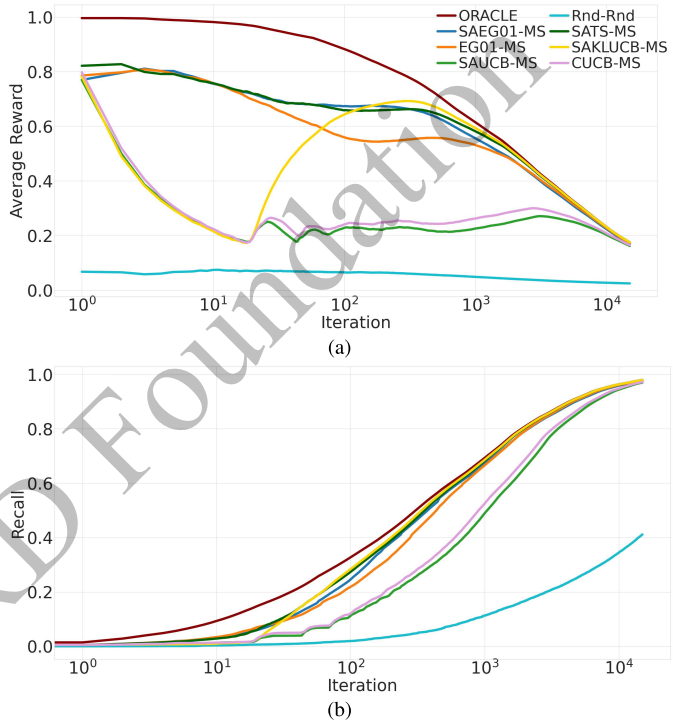


Fig. 7. Average reward and recall during the investigation (RQ 4) ($\gamma^* = 1$, $P_{fp} = P_{fn} = 0$).

better for attacks with up to 1K artifacts and EG01-MS leads afterwards. Despite theoretical guarantees CUCB-MS and SAUCB-MS have the worst performance.

Detailed results on performance of attacks with varying number of related artifacts CTI and varying fraction of artifacts shared with other attacks can be found in the Supplemental Materials in figs. 23 to 25.

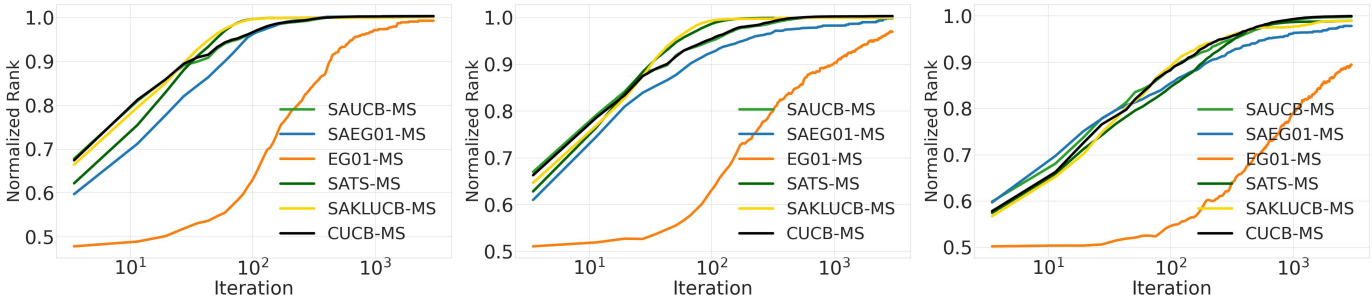
d) Efficiency through time: This experiment was executed with a fixed budget for all attacks. Figure 7 depicts the efficiency of the policies in terms of recall and average reward. There is no policy that consistently outperforms the others through time. EG policies and SATS-MS lead for the first 100 iterations (approx.). SAKLUCB-MS performs similarly to SAUCB-MS and CUCB-MS for the first few iterations breaking out and leading after a few hundreds of iterations until the end. The recall does not reach 1 because even for the oracle because some attacks have more artifacts than the budget. Additional results with the actual attack absent in AttackDB (tough case) can be found in the Supplemental Materials in fig. 17. The average reward decreases even for the oracle. This is explained by the skewed distribution of the amount of CTI associated with the different attacks (see fig. 12). In most cases the policies quickly find all relevant artifacts and continue the investigation with no reward.[9]

[9]In general, the data collection can be stopped when the policy has attempted searching all artifacts of the attack with the highest rank.

