Using Bandits for Effective Database Activity Monitoring

Hagit Grushka-Cohen¹(✉), Ofer Biller², Oded Sofer², Lior Rokach¹,
and Bracha Shapira¹

¹ Ben-Gurion University of the Negev, Beer-Sheva, Israel
{hgrushka, liorrk}@post.bgu.ac.il, bshapira@bgu.ac.il

² IBM Security Division, New York, USA
{ofer.biller, odedso}@il.ibm.com

Abstract. Database activity monitoring systems aim to protect organizational data by logging users' activity to Identify and document malicious activity. High-velocity streams and operating costs, restrict these systems to examining only a sample of the activity. Current solutions use manual policies to decide which transactions to monitor. This limits the diversity of the data collected, creating a “filter bubble” over representing specific subsets of the data such as high-risk users and under-representing the rest of the population which may never be sampled. In recommendation systems, Bandit algorithms have recently been used to address this problem. We propose addressing the sampling for database activity monitoring problem as a recommender system. In this work, we redefine the data sampling problem as a special case of the multi-armed bandit problem and present a novel algorithm, C- ϵ -Greedy, which combines expert knowledge with random exploration. We analyze the effect of diversity on coverage and downstream event detection using simulated data. In doing so, we find that adding diversity to the sampling using the bandit-based approach works well for this task, maximizing population coverage without decreasing the quality in terms of issuing alerts about events, and outperforming policies manually crafted by experts and other sampling methods.

Keywords: Multi-armed bandit · Database activity monitoring · Filter bubble · Sampling

1 Introduction

Databases are at the heart of organizational IT infrastructure. For data security, privacy protection, and data leakage prevention, many organizations monitor database operations in real-time using database activity monitoring (DAM) systems. Such systems are widely used to help implement security policies, and detect attacks and data abuse. The multi-billion dollar Google-Waymo vs. Uber case¹, in which the data collected (and more specifically, data activity logs)

¹ <https://www.newyorker.com/magazine/2018/10/22/did-uber-steal-googles-intellectual-property>.

was used to document data theft, provides a good example of DAM system use and the value of such a system to an organization.

With information overload of hundreds of thousands of transactions per second, database activity monitoring systems cannot process and log all of the activity. Instead of logging all transactions, DAM systems use policies to decide which transactions to save; these policies are based on rules and users/activity groups defined by the security officer (SO) during the system setup. Policy changes require significant manual effort, resulting in a situation in which policies rarely change once they are defined making the policy vulnerable to concept drifts (such as changes in users' roles).

This static approach may cause the "filter bubble" phenomenon in which SOs (the users of the DAM system) are trapped in a subspace of options that are too similar to the defined risk profile, thereby losing the ability to explore beyond what they already know. Such filter bubbles are a known issue for recommendation systems [4,14].

To address this, we suggest incorporating the concept of diversity from recommendation systems [12] into logging policies. Unlike search engines or recommendations, sampling a more diverse group of users is not technically complicated as the user's transactions risk can be aggregated into a single score [7,9]. However, logging capacity is constrained, and by focusing solely on diversity, undocumented malicious activity in the high risk group can be missed. A solution must provide a balance between the exploration of activities and users suspected to be of low risk with the exploitation of activities and users suspected to be of high risk.

Multi-armed bandits (MABs) are sampling-based decision-making strategies used for policy setting [1,2]. To the best of our knowledge, MABs haven't been used by an anomaly detection system to set data collection policy. Since auditing and anomaly detection systems can only learn and model the data available to it, the policy determining which transactions are logged has great influence on the ability to detect anomalies, which anomalies can be detected, and what actions would be possible to audit later. MABs are used for balancing exploration and exploitation in order to maximize a reward in many domains, including reinforcement learning and recommendation systems. In the stochastic multi-armed bandit problem, the goal is to maximize the reward from a slot machine with multiple arms. Various strategies for balancing exploration/exploitation have been studied [1].

