FARE: Schema-Agnostic Anomaly Detection in Social Event Logs

Neil Shah
Snap Inc.
Santa Monica, CA
nshah@snap.com

***Abstract*—Online social platforms are constantly under attack by bad actors. These bad actors often leverage *resources* (e.g. IPs, devices) under their control to attack the platform by targeting various, vulnerable endpoints (e.g. account authentication, sybil account creation, friending) which may process millions to billions of *events* every day. As the scale and multifacetedness of malicious behaviors grows, and new endpoints and corresponding events are utilized and processed every day, the development of fast, extensible and *schema-agnostic* anomaly detection approaches to enable standardized protocols for different classes of events is critical. This is a notable challenge given that practitioners often have neither time nor means to custom-build anomaly detection services for each new event class. Moreover, labeled data is rarely available in such diverse settings, making unsupervised methods appealing. In this work, we study unsupervised, schema-agnostic detection of resource usage anomalies in social event logs. We propose an efficient algorithmic approach to this end, and evaluate it with promising results on several log datasets of different event classes. Specifically, our contributions include a) *formulation:* a novel articulation of the schema-agnostic anomaly detection problem for event logs, b) *approach:* we propose FARE (Finding Anomalous Resources and Events), which integrates online resource anomaly detection and offline event culpability identification components, and c) *efficacy:* demonstrated accuracy (100% precision@250 on two industrial datasets from the Snapchat social platform, with 50% anomalies previously uncaught by state-of-the-art production defenses), robustness (high precision/recall over suitable synthetic attacks and parameter choices) and scalability (near-linear in the number of events).**

***Index Terms*—misbehavior, event log, anomaly, schema-agnostic**

I. INTRODUCTION

The prevalence of abusive behavior on online social platforms has risen considerably in recent years. As social platforms become increasingly omnipresent communication and information sharing platforms, bad actors are incentivized to engage in abusive behaviors on behalf of personal or corporate interests for gain. These behaviors can include account creation efforts, social spam by propagating content over a network [1], account hijacking efforts for purposes of humiliation or exploitation, and more. As a result, the broader research community has invested substantial resources on tackling a wide variety of application-specific anti-abuse anomaly detection problems spanning e-mail phishing attacks [2], malware detection [3], network intrusion [4], livestreaming abuse [5], social and content spam [6], fake account registration [7], account compromise [8] and more. Most of these works focus on identifying malicious users, either based on supervised learning approaches which necessitate a large quantity and diversity of labeled data, or via unsupervised approaches which compute suspiciousness scores given a pre-processed feature set.

Such works meet several major challenges in real-world anti-abuse scenarios. Firstly, supervised approaches are often unsuitable due to the sparsity or bias in existing labeled data, and the challenges associated in collecting new labels capable of handling constantly changing adversarial behaviors at test time. Lastly, unsupervised methods which rely on hand-tailored, application-specific mechanisms are challenging to curate, unwieldy to update and deprecate over time, and are prone to missing attack types beyond the biases incorporated in the rules. Worst of all, such systems are hyper-specialized to single event classes and unique features by construction (i.e. different rules for flagging supposed account hijacks, fake registrations, fake followers, etc.), with poor obvious translation between systems. Despite these challenges, practitioners every day are met with the task of deploying new interaction logging systems which process different event classes with unique features, potential threats and abuse signatures, and have limited alternatives other than these to stem attacks. This is due to the cost and time-prohibitive nature of designing custom-built anomaly detection solutions, and the lack of suitably flexible detection frameworks which can adapt to different schema definitions and data types.

In this work, we take a step back to consider the commonalities of many such problems: each can be cast as an instance of detecting anomalous *resources* (IP, device, user identifiers) behaviors over the underlying produced *events*. In this setting, resources represent limited and typically "expensive" constraints used by bad actors to conduct attacks, and events are log entries which describe fine-grained actions associated with a given event *class* (account created, login attempted, etc.) Moreover, we consider that while such event classes contain diverse feature sets (*schemas*), they often contain the same few allowable feature types(i.e. numerical like *length of session*, categorical like *true*, or textual like *e-mail address*), which we call the *typeset*. We utilize these abstractions to propose the *schema-agnostic anomaly detection* problem for event log mining, as below:

Problem 1 (Schema-Agnostic Anomaly Detection)**.** ***Given*** *a*

978-1-7281-4493-1/19/$31.00 ©2019 IEEE
DOI 10.1109/DSAA.2019.00049

log of same-class events $\mathcal{E} = \{e_1 \dots e_n\}$*, where each event* $e_i \in \mathcal{E}$ *has an associated resource identifier* r_i*, timestamp* t_i*, and* $d \geq 1$ *features* $(f_1 \dots f_d)$ *allowed by the typeset* $\mathcal{T}$*,* ***detect*** *the anomalous resources* $\mathcal{R}_A$ *and corresponding anomalous events* $\mathcal{E}_A$*.*

Note that our goal in this work is to propose abstraction and implementation for a detection framework which enables low-overhead adaptation to multiple, independent event logs with potentially *different* schemas, composed of building blocks of the aforementioned feature types. This is in order to support practitioner use-cases of quickly leveraging such a framework for a flexible, unsupervised anomaly detection solutions for a wide variety of systems and associated event classes. This is in stark contrast to traditional solutions with high specificity but limited extensibility and long development time.

To solve Problem 1, we propose the FARE (Finding Anomalous Resources and Events) approach which involves fast, modular and unsupervised components for online detection of anomalous resources, and offline identification of culprit events – see Figure 1. FARE is designed to flexibly operate on social event log data containing numerical, categorical and textual types, and with realistic, skewed data distributions prevalent in social domains in mind. We evaluated FARE on various kinds of social event logs from the Snapchat platform, and demonstrate high accuracy, generality and scalability in detecting resource anomalies and associated abusive events on event data from notable registration and login flows. Summarily, our contributions are

1) **Novel formulation:** We posit a novel formulation of the schema-agnostic anomaly detection problem, focused on identifying resource anomalies and culprit events in social event logs with schemas capable of supporting a wide range of event classes.
2) **Proposed approach:** We develop FARE, which incorporates online resource anomaly detection for near-realtime actioning using stochastic evaluations, and offline suspicious event detection using a novel event-based metric learning formulation and subgraph mining techniques.
3) **Demonstrated efficacy:** We demonstrate that FARE is accurate (100% manual validation precision on real data), robust (high precision/recall over realistic attack models) and scalable in the number of events on a variety of industrial event log datasets.

II. Related Work

We discuss prior work on two fronts: event log mining, and anomaly detection for anti-abuse usecases.

Event log mining. Most prior works in log analysis focus on the task of mining a system or console trace log which overlays multiple classes of events associated with resource identifier, thus forming sequential traces. [9] proposes an algorithm for mining trace logs to infer workflow process models akin to state machines which are used in describing how users navigate through a system. Several works tackle anomaly detection in such logs: [10], [11] constructs such models from traces

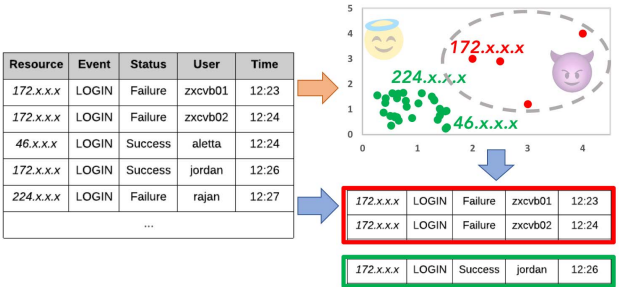


Fig. 1: An overview of our approach: Starting with an input event log containing events and associated resources, FARE enables unsupervised *online* detection of resource usage anomalies. Later, FARE's *offline* component identifies the suspected abusive events associated with pinpointed anomalous resources.

and identifies those which violate the model or have too low support as anomalies. [12] aims to correlate event sequences with systematic node failures (having encountered the same error) in cluster environments. [13] featurizes console logs based on counts of event classes in a trace, and uses offline principal component analysis to identify resources with odd class distributions. Several works focus on clustering, in which the task is to group together textually similar log messages: [14], [15] use iterative partitioning to extract such message templates, while [16] tackles this problem by mining frequent item-sets of message tokens using an apriori-like algorithm. *Our work differs in that we consider logs which contain events of a single class. Moreover, we aim to mine resources and events which are anomalous based on the associated statistical properties of event features, rather than low or no support given a state model.*