E. Mild Case Imperfect CTI

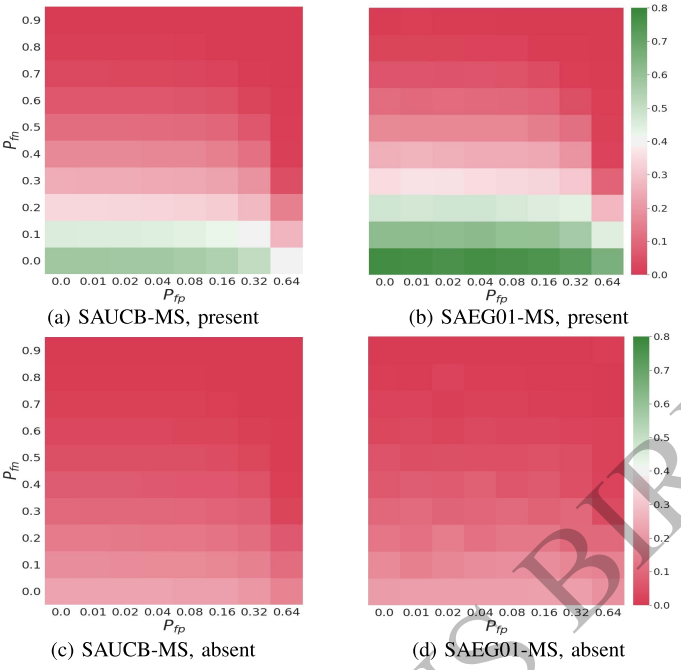
The ability to home fast and accurately on correct leads is crucial in resource constrained investigation (RQ 5). In fig. 8 policies are compared based on the reciprocal rank of the actual attack during the investigation with different levels of the CTI quality. SAKLUCB-MS and SATS-MS are the first to pinpoint the actual attack with high and medium quality CTI. Nevertheless, CUCB-MS and SAUCB-MS eventually outperform SAKLUCB-MS and SATS-MS by a small margin. EG policies are the worst with EG01-MS falling far behind because they fail to efficiently explore the reward distributions.

With medium and low CTI quality, all ϵ-greedy policies failed to rank the actual attack as the highest even after 3000 iterations. All UCB-based policies are relatively robust to the quality of the CTI. But the benefit of SAKLUCB-MS and SATS-MS over CUCB-MS diminishes as the quality of the CTI drops. (see fig. 8c) Approximately 1000 iterations are sufficient to pinpoint the actual attack.



(a) $P_{fp} = 0$, $P_{fn} = 0$ (b) $P_{fp} = 0.01$, $P_{fn} = 0.1$ (c) $P_{fp} = 0.05$, $P_{fn} = 0.5$

Fig. 8. The normalized attack rank during the search process with various levels of CTI quality. (RQs 4 and 5) ($\gamma^* = 1$).



(a) SAUCB-MS, present (b) SAEG01-MS, present

(c) SAUCB-MS, absent (d) SAEG01-MS, absent

Fig. 9. MAP of SAUCB-MS and SAE01-MS as a function of P_{fp} and P_{fn}. The actual attack is present (top, $\gamma^* = 1$) or absent (bottom, $\gamma^* < 1$) in the AttackDB (RQ 6).