In this work, we suggest viewing the diversity problem for DAM system sampling strategies as a MAB problem, where the risk of the transactions logged is used as the reward function.

Unlike the classic MAB problem, in this special case, the risk distribution of a user is not static: it may change naturally when a user changes his/her role, or due to hacking or malware, or when an employee has been compromised. Another difference is that in each round we are not sampling one user's activity but rather are sampling multiple users to monitor, *i.e.* multiple arms are pulled in each round (the collector logs all of the transactions for a list of users).

The main contributions of this paper are summarized as follows:

- **Problem Formulation.** We formally define the problem of sampling for database activity monitoring and formulate the problem within the multi-armed bandit framework, which naturally provides a solution that addresses the need for balancing exploration-exploitation.
- **Algorithm.** We define a novel variant of the MAB problem – the Budget-Constrained-Dynamic-MAB – to incorporate capacity, pulling multiple levers in each timeframe, with a reward function which may change. We present a novel sampling strategy for sampling Budget-Constrained-Dynamic-MAB, a variant of the ϵ -Greedy algorithm, named C- ϵ -Greedy.
- **Evaluation.** Using simulated data-sets created with [8], we assess the effect of diversity on sampling policies. We introduce two evaluation measures: (i) coverage of user activity, *i.e.*, how much do we know about each user in the population, and (ii) downstream effectiveness in identifying anomalies. Using these measures, we compare a number of strong baselines and show that the proposed algorithm outperforms oracle-knowledge policy sampling and the Gibbs-prior sampling approach suggested in [8].

2 Background and Motivation

2.1 Database Monitoring

Monitoring database and file activity enables organizations to issue alerts when dangerous events have occurred, and database monitoring is implemented in many domains, including health, finance, and insurance. Security information and event management (SIEM) and DAM systems audit database activity and apply anomaly detection algorithms in an attempt to detect data misuse, data leakage, impersonation, and attacks on database systems. These systems are used by an organization’s SO to identify suspicious activity [10]. Unlike recommendation systems where the users modeled are also the ones receiving the recommendation, in DAM settings the aim is to model the database users’ activity in order to make recommendations for the SO regarding which users should be monitored.

When monitoring only users suspected of preforming risky activity without exploring and diversifying the monitored user list, the SO can only learn about the “normal behavior” of a small group of users without getting a sense of what is normal for the majority of users and activities.

2.2 Transaction Risk

When an SO assigns risk to a transaction, various contextual information is used, such as time of day, user activity profile, location (IP address), the nature of the activity (*i.e.* is it permitted), data sensitivity, and the resulting data volume.

When a DAM system is installed in an organization, these rules can be defined manually (by the SO) as a risk policy or be learned, by annotating risk scores

on some representative transaction using a classifier, such as CyberRank [9], or using an incremental regression tree learner to train a model to predict the rewards of transactions [15].

This allows for the aggregation of the overall user activity and the assignment of a single risk score for user activity within a timeframe. Methods for aggregating this are out of scope of this work, but simple aggregations, such as maximum or median risk during a particular hour, can be used.

2.3 DAM Sampling as a Multi Armed Bandit Problem

The problem of maximizing reward when sampling in the MAB problem has been explored extensively [1, 5, 13]. In multi-armed bandit problems, the optimal sampling (exploration/exploitation) strategy of playing against a slot machine with N arms each with an unknown reward distribution, is sought.

In the data-base monitoring domain, the goal is to find the optimal strategy for sampling users' database transactions, using the available resources in order to maximize risk monitoring. This does not fit the classical MAB setting as we are pulling multiple levers in every round. The scenario of performing multiple actions in each round with a constraint on budget was recently described by [17] as Budget-Constrained MABs with Multiple Plays. However, in our case, the reward distribution for each arm may change during the game and the rewards are not sparse as some reward is gained from every arm pulled (unlike vanilla MAB problems where most actions result in no reward). We name this MAB variant the Budget-Constrained-Dynamic-MAB.