Anomaly detection for anti-abuse. Prior work in combating abuse via anomaly detection proposes both individual and group-based methods. Individual methods focus on detecting isolated entities (accounts, devices, etc.) [17], [18] take unsupervised, application-specific tacks at identifying fraudulent users and products in e-commerce graphs with probabilistic and information theoretic methods. [19], [20] use supervised methods to discover spam accounts and knowledge-base vandals based on specific feature sets. [21], [22] use semi-supervised learning in graphs to propagate suspiciousness from known review spammers and misinformative articles in order to discover others. Several works also tackle intrusion detection by using packet traffic logs [4]. Group-based methods are motivated from literature which suggests that abusive behavior often manifests in synchronous signatures, due to patterns left by scripts and bot automation. [7], [23], [24] show synchrony in various abuse contexts, including fake account handles and social graph connectivity. Most group-based methods are unsupervised. [25], [26] discern artificially bought reshare cascades and [5] finds fake livestreaming views. Other works tackle mining overly densely connected connected groups from interactions: [6], [27]–[29] use pruning and expansion

techniques to find clusters of synchronous entities in interaction graphs. [30], [31] use graph-based community detection approaches for the same. *Our work differs in that is the only one able to detect anomalies in event logs, and is also not dependent on a fixed set of application-specific features. Moreover, unlike others, our approach merges both individual and group-based elements for resource detection and event identification respectively.*

III. Schema-Agnostic Anomaly Detection

A. Problem Formulation

Large-scale online systems typically log *events* of different *classes* (i.e. account creation, login) with relevant *features* which are associated with certain actions such as endpoint requests, and capture fine-grained behaviors on the platform.

Formally, we consider a single *event log* as a multiset $\mathcal{E}$ of same-class events $\{e_1 \dots e_n\}$, where each event e_i is associated with a resource identifier $r_i \in \mathcal{R}$, timestamp t_i, and features $(f_{(i,1)}, f_{(i,2)} \dots f_{(i,d)})$, which are defined on the *schema* $F_1 \times F_2 \times \dots F_d$ given feature domains $F_1 \dots F_d$. Note that two event logs of different classes may share the same *schema* given their structures. $F_d \dots F_d$ are allowable given a *typeset* $\mathcal{T}$, which is to say that each $F_i \in \mathcal{T}$ – in this work, we consider the typeset to include categorical, numerical and textual feature types. For convenience, we also define $\mathcal{E}(r_i) = \{e_j \in \mathcal{E} \,|\, r_j = r_i\}$ to index the set of all events associated with resource identifier r_i. While we could consider any resource identifier (device, user, IP address), we use IP in this work due to its prevalence. Figure 1 shows a toy example of such an event log, with a dud field showing the *LOGIN* class, and features *(*Status, User) which could be interpreted as following the schema *Categorical, Textual*.

In practice, one major challenge is that given the sheer number, scale and most notably schema diversity of these logs across event classes, application-specific anti-abuse solutions are unwieldy to build and manage against diverse abuse vectors. The features available to leverage for detection of suspicious login events may be totally different than those available for registration events, for example, limiting extensibility of each detection system to others. Moreover, even if their schemas were alike, the signatures of suspicious behaviors may be entirely different. *Our work aims to mitigate this extensibility challenge by tackling anomaly detection in event logs in a schema-agnostic way. In this work, we consider the schema-agnosticism of a detection framework to reflect its capacity to handle multiple logs and classes of possibly distinct schemas*.

Our goals are multi-fold. Firstly, we aim to quickly (in near-realtime) identify the most abnormal resources $\mathcal{R}_A$ which produce suspiciously different events from their peers, such that we can action against them (i.e. issue challenges, rate limits, penalties) and dissuade their abusive activities. More formally, we have:

Problem 2 (Anomalous Resource Detection)**.** *Given an event log $\mathcal{E}$ of events $e_i = (f_{(i,1)}, f_{(i,2)} \dots f_{(i,d)})$ from resource $\mathcal{R}(e_i)$ at time $\mathcal{T}(e_i)$, discern the most anomalous resources $\mathcal{R}_A$.*

Note that the above problem considers the log $\mathcal{E}$ in a streaming fashion. Secondly, for those resources in $\mathcal{R}_A$, we seek to later (offline) discern the suspicious events $\mathcal{E}_A$ from normal ones:

Problem 3 (Culpable Event Identification)**.** *Given a set of anomalous resources $\mathcal{R}_A$ and associated candidate event set $\mathcal{C} = \{e_i \in \mathcal{E} \,|\, r_i \in \mathcal{R}_A\}$, identify the suspected culpable events $\mathcal{E}_A \in C$.*

Note that the desired time-to-action for Problems 2 and 3 are different in practical abuse handling (urgent mitigation but delayed fine-grained, post-mortem analysis), and thus we address them distinctly.