1) Varying Attack Novelty and CTI Quality: Next we focus on the sensitivity of the targeted data collection to the quality of CTI (RQ 6) for both mild and tough cases with different levels CTI quality. We report MAP in fig. 9 to answer this research question because MAP captures the performance of the entire search process.

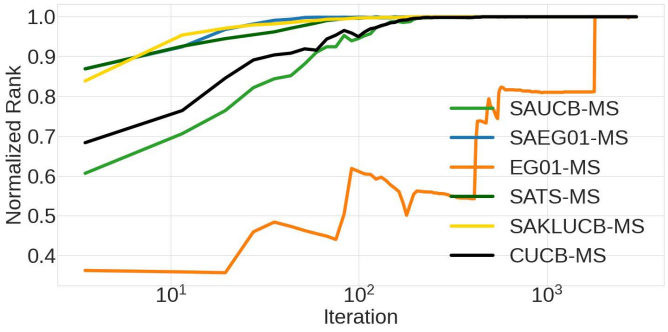
Surprisingly, the collection policies are very robust to P_{fp} especially when the actual attack is not novel. It means that superfluous information collected during the investigation process does not mislead the MAB policies. In fact, UCB-based policies perform significantly better than random as long as $P_{fp} < 1 - P_{fn}$. Nevertheless, at high levels of false positives ($P_{fp} > 0.3$) SAEG01-MS outperforms SAUCB-MS suggesting the choice of a policy provided insights about the quality of the CTI. The collection policies are less robust to P_{fn} then they are to P_{fp}. P_{fn} mimics the attacker's effort to alter the observable artifacts produced by his malware.

2) Adversarial Analysis: We conclude the experimental section with a discussion and results highlighting the measures attacker may take to impair the investigation process. MAB algorithms prefer to exploit arms that were successful in the past. To evade the discovery by MAB collection policies, the actual attack should not stand out from the policy's point of view. That is, the probability of the actual attack being identified as the actual attack should be similar to the probability of other attacks in the knowledge base. Perturbing or eliminating any evidence left by the attack is the trivial yet very expensive solution from the attacker's perspective. It is also covered by previous experiments where $P_{fn} \to 1$.

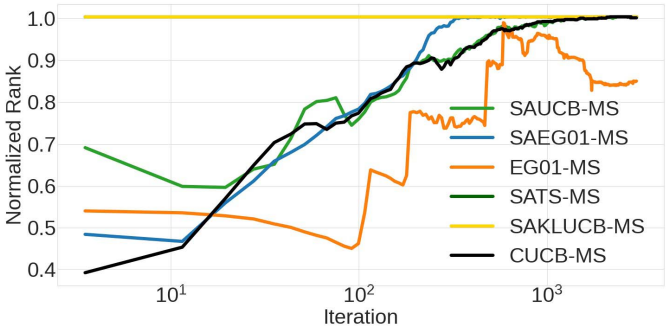
Here we investigate a different approach of *stalling* the policy during the search process (RQ 7). By creating an attack that shares artifacts with many decoy attacks in the knowledge base, an attacker may convince a policy to exploit the sharing attacks instead of the actual attack. EG-based policies are likely to focus on a wrong attack and stick with it. UCB-based policies may fail focusing on any particular attack probing each decoy with the same frequency as the actual attack. This effect is magnified with the lower Gini coefficient of the shared artifacts. Being forced to explore multiple attacks in parallel, UCB requires putting additional effort to detect the correct investigation lead. When the investigation budget runs out, we would have few artifacts associated with many different attacks instead of many artifacts associated with a single attack.

We created an artificial evading attack (EA) by taking 30% of the artifacts associated with 55 different attacks in the knowledge base, yielding a new attack associated with 5000 artifacts. The 55 attacks used to construct the EA were selected randomly from all attacks associated with less then 1000 artifacts. The number 5000 was chosen because in preliminary experiments, the stalling was effective for attacks associated with at least 4000 artifacts.