Once we redefine the security sampling problem as a MAB problem we can define the sampling algorithms in MAB terms of exploration and exploitation and use reward as the objective, aligned to the task of maximizing the collection of informative and useful logs.

2.4 Using Multi-Armed Bandits for Anomaly Detection

In [15], the authors address a similar problem of diversity in anomaly detection in the domain of credit card fraud detection. They suggest adding a MAB where the arms are incremental regression tree nodes to the decision process, in order to predict the transaction score. A contextual bandit framework called GraphUCB is suggested in [6] to diversify the anomalies reviewed by domain experts on attributed networks. Unlike in our problem, both models are located downstream of anomaly detection and are used to prioritize and select the transactions investigated (reviewed) by the expert user (as the expert cannot review all the transactions).

Unlike these anomaly detection diversity problems, in the case of sampling for database activity monitoring, most data points are simply not observed by the anomaly detection system (therefore, it is not surprising that a monitoring policy which looks at just a small percentage of users misses anomalies in the rest of the population). The motivation for our work is to improve upon the common solution of human crafted policies for choosing the sample of the data seen by the anomaly detection system.

3 Problem Setting

We address the problem of diversifying the list of users whose transactions are being monitored based on the need of the system to monitor and capture all potential malicious activity, and the need of the SO to learn about the different users activity. Hence, we define the learning to diversify problem for database activity monitoring along with maximization of risk “reward” subject to the available storage and computing capacities. We now present a few notations to explain the problem. Let’s consider a policy p_t at a certain time iteration $t \in \{0, \dots, \infty\}$ which monitors S_t subset of the system users $U = \{u_1, \dots, u_n\}$ whose risk scores for time t are given by $\{r_1, \dots, r_n\}$. Our goal is to select the best policy p_t (a subset of users from the n users) for each timeframe t such that the resulting set maximizes the monitored risk score, subject to the given capacity C . Let us use indicator variables $x_{jt} = \{0, 1\}$ to denote if the user u_j is selected by policy p_t to be monitored during timeframe t , and $z_{jt} = \{0, 1\}$ to denote if the user u_j is selected by the oracle policy o_t to be monitored during timeframe t . The reward generated from policy p_t can be computed as $\rho_t = \sum_{j=1}^{j=n} x_{jt} r_{jt}$. The reward proportion r_t for time t can be computed by

$$R_t = \frac{\rho_t}{\sum_{j=1}^{j=n} z_{jt} r_{jt}}$$

The total reward can then be given by $R_T = \sum_{t=1}^{t=T} R_t$. Given the above setup, the learning to diversify problem in database activity monitoring can be mapped to maximizing R_T for a given capacity C without degrading the anomalies found.

Each user’s activity is either fully monitored or fully discarded. The anomaly detection system aims to detect advanced persistent threats (APTs) [16]. A single risk score for the user’s timeframe activity is aligned with these goals.

4 Evaluation

DAM systems have two objectives: monitoring (documenting activity) and raising alerts about anomalous activity. The MAB algorithm determines what is documented; while it is not optimized for anomaly detection, we evaluate the effect of the sampling strategy on anomaly detection (note that the manually crafted strategy does well on documenting risky transactions but fails to gather knowledge about the rest of the population). We use the following metrics to compare sampling strategies:

- **Reward.** The main objective of the sampling strategy is to maximize the monitored risk. The oracle strategy, detects the maximal risk $\rho_{opt} = \sum_{j=1}^{j=n} z_{jt} r_{jt}$ for a given capacity at time t . The reward for a time-frame is the proportion of the detected risk out of the oracle strategy’s detected risk: $R_t = \frac{\rho_t}{\rho_{oracle}}$.