B. Proposed Approach: FARE

To tackle these two problems, we propose FARE. FARE has both an online anomalous resource detection component addressing Problem 2 (see Section III-B1), as well as an offline culpable event identification component addressing Problem 3 (see Section III-B2).

1) Detecting Resource Anomalies: The first component of our approach involves online, near-realtime identification of abnormally behaving resources. This enables us to employ dissuasion measures based on early detection results to mitigate continued and future negative behaviors proactively.

Our high-level approach to this task involves extracting and maintaining live *resource-level* representations, which are summaries of the *event-level* features associated with each resource's events. As we adapt these resource-level representations, we leverage a trained unsupervised anomaly detector to identify the outlying resources and are enabled to penalize these resources until we believe their behavior is sufficiently normal. We elaborate below.

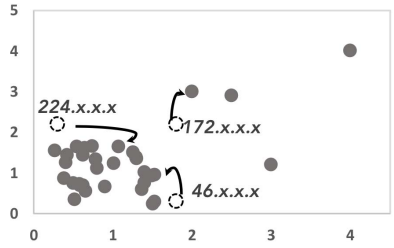
Choosing summary statistics. We propose to represent a resource r_i with an ensemble of resource-level statistics on event-level features $f_{(i,1)} \dots f_{(i,d)}$ of the associated events $\mathcal{E}(r_i)$, such that the resulting resource-level representation $\vec{r_i} \in \mathbb{R}^k$. Note that $d \neq k$ in general, as it is possible to extract any number of summary statistics from even a single event-level feature.This leads us to the question, which summary statistics should we extract?

Recall that event-level features can be categorical, numerical or textual. In this work, we handle text by constructing numerical and categorical meta-features. Thus, we limit our scope to choosing summary statistics for these feature types only. We consider several common candidate statistics (see Table I), positing that not all are equally suitable for the anomalous resource detection task.

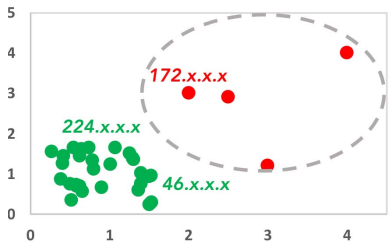
Firstly, we consider that *sampling distributions are not invariant to sample size*. Although sample statistics converge in the limit if it exists given the Law of Large Numbers, in practice we have many small samples. Moreover, given an arbitrary data distribution and statistic, the sampling distribution may shift with sample size (event count); see Figure

Resource	Event	Status	User	Time
172.x.x.x	LOGIN	Failure	zxcvb01	12:23
172.x.x.x	LOGIN	Failure	zxcvb02	12:24
46.x.x.x	LOGIN	Success	aletta	12:24
172.x.x.x	LOGIN	Success	jordan	12:26
224.x.x.x	LOGIN	Failure	rajan	12:27
...				

(a) Incoming event stream

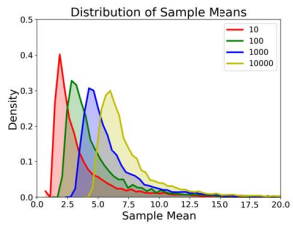


(b) Near-realtime resource feature updates

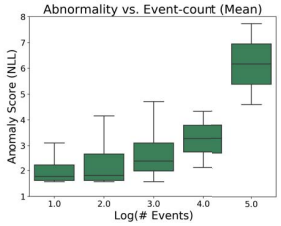


(c) Anomalous resource detection

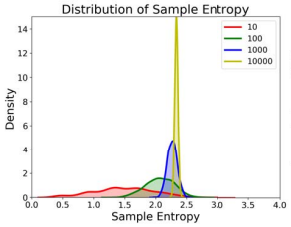
Fig. 2: Given a stream of events (a), FARE maintains and rapidly updates multi-dimensional resource (IP) representations with incoming events (b) to detect abnormal resources in a near-realtime, unsupervised fashion (c).