Similar to RQ 5 the EA was set as the actual attack, and its rank was reported. We compare the investigation of the EA to the investigation of a typical non-evading attack (non-EA) associated with similar number of artifacts in fig. 10. EA is effective in delaying UCB-based policies. While UCB1-MS policy pinpointed the non-EA after 750 iterations (fig. 10a), it failed to detect the EA even after 3000 iterations (fig. 10b). SAUCB-MS eventually detects the EA 2.5 times later than it detects the non-EA (iteration 1250 vs. less than



(a) Non-evading attack associated with medium number of artifacts.



(b) Evading attack associated with medium number of artifacts.

Fig. 10. Evading vs. non evading attack with meium number of artifacts and medium CTI quality (RQs 6, 4 and 7).

500 respectively). The SAUCB proves its advantage over UCB1, utilizing the shared artifacts for efficient data collection. The EA confused the SAEG01-MS only a little bit in most trials delaying the detection of the attack by at most 50%. Additional results with different CTI quality levels and for EA associated with larger numbers of artifacts can be found in the Supplemental Materials figs. 18 to 22.

Adversarial MAB, e.g. EXP3 [32], can be adapted to mitigate the evasion heuristic discussed here. To support our problem settings, EXP3 should assume predetermined set of artifacts rather than predetermines sequence of rewards from each arm.

VII. Related Work

Various problems were modeled and solved successfully using MAB algorithms, such as finding the optimal recommendation in recommender systems [33], [34], trials automation [25], and communication networks [35]. Wang et al. [34] are modeling items for recommendation as machines' arms; suggesting an item is modeled as pulling an arm and user clicks are the reward. In trials automation, MAB was utilized to select which trials to conduct. Similar to this work, Liu et al. [25], are suggesting a variant of UCB1, which adaptively keeps a user-specified ratio between the exploration and exploitation components. The ratio mentioned above is needed to work in high-stakes domains (i.e., domains that cannot allow vast exploring freedom, e.g., education).

MAB was also applied successfully to security-related problems [7], [12], [36], [37]. Ferdowsi et al. [12], suggest MAB as one of two approaches for the detection of cyber attacks on autonomous connected vehicle (ACV) sensors. By modeling the possible subsets of the vehicle's sensors as a single armed slot machines, Ferdowsi et al. were able to apply UCB algorithms in order to find the optimal subset sensors (i.e., the most secure sensors). Based on the optimal subset, the vehicle can function using sensors with lower error and a possibly lower amount of injected data (i.e., data injected by attackers who seek to damage the ACV). Yue et al. [7], utilized MAB for seed selection in fuzz testing, modeling the seeds as slot-machines. Also, they used the reward probability of the slot-machines to assign energy to the seeds.

Similarly to this work, Grushka et al. [37], modeled the problem of efficient data sampling as a special case of the MAB problem. Due to resource constraints, Grushka et al. select at each iteration k users to monitor and log their activity. The selection of users is done by k consequent arm-pulls at each iteration of the policy.

While in our case we are performing multiple arm pulls at each iteration, they all collect the same artifact. Since we pay budget for collecting an artifact, collecting k different artifacts at a single iteration will not improve the budget consumption of the policy. Also, the best policy in terms of reward presented in [37] is the classic ε-greedy policy ($\varepsilon = 0.2$) in a k-pulls variant. While ε-greedy is performing well empirically, it is not regret bounded, and as we see in the experiments results, it is less stable comparing to UCB1 (especially when the CTI quality is impaired).

A. Dependent MAB

One of the first researches to formalize and study MAB with dependent arms (henceforth, *dependent bandits*) was presented by Pandey et al. [28]. They construct policies that perform better then those for independent bandits by exploiting the similarity between arms. Dependent arms are grouped into clusters. A general two-level bandit policy (TLP) was suggested and instantiated. TLP uses as a subroutine any policy for independent bandits, using it first to choose a cluster and then reused in the selected cluster to choose an arm to pull. TLP may be more efficient to solve due to the lower number of bandits (clusters can be seen as bandits to) to work with [28]. Thus, the method's efficiency is affected by the input data, and a big cluster size will impair the efficiency of the method.