- **Coverage.** Anomaly detection systems rely on historical data in order to detect anomalies. Therefore acquiring risk profiles for as many users as quickly as possible is extremely important; acquiring risk profiles is also important for avoiding the cold start problem. If we never sample the users suspected to be low risk users, the anomaly detection system will be unable to detect a change in their activity. Moreover, the SO's perception of the user's risk profile may be inaccurate or just missing. We measure how long it takes the system to collect data regarding the entire population. Therefore, we measure the percent of the population for which the sampling algorithm logged at least one timeframe of activity. Let us define $users_coverage_{t_i} = \{U_t \mid t \in \{1, \dots, t_i\}\}$ where U_t is a list of all users monitored during time t .
- **Recall of Anomalies.** Downstream of the logged data, anomaly detection is applied to user activity. Previous work [3] found that sampling adds bias to the results of anomaly detection. We measure the recall of an identical anomaly detection algorithm when applied to the different datasets produced by each sampling strategy. We normalize the recall to the recall discovered by applying anomaly detection on the full dataset (no sampling).

5 Sampling Algorithms

We compare three algorithms for the problem. The first two baseline algorithms were proposed in [8]. In addition we introduce a novel algorithm named C- ϵ -Greedy based on ϵ -greedy algorithm for the Budget-Constrained-Dynamic-MAB problem.

SO-Policy. In each round, the C users with the highest known risk are sampled (these remain constant, as no exploration is performed). Note that initialization is explained in the Experimental Settings section.

Gibbs-by-Risk. In the Gibbs-exploration strategy, suggested by [8], the user probability to be sampled is proportional to the risk estimate.

C- ϵ -Greedy. The ϵ -greedy strategy for MAB has been shown to outperform other algorithms in most settings [11]. Thompson sampling, another top performing algorithm for MAB, is not applicable to the non-discrete nature of the proposed Budget-Constrained-Dynamic-MAB. for our setting, we present the C- ϵ -Greedy strategy, where in each step, the $\epsilon \times C$ (capacity) users with the highest risk are sampled, and $(1 - \epsilon) \times C$ random users are sampled for exploration.

Algorithm 1. C- ϵ -Greedy**Data:** $U = \{u_1, \dots, u_n\}$, $R_{i,t} = \{r_{1,1}, \dots, r_{n,n}\}$, $C > 0$, $k > 0$ **Result:** list of users for time t

```

1 monitor_users = {}
2 for  $u_i \in U$  do
3    $aggrisk_{u_i} = \sum_{t=T-k}^T r_{u_i,t}$ 
4    $p_{u_i} = aggrisk_{u_i} / k$ 
5 end
6 sorted_users  $\leftarrow \text{sort}((u_1, p_1), \dots, (u_n, p_n)), \text{desc}$ 
7  $i = 0$ 
8 while  $i < \epsilon \times C$  do
9   monitor_users.append(sorted_users[i][0])
10   $i = i + 1$ 
11 end
12  $i = 0$ 
13 while  $i < (1 - \epsilon) \times C$  do
14   draw  $j$  with probability  $U \sim (1, \dots, n)$ 
15   if sorted_users[j][0]  $\notin$  monitor_users then
16     monitor_users.append(sorted_users[j][0])  $i = i + 1$ 
17   end
18 end

```

The SO policy can be viewed as a special case of the C- ϵ -Greedy strategy, where the exploitation is set to 100%.

Random Sampling. Random sampling is a baseline strategy where users are sampled completely randomly at every timeframe. This can be viewed as a special case of the C- ϵ -Greedy algorithm where the exploration is set to 100%.

6 Simulating DAM Monitoring Data

Using data collected from an operating DAM system is irrelevant in our case, mainly because it contains bias introduced by the data collection process. Due to the high velocity nature of database activity, it is impossible to log all of the activity; therefore, the data is collected using a sampling policy, and hence contains bias. Additional reasons for our decision to work with simulated data are as follows: (1) The logged data may contain organizational intellectual property (IP) and sensitive data, and as such, it is difficult to convince commercial organizations to share it. (2) Security events are not tagged in the logged data; therefore for each suspected activity, a consultation with the SO is needed. Verifying that the activity is malicious requires manual investigation of the activity and its context, making it impractical for research purposes. Using simulated data allows us to evaluate the effect of the sampling strategy using all the evaluation metrics described above (in Sect. 4). [8] introduced a simulation package

made of low complexity data, where the user activity for a timeframe is represented by a single aggregated risk score. Using simulated data also helps us avoid the bias introduced to the dataset during the collecting stage and allows us to introduce anomalies in a controlled fashion.