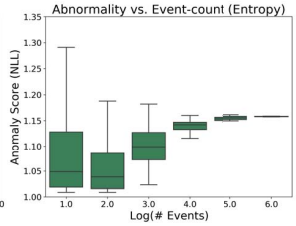
(a) Sample Mean



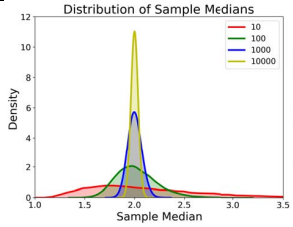
(b) ↑ #events, ↑ abnormality



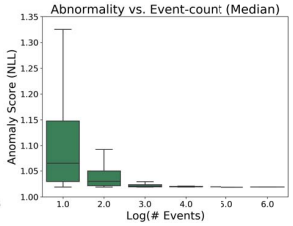
(c) Sample Entropy



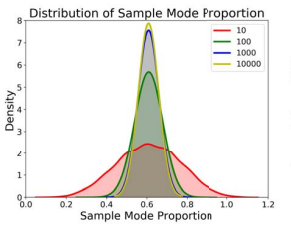
(d) ↑ #events, ↑ abnormality



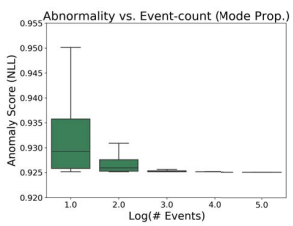
(e) Sample Median



(f) ↑ #events, ↓ abnormality



(g) Sample Mode Proportion



(h) ↑ #events, ↓ abnormality

Fig. 3: Certain summary statistics can bias anomaly detectors towards marking high event-count resources as most anomalous in realistic, skewed numerical (left) and categorical (right) data settings, despite fixed event-generating processes. (a)/(b) and (c)/(d) show that moment and entropy features suffer, while (e)/(f) and (g)/(h) show that quantile and frequency features triumph in such cases (box-whisker plots show spread of anomaly scores for resources with near-equal # events). These differences are caused by notable shifts in statistics' sampling distributions over varying event counts (see legends) in (a)/(c), versus more-aligned distributions in (e)/(g).

3a for an example. The implication is that when comparing two resources r_i and r_j, $\vec{r_i}$ and $\vec{r_j}$ may be significantly different *even if the resources have the same underlying event-generating process*, due to differences in event counts $|\mathcal{E}(r_i)|$ and $|\mathcal{E}(r_j)|$.

Secondly, *resource usage is highly skewed* in practice. Most resources are associated with just a few events (less prolific), but a few resources are associated with many (more prolific). Such usage distributions often resemble Zipf's law, in which the popularity of resources with k events, p_k, is inversely proportional to their (exponentiated) frequency ranks: $p_k \propto k^{-\alpha}$. The implication is that there will be many more vectors like $\vec{r_i}$ than $\vec{r_j}$, if $|\mathcal{E}(r_i)| < |\mathcal{E}(r_j)|$.

This begets a key observation: assuming skewed resource usage, if a statistic's sampling distribution shifts notably with event count, then more prolific resources will be marked as anomalous by an unsupervised detector, even if they are the closest distributional approximation of the underlying event-generating process. The ramifications of this outcome are disastrous, and would negatively impact popular, shared resources like universities, companies and small countries with limited network infrastructure. Similar analogs exist for different resource identifier types (i.e. highly-engaged power users for usernames, shared or common devices).

In reality, a statistic's sampling distribution, sample size sensitivity, and anomaly detection impact depends on many factors.This is a complex interplay, and not the main focus of this paper. However, we take these considerations into account in choosing suitable summary statistics, and keep in mind that many real-world data distributions are highly skewed, and thus our statistics should robustly handle skewed data. As such, we experimented with the statistics listed in Table I in a simulated, but realistic, setting in which both event-level features as well as underlying resource usage distribution were skewed.

Numerical	Categorical
Extrema (*min*, *max*)	Inequality (*entropy*, *gini*)
Moments (*mean*, *variance*)	Distincts (*unique_prop*)
Quantiles (Q_1, Q_2, Q_3)	Frequents (*mode_prop*, *sec_prop*)

TABLE I: Candidate statistics for event-level summarization.

Specifically, we modeled cases in which resource event-count was $Zipf(\alpha = 2.0)$ distributed, and events had a single feature f_1. We considered skewed categorical and numerical f_1, such that $f_1 \sim Zipf(\alpha = 2.0)$ and $f_1 \sim Pareto(\alpha = 2.0)$ respectively; previous works [32], [33] show that $\alpha \approx 2.0$ holds for many real datasets. We then used a 1-D kernel density estimator to evaluate anomaly (likelihood) scores of varying event-count resources.