B. Budgeted MAB

In the budgeted MAB problem, the player needs to pay a cost after pulling an arm while having a limited resource set. Xia et al. [38] extend two known MAB policies: UCB1 and ε-greedy by embedding the cost term in their formulations. Next, they evaluate and prove sublinear regret bounds of the extended policies with respect to the budget. One of the resulting budgeted policies is i-UCB, which uses an average reward-to-cost ratio instead of only average reward in UCB1. In our research, we assume a constant cost of one for every arm pull, making i-UCB and the other modified policies behave like the corresponding original policy. For instance, when the cost is one, i-UCB behaves as UCB1.

Besides the literature mentioned above, there also exist some works studying MAB problems with multiple budget constraints. For instance, Badanidiyuru et al. [39], studies the *bandits with knapsacks* setting in which the total number of plays is constrained by a predefined number in addition to the total-cost budget constraint. Due to our cost policy, here, the budget constraint is acting as both constraints: total plays cost and the total number of plays together.

C. Reinforcement Learning

Our work is also related to reinforcement learning, which tries to solve sequential decision problems as well. In [36] the authors consider the use of adaptive reinforcement learning to prevent damage caused by malicious attacks on IT systems. The defender iteratively reinforces its defense by deciding where and what defense to deploy to set optimal defenses against the attacker.

VIII. Conclusion

In this paper, we strive to increase the level of automation during hunting by collecting artifacts most relevant to the currently investigated yet unknown attack. We formulated the problem of targeted data collection as a variant of the MAB problem, introducing the Multi Shared-Arms Bandit (MSAB) problem in § III-B. By utilizing the interrelations of attacks, MSAB increases the effectiveness of the data collection during threat hunting, a critical aspect in resource constrained systems. We investigate shared-arms (SA) variants of four MAB policies, UCB1, ε-Greedy, Thompson sampling, and KL-UCB and evaluate them on a real world dataset of 53K behavioral reports. We provide theoretically proven bounds on the regret of the SA variant of UCB1 (SAUCB) in § IV-D.

Results are showing the superiority of the SA variants over other policies, detecting the relevant attack quicker by feeding the policy with the added information from the shared-arms pulls. The proposed policies are robust to perturbations in the collected information captured by the quality of the CTI but are significantly affected by the number of artifacts associated with the investigated attack and the percent of shared artifacts. Adversarial analysis of the collection policies reveals that UCB-based policies are exposed to stalling, which is partially mitigated using the SAEG01 policy.

Future Work: (1) Measure the reward of artifacts based on their actual contribution to the inference of attack techniques, attack attribution, and attack detection tasks. (2) Extend the MSAB theory to accommodate arbitrary cost and arbitrary reward. (3) Employ reinforcement learning for artifacts selection policies once an attack is chosen. (4) Use machine learning to select promising investigation leads at the very beginning of each investigation instead of probing all attacks first. (5) Future extensions may follow [37] to employ schemes where multiple artifacts are retrieved in parallel. (6) Future adaptation of CUCB to MSAB may definitely improve the performance of SA variants of policies suggested in this paper.

References

[1] M. J. Haber and B. Hibbert, *Threat Hunting*. Berkeley, CA, USA: Apress, 2018, pp. 75–78, doi: 10.1007/978-1-4842-3048-0_7.

[2] I. Tyler Technologies. (2018). *A Guide to Cyber Threat Hunting*. [Online]. Available: https://www.tylertech.com/services/ndiscovery/nDiscovery-Threat-Hunting.pdf

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

[4] H. Rasheed, A. Hadi, and M. Khader, "Threat hunting using GRR rapid response," in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 155–160.