The simulated data is designed to mimic the properties of real users whose behavior is not random: users exhibit power log distribution of risk profile, few users produce most of the risky transactions, and transactions per user over time demonstrate a trend (see [8]).

7 Experimental Settings

We used the simulated data to produce 10 datasets using different random seeds and report on the average results. Each dataset simulates the activity profiles of 200 users for 3,000 timeframes. The capacity C was set to 10% of the numbers of users.

7.1 Initialization

The setup of a DAM system relies on the SO's familiarity with the database users, as well as his/her familiarity with the domain's potential risks. During the setup process, the SO defines a monitoring policy based on rules and user groups which represent the users and activities suspected of being the main threats to the organization's data. To examine the effect of that knowledge on the quality of monitoring and anomaly detection and to assess whether the results of the SO's manual efforts can be leveraged as a prior for the cold start scenario, we compare two settings:

(1) A setting in which there is *oracle initial knowledge* - This setting assumes that the SO, defining the policy, has perfect knowledge. To mimic this knowledge, we sample the risk for each user at time t_0 and use it as the risk prior. (2) A setting in which there is a noisy oracle - This settings assumes that the SO has imperfect knowledge about the user's activity and the risk he/she present to the organization. In such cases, we use the risk from time t_0 mixed with noise from another randomly generated user (risk distribution) as a risk prior for each user.

7.2 Introducing Security Events

The anomaly detection component of DAM systems relies on modeling user's activity and detecting activities that are incompatible with that distribution. To evaluate the effect of the sampling strategy on the anomaly detection component, we simulated security events. A security event in which a user has been compromised or abused his/her permissions is continuous (lasts more than a single timeframe) and characterized by a change in the user's risk distribution.

7.3 Controlling Diversity

To evaluate the effect of different amounts of introduced diversity, we tweak the ϵ parameter of the C- ϵ -Greedy algorithm. When ϵ equals to one, we sample only the users with the highest risk and no exploration occurs; this is essentially the SO-policy baseline. Setting ϵ to zero means that the system is in pure exploration mode, sampling users at random. Three other settings of ϵ are evaluated: 0.2 (sampling 80% of capacity at random), 0.5 (sampling 50% of capacity at random and 50% from the highest risk users) and 0.8 (sampling only 20% of capacity at random).

8 Results

Table 1. Reward and Recall for sampling strategies

Strategy	Reward (detected risk, Oracle SO)	Reward (with Noisy-Oracle SO)	Recall (detected anomalies)
SO policy	0.677	0.514	0.084
Random	0.264	0.264	0.878
Gibbs by Risk	0.514	0.39	0.56
C- ϵ - Greedy 0.2	0.479	0.48	0.834
C- ϵ - Greedy 0.5	0.703	0.702	0.779
C- ϵ - Greedy 0.8	0.864	0.863	0.67

Performance

Table 1 presents the performance for the reward and recall metrics when using the various sampling strategies. As can be seen, the C- ϵ -Greedy sampling strategy, with exploration set at 20% or 50% (ϵ set at 80% or 50%) out-performed all of the other strategies. C- ϵ -Greedy, with exploration of 20% (ϵ set at 80%) performed the best in terms of reward, with a reward of 86%. When initialized with perfect SO knowledge (assuming the SO has the correct estimation of all of the users’ activity during setup), the SO policy of sampling only the riskiest users achieved 67% of the optimal reward, however when the initialization is noisy (imperfect expert knowledge at system setup) the results degraded to 51% of the reward, while the C- ϵ -Greedy strategies did not degrade significantly.