These scenarios empirically justify our observations regarding the interplay between statistic choice, resource skewness and anomaly detection performance: Figure 3 reflects how sampling distributions of *mean* (a) versus *median* (e) for numerical data and *entropy* (c) versus *mode_prop* (g) for categorical data impacted anomaly score distribution in (b)/(d)/(f)/(h) over resources with varying event-counts. Specifically, one can see how the *mean* and *entropy* features produced unintuitive anomaly detection results, where the anomaly score (negative log-likelihood) increases with the event count in (b)/(d), unlike for *median* and *mode_prop* in (f)/(h). We exclude similar plots for other features from Table I for space reasons, but we rationalize similarly for including and excluding other features for summarization: in line with human intuition, we chose to use features for which *increasing* event count resulted in *decreasing* anomaly score given fixed event-generating process.

Numerical. Extrema (*min*, *max*) and moments (*mean*, *variance*) showed significant bias against prolific resources. These features are sensitive to draws from the tail, which are likely with large samples. Conversely, quantiles performed well, given lower sensitivity to outliers; this facet better aligns with our goal to describe the abnormality of a resource's *aggregate* behavior, rather than that of a few events. Thus, we use Q_1, Q_2 and Q_3 to describe each numerical feature.

Categorical. Inequality measures such as *entropy* and *gini* (Gini score, and distinct-value statistics such as *unique_prop* (ratio of unique values) were in heavily biased against prolific resources. Conversely, frequency statistics such as *mode_prop* and *sec_prop* (ratios of first and second most-frequent values) behaved intuitively. We use *sec_prop* in addition to *mode_prop* as [7] showed its use in detecting bimodal abusers who switch between multiple patterns.

For textual features, we extract three meta-features. Firstly, we consider *enc_pattern* (encoding pattern, as described in [7]), which maps each lowercase character to 'L', uppercase to 'U', digit to 'D' and other to 'O'. This produces a categorical feature whose distribution describes textual structure, and can indicate structural similarity. Secondly, we use *str_len*, a numerical feature denoting text length, which can identify excessively long or short texts, as well as too-uniform text. Finally, we use *str_count*, which maintains counts of each unique string over the set of resource events. The distribution of this feature describes recurrence of the *exact* same text. From these features, we extract numerical and categorical statistics.

Maintaining resource representations. Given chosen statistics, we next need to efficiently maintain and compute live representations for all $r_i \in \mathcal{R}$. We next discuss implementation choices for handling adaptive computations for each statistic. We aim to avoid keeping the full stream in memory, as this would be costly. Moreover, we aim for quick and cheap computations in terms of time complexity.

Numerical. We could naïvely compute quantiles on an evolving data stream by storing all elements and running the necessary quantile computations as needed. Unfortunately, this results in $O(|\mathcal{E}(r_i)|)$ space footprint for a resource with $|\mathcal{E}(r_i)|$ events, which can be prohibitive in large datasets. Moreover, naïve quantile computation involves sorting and iterating through the data, resulting in $O(|\mathcal{E}(r_i)| \log(|\mathcal{E}(r_i)|))$ time. In response, a several sketch-based quantile approximation algorithms approaches have been proposed given fixed space; [34] gives an overview. Unfortunately, these approaches become infeasibly space intensive to maintain per resource when $|\mathcal{E}(r_i)|$ is generally small, as in real use-cases [35]. As a result, we avoid sketch-based approaches. In practice, we find that reservoir sampling [36] is suitable, allowing $O(k)$ space for a k-size reservoir, and tuning k to minimize space or maximize accuracy given constraints. Quantiles can then be computed on the reservoir in $O(k)$ time per resource using a selection algorithm.

Categorical. As with numerical data, the most naïve option for computing frequencies is to store all elements and evaluate element proportions by counting as needed, which requires $O(|\mathcal{E}(r_i)|)$ space for a resource with $|\mathcal{E}(r_i)|$ events. The naïve proportion computation method would involve tallying and ordering unique element counts, with time complexity $O(|\mathcal{E}(r_i)| + u \log(u))$ for u unique elements. As noted above, we avoid sketch approximations (for example, Count-Min [37]) as they pose excessive space overhead for most resources. We observe that since the number of categories/uniques is often limited and known a priori in practice, we can do much better by maintaining a hashmap of uniques and seen counts on the fly while maintaining a separate counter for total seen elements, requiring only $O(u)$ space. This approach works best when distributions are skewed, since $u \ll |\mathcal{E}(r_i)|$. Frequency statistics can then be computed over the hashmap using $O(u \log(u))$ time. Given a sample size and underlying distribution, we can express $\mathbb{E}[u]$ as