[5] R. Brewer, "Could SOAR save skills-short SOCs?" *Comput. Fraud Secur.*, vol. 2019, no. 10, pp. 8–11, Jan. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S136137231930106X

[6] R. Weber, "On the Gittins index for multiarmed bandits," *Ann. Appl. Prob.*, vol. 2, no. 4, pp. 1024–1033, 1992.

[7] T. Yue et al., "Ecofuzz: Adaptive energy-saving greybox fuzzing as a variant of the adversarial multi-armed bandit," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 2307–2324.

[8] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 151–159.

[9] S. Agrawal and N. Goyal, "Analysis of Thompson sampling for the multi-armed bandit problem," in *Proc. Conf. Learn. Theory*, 2012, pp. 1–39.

[10] A. Garivier and O. Cappe, "The KL-UCB algorithm for bounded stochastic bandits and beyond," in *Proc. 24th Annu. Conf. Learn. Theory*, 2011, pp. 359–376.

[11] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[12] A. Ferdowsi, S. Ali, W. Saad, and N. B. Mandayam, "Cyber-physical security and safety of autonomous connected vehicles: Optimal control meets multi-armed bandit learning," *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7228–7244, Oct. 2019.

[13] S. E. Jasper, "US cyber threat intelligence sharing frameworks," *Int. J. Intell. CounterIntelligence*, vol. 30, no. 1, pp. 53–65, Jan. 2017, doi: 10.1080/08850607.2016.1230701.

[14] D. Shackleford, "Who's using cyberthreat intelligence and how," SANS Inst., Bethesda, MD, USA, Tech. Rep. 2014-35507, 2015.

[15] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. Van Eeten, "A different cup of TI? The added value of commercial threat intelligence," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 433–450.

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

[17] McAfee. (2019). *Mcafee Threat Intelligence Exchange (Datasheet)*. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/data-sheets/ds-threat-intelligence-exchange.pdf

[18] T. Plona, K. Maxwell, and B. Varni. (2017). *Threat Intelligence in Splunk*. [Online]. Available: https://conf.splunk.com/files/2017/slides/do-you-really-know-my-adversaries-prove-it.pdf

[19] S. Alonso. (Jan. 2016). *Cyber Threat Hunting (1): Intro*. [Online]. Available: https://cyber-ir.com/2016/01/21/cyber-threat-hunting-1-intro/

[20] I. Sqrrl Data. (2016). *A Framework for Cyber Threat Hunting*. [Online]. Available: https://sqrrl.com/media/Framework-for-Threat-Hunting-Whitepaper.pdf, and https://www.threathunting.net/files/framework-for-threat-hunting-whitepaper.pdf

[21] D. Enterprise. (2019). *Automate Threat Hunting Operations*. [Online]. Available: https://www.demisto.com/solutions-threat-hunting/

[22] B. Thomas, D. Scott, F. Brott, and P. Smith, "Dynamic adaptive defense for cyber-security threats," U.S. Patent 1 558 1471, Aug. 10, 2017.

[23] McAfee. (2019). *Mcafee Investigator: Transform Analysts Into Expert Investigators*. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/data-sheets/ds-investigator.pdf

[24] Crowdstrike. (2019). *Crowdstrike Falcon Overwatch Managed Threat Hunting*. [Online]. Available: https://www.crowdstrike.com/wp-content/brochures/OverWatch_Overview_v3.pdf

[25] Y.-E. Liu, T. Mandel, E. Brunskill, and Z. Popovic, "Trading off scientific knowledge and user learning with multi-armed bandits," in *IEDM Tech. Dig.*, 2014, pp. 161–168.

[26] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration–exploitation tradeoff using variance estimates in multi-armed bandits," *Theor. Comput. Sci.*, vol. 410, no. 19, pp. 1876–1902, 2009.