Figure 1, which describes the reward achieved in each timeframe, shows that the C- ϵ -Greedy sampling strategy, both with the 80%/20% exploitation/exploration ratio and 50%/50% exploitation/exploration ratio, outperforms all other strategies, while random sampling consistent results in the poorest performance. In the figure, we see that C- ϵ -Greedy 80% (using 20% random exploration) takes about 100 timeframes to explore, discovering the most risky users

Fig. 1. Risk Reward per timeframe

and then exploiting gaining 20% greater reward than C- ϵ -Greedy 50%. In time-frame 460, we see another event in which a change in the user population (in this case, a non-risky user was affected by an anomaly which made that user perform high risk transactions) caused both C- ϵ -Greedy 80% and 50% to experience a drop in reward; the reward increases once the offending user is discovered.

In terms of anomaly recall, the SO knowledge policy detected only 8.5% of the anomalies, which is not surprising, as only 10% of the users are ever monitored. The highest recall was found for random sampling which detected 88% of the anomalies. The C- ϵ -Greedy sampling strategy detected between 67% and 83% of the anomalies. Gibbs-by-risk sampling had significantly lower recall with 56%.

Fig. 2. Knowledge acquisition rate

Fig. 3. Reward vs Recall as a function of exploration rate

Coverage Experiment

In Fig. 2 we compare the coverage of the different algorithms. The x-axis shows the time, and the y-axis indicates the corresponding coverage. We considered a user as “covered” when the method gathered two samples of the user. The SO policy always samples the same group of users; therefore it is constant from the

beginning as no other users are explored. With the Gibbs-by-risk method there is a probability for any user to be sampled in proportion to their risk. We observe that users who exhibit low-risk in initialization are not likely to be sampled, which slows down the exploration of all users. The strategies preferring completely random exploration had faster knowledge acquisition, depending on the proportion of the sample capacity devoted to exploration. When sampling with a 100% exploration rate (completely random), 80% exploration (epsilon_0.2), or 50% exploration the strategies had gathered two samples for most users by the 100th timeframe.

Figure 3 shows that anomaly recall improves when the exploration rate increases. However, the highest exploration rate yields low reward.

9 Conclusions

In this paper, we explored the effects of diversity when sampling activity for monitoring. We formulated the problem of sampling transactions in the security domain as a MAB problem, introducing the Budget-Constrained-Dynamic-MAB, a multi-armed bandit, where in each time-frame, we sample a number of arms defined by the capacity of the system. As a user's risk profile may change when they switch positions or become compromised, the reward probability may change during the user's life cycle. We extended the ϵ -Greedy algorithm, one of the best performing MAB approaches [5], to create the Budget-Constrained-Dynamic-MAB. Our variant, C- ϵ -Greedy splits the sampling capacity in each turn into an exploration part and exploitation part.

In the case of sampling, most data points are simply not observed by the anomaly detection system. The motivation for our work was to improve upon human crafted policies which are the most common solution in DAM systems. Therefore, it is not surprising that a monitoring policy which looks at just a small percentage of users misses anomalies in the rest of the population. It is important to note that monitoring aims at capturing a paper trail for the riskiest transactions just as much as anomaly detection, and the bandit solutions can improve recall without harming monitoring efficiency. We showed that ensuring diversity when sampling database activity is important for understanding user behavior (coverage) and enhances downstream anomaly detection without incurring adverse effects by allocating some capacity to diversity.

The C- ϵ -Greedy strategy that used 20% of the capacity for exploration was able to beat a strong oracle policy baseline in the collection of risky activity, while achieving good coverage for the entire population. The baseline method of a constant policy achieved acceptable results in terms of reward, collecting logs of risky activity, but it was sensitive to bad initialization (depending on the SO's familiarity with all of the users). Having better coverage, C- ϵ -Greedy was able to identify users whose risky behavior increased, achieving better reward overall, and was immune to poor initialization.