Lemma 1 (Categorical Uniques)**.** *Given a size-n sample from a feature with categorical distribution P over j categories, the expected number of unique elements $\mathbb{E}[u] = \sum_{i=1}^{j} 1 - (1 - p_i)^n$, where p_i denotes the probability mass on the i^{th} category of P.*

At evaluation time, we can use these strategies to quickly produce a vector representation $\vec{r_i}$ for each $r_i \in \mathcal{R}$. This better aligns our problem with traditional vector-based anomaly detection.

Pinpointing anomalous resources. Given resource representations, our next goal is to leverage these to pinpoint anomalous resources. As our goal from Problem 2 requires us to *quickly* identify misbehaving resources for enforcement, our approach must detect anomalies on-the-fly. This suggests the use of a stream-suitable anomaly detection model, in which we can easily evaluate the abnormality of any single resource at a given time.

Model Structure. Our approach involves the use of a pre-trained, windowed detection model. In a nutshell, we partition the event stream $\mathcal{E}$ into consecutive and non-overlapping time-windows, such that detection on a current window depends on a detector built from a prior window. Formally, we consider that at any given time we are concerned with two subsets of $\mathcal{E}$, called the model *reference window* $\mathcal{M}_{(t_0,t_1)} = \{e_i \in \mathcal{E} | t_0 \leq \mathcal{T}(e_i) < t_1\}$ from which we build an anomaly detector $\mathcal{D}_{\mathcal{M}}$ over the respective resource representations produced at the end of $\mathcal{R}$, and the *current window* $\mathcal{C}_{(t_1,t_2)} = \{e_i \in \mathcal{E} | t_1 \leq \mathcal{T}(e_i) < t_2\}$ in which we are ingesting events in real-time and maintaining live resource representations. Then, given $\mathcal{D}_{\mathcal{M}}$, we can quickly evaluate any $\vec{r_i}$ in the current window $\mathcal{C}$ each time it changes, accounting for a newly seen event. Upon the completion of $\mathcal{C}$'s timeframe, we reset resource representations for the next window.

Practical Considerations. In practice, we posit that the associated detection model $\mathcal{D}_{\mathcal{M}}$ need not be updated very frequently over new reference windows, because aggregate resource behaviors are not expected to be highly dynamic. Thus, we could utilize the same $\mathcal{D}_{\mathcal{M}}$ for many subsequent "current" $\mathcal{C}$ windows as time progresses, further amortizing the model training cost. Moreover, we can augment our resource evaluation strategy in two ways to trade-off liveness and scalability: Firstly, we can impose an event-count based detector training and evaluation threshold τ, which limits our inputs to only resources with $\geq \tau$ events. This enables us to (a) improve our estimates of normal behavior by discarding noise from low event-count resources, (b) avoid penalizing lightly used "personal" resources, and (c) greatly cheapen the computational cost of model training and evaluation. Secondly, we can introduce stochasticity into the resource evaluation step so that at each representation update, we evaluate the resource with a p_{eval} chance. The main benefit is that we can avoid frequent evaluations for high event-count resources, and proportionally reduce runtime at the cost of inducing a stochastic delay in evaluation. We can express the probability of the length of such a delay as follows:

Lemma 2 (Skipped Evaluations)**.** *Given an evaluation probability p_{eval}, the probability of waiting "too long" ($> s$ events beyond the expected) before an evaluation is*

$$P(X - \mathbb{E}[X] > s) = (1 - p_{eval})^{s + p_{eval}^{-1}}$$

This probability decays *exponentially* in observed event-count. To further quantify the benefits of adjusting τ and p_{eval}, we can write the ratio of expected evaluations to total events:

Lemma 3 (Event Evaluation Ratio)**.** *Given n events across m resources $r_1 \dots r_m$ such that event count $|\mathcal{E}(r_i)|$ is distributed as C, an event count evaluation threshold τ and probability p_{eval}, the expected ratio of evaluated to total events is*

$$\mathbb{E}[n_{eval}] = n^{-1} \cdot m \cdot p_{eval} \cdot \sum\nolimits_{k=\tau}^{K} [p_k \cdot (k - \tau + 1)]$$

where $K = max(dom(C))$ and $p_k = P(|\mathcal{E}(r_i)| = k)$.