[27] T. Lattimore and C. Szepesvari, *Bandit Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[28] S. Pandey, D. Chakrabarti, and D. Agarwal, "Multi-armed bandit problems with dependent arms," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 721–728.
[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[30] T.-Y. Liu, *Learning to Rank for Information Retrieval*. Boston, MA, USA: Now, 2011.
[31] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2007, pp. 271–278.
[32] Y. Seldin, C. Szepesvari, P. Auer, and Y. Abbasi-Yadkori, "Evaluation and analysis of the performance of the EXP3 algorithm in stochastic environments," in *Proc. Eur. Workshop Reinforcement Learn.*, 2013, pp. 103–116.
[33] R. Cañamares, M. Redondo, and P. Castells, "Multi-armed recommender system bandit ensembles," in *Proc. 13th ACM Conf. Recommender Syst.*, Sep. 2019, pp. 432–436.
[34] Q. Wang et al., "Online interactive collaborative filtering using multi-armed bandit with dependent arms," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1569–1580, Aug. 2019.
[35] J. Zhu, Y. Song, D. Jiang, and H. Song, "Multi-armed bandit channel access scheme with cognitive radio technology in wireless sensor networks for the Internet of Things," *IEEE Access*, vol. 4, pp. 4609–4617, 2016.
[36] M. Zhu, Z. Hu, and P. Liu, "Reinforcement learning algorithms for adaptive cyber defense against heartbleed," in *Proc. 1st ACM Workshop Moving Target Defense (MTD)*, 2014, pp. 51–58.
[37] H. Grushka-Cohen, O. Biller, O. Sofer, L. Rokach, and B. Shapira, "Using bandits for effective database activity monitoring," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2020, pp. 701–713.
[38] Y. Xia et al., "Finite budget analysis of multi-armed bandit problems," *Neurocomputing*, vol. 258, pp. 13–29, Oct. 2017.
[39] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandits with knapsacks," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 207–216.

Liad Dekel received the B.Sc. degree in computer science and the M.Sc. degree in information systems engineering from the Ben-Gurion University of the Negev. He is currently a Senior Embedded Engineer. He also manages a team of developers, which are focused on network research fields, such as ad-hoc networks and low resource platforms in extreme conditions. His main research interests include network and distributed systems, cyber threat intelligence, cyber security, and side-channel attacks.

Ilia Leybovich received the B.Sc. and M.Sc. degrees in mathematics and the Ph.D. degree from the Department of Information Systems Engineering, Ben-Gurion University of the Negev. He currently works as a Lecturer at the Ben-Gurion University of the Negev, and as a Logic and Algorithm Expert in the Hi-Tech industry. His research interests include network science, search and optimization algorithms, and mathematical applications to computer science.

Polina Zilberman received the B.A. degree in computer science from Open University, and the M.Sc. and Ph.D. degrees in computer science from the Ben-Gurion University of the Negev. One of the projects she managed is a research project that included emulating multi-step cyber attacks, threat hunting, attack hypotheses generation, topology analysis, and cyber threat intelligence. Her main research interests include traffic and topology analysis with applications to security, web intelligence, computer communication, and simulations.

Rami Puzis received the B.Sc. degree (Hons.) in software engineering and the M.Sc. and Ph.D. degrees (Hons.) in information systems engineering from the Ben-Gurion University of the Negev. He was a Post-Doctoral Research Associate with the Laboratory for Computational Cultural Dynamics, University of Maryland. He is currently a Senior Lecturer (Assistant Professor) at the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev. Over the past years, he has managed multiple research projects funded by Deutsche Telekom AG, Dell-EMC, Verint, IBM, Amdocs, Israeli Ministry of Defense, Israeli Ministry of Trade and Commerce, and leading cyber security industries in Israel. His recent research projects focused on web intelligence, cyber threat intelligence and emulation, security awareness, and attack graphs. His main research interests include network analysis with applications to security, social networks, computer communication, and simulations.