We found that recall is driven by the exploration part of the strategy and that completely random exploration had the best recall but the worst reward (most

of the logging capacity was spent on non-risky activity). In DAM systems, using the most common solution of human crafted policies for sampling most data points, means that many users are simply not observed by the anomaly detection system. Therefore, it is not surprising that a monitoring policy which looks at just a small percentage of users misses anomalies in the rest of the population. The motivation for our work was to improve these policies. It is important to note that monitoring aims at capturing a paper trail for the riskiest transactions just as much as for anomaly detection. We've shown that bandit solutions can improve recall without harming monitoring efficiency.

Therefore, we advise using methods for sampling that allow the SO to dedicate some capacity for exploration and diversity. We find that the C - ϵ -Greedy strategy is easy to implement, the ϵ is an easy knob to turn for enhancing exploration when needed, it supports initialization with the SO knowledge, avoiding cold start issues and keeping humans in the loop, while maximizing the audit trail for all users and supporting robust anomaly detection downstream.

References

1. Agrawal, S., Goyal, N.: Analysis of thompson sampling for the multi-armed bandit problem. In: Conference on Learning Theory (2012)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* **47**, 235–256 (2002)
3. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv. (CSUR)* **41**, 1–58 (2009)
4. Chen, L., Yang, Y., Wang, N., Yang, K., Yuan, Q.: How serendipity improves user satisfaction with recommendations? A large-scale user evaluation. In: WWW (2019)
5. Chen, W., Wang, Y., Yuan, Y.: Combinatorial multi-armed bandit: general framework and applications. In: ICML (2013)
6. Ding, K., Li, J., Liu, H.: Interactive anomaly detection on attributed networks. In: ICWD, pp. 357–365. ACM (2019)
7. Evina, P.A., et al.: Enforcing a risk assessment approach in access control policies management: analysis, correlation study and model enhancement. In: IWCMC (2019)
8. Grushka-Cohen, H., Biller, O., Sofer, O., Rokach, L., Shapira, B.: Simulating user activity for assessing effect of sampling on DB activity monitoring anomaly detection. In: Calo, S., Bertino, E., Verma, D. (eds.) *Policy-Based Autonomic Data Governance*. LNCS, vol. 11550, pp. 82–90. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17277-0_5
9. Grushka-Cohen, H., Sofer, O., Biller, O., Shapira, B., Rokach, L.: CyberRank: knowledge elicitation for risk assessment of database security. In: CIKM (2016)
10. Kaplan, J., Sharma, S., Weinberg, A.: Meeting the cybersecurity challenge. Digit, McKinsey (2011)
11. Kuleshov, V., Precup, D.: Algorithms for multi-armed bandit problems. arXiv preprint [arXiv:1402.6028](https://arxiv.org/abs/1402.6028) (2014)
12. Matt, C., Benlian, A., Hess, T., Weiß, C.: Escaping from the filter bubble? The effects of novelty and serendipity on users' evaluations of online recommendations (2014)

13. May, B.C., Leslie, D.S.: Simulation studies in optimistic Bayesian sampling in contextual-bandit problems. Statistics Group (2011)
14. Nguyen, T.T., Hui, P.M., Harper, F.M., Terveen, L., Konstan, J.A.: Exploring the filter bubble: the effect of using recommender systems on content diversity. In: WWW (2014)
15. Soemers, D.J., Brys, T., Driessens, K., Winands, M.H., Nowé, A.: Adapting to concept drift in credit card transaction data streams using contextual bandits and decision trees. In: AAAI (2018)
16. Tankard, C.: Advanced persistent threats and how to monitor and deter them. *Netw. Secur.* **2011**, 16–19 (2011)
17. Zhou, D.P., Tomlin, C.J.: Budget-constrained multi-armed bandits with multiple plays. In: AAAI (2018)