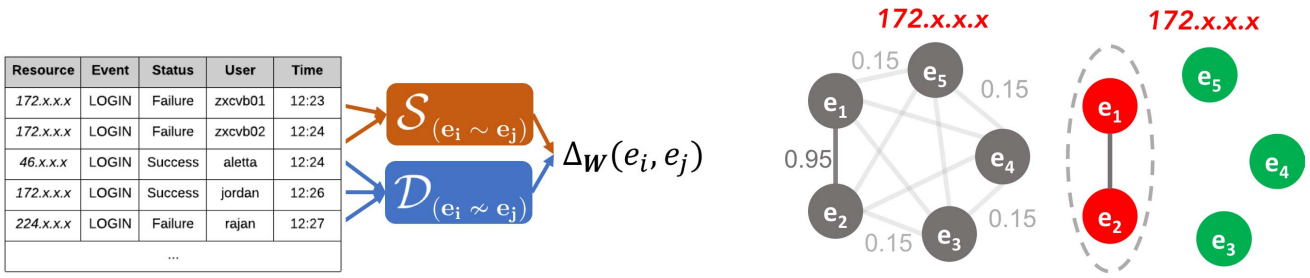
Proofs for all lemmas included in Section VI.

Our approach lets us extensibly use any suitable anomaly detector as $\mathcal{D}_{\mathcal{R}}$. In our experiments, we utilize Isolation Forest [38], due to its fast, near-linear train and test time complexity with constant model footprint. This approach gracefully handles continuous and discrete data due to its random splitting strategy, unlike density estimation approaches which suffer given mixed data. We transform unnormalized resource scores to $(0, 1)$ using the empirical CDF given by $\mathcal{M}$, enabling us to flag resources above a threshold (i.e. top $\epsilon\%$) in online fashion. These resources constitute our abnormal set, $\mathcal{R}_A$. They can then be challenged, rate-limited or otherwise penalized.

2) Identifying Culpable Events: Upon proactively having identified anomalous resources $\mathcal{R}_A$, our next goal is to identify the events culpable for abnormality $\mathcal{E}_A$ for further investigation. Given that these events have already had to occur for the resource to be flagged as abnormal, we tackle their identification in an offline manner, once all events from a window $\mathcal{C}$ have been observed.

Our high-level approach for this task stems from the intuition that groups of too-similar events are suspicious. A considerable amount of prior literature substantiates that actions taken by bad actors are interrelated and often in "lockstep," meaning that they are overtly synchronous. Leveraging this insight of excessive similarity as a negative signal, our goal becomes to identify and group together events taken by resources to isolate suspicious from normal events. Our high-level approach involves constructing and learning a principled, event-based distance metric from optionally available practitioner input, and subsequently identifying clusters of events for which the inferred distance is excessively low (implying high similarity), as shown in Figure 4. We elaborate below.

Computing event similarity. To quantify similarity, we first require a notion of distance. Euclidean distance is one of the most commonly used metrics over points in $\mathbb{R}^d$, satisfying the formally defined metric properties of non-negativity, identity of indiscernibles, symmetry and subadditivity. However, Euclidean distance naturally treats all features with equal weight. In reality, this assumption may not be ideal for the distance-reliant task in question; in clustering and anomaly detection applications, "closeness" in certain features often reflects proximity better than others; for example, tobacco



(a) Event-based distance metric learning

(b) Identifying highly synchronous event clusters

Fig. 4: FARE incorporates learning of a between-event distance metric which obeys human intuitions regarding feature importance (a), and produces coherent clusters of suspected events which are excessively similar (b).

consumption may be more important than age, in computing distance for the purpose of separating lung cancer-prone and healthy patients. To make distance metrics sensitive to such differences, prior work introduces the *distance metric learning* problem, which aims to learn distance metrics which respect human intuition in the form of marked "similar" and "dissimilar" point sets, $\mathcal{S}$ and $\mathcal{D}$. [39] proposes learning a weighted Euclidean (i.e. Mahalanobis) distance metric for points in $\mathbb{R}^d$ of the form $\|\vec{x}-\vec{y}\|_{\mathbf{W}} = \sqrt{(\vec{x}-\vec{y})^T\mathbf{W}(\vec{x}-\vec{y})}$, where identity $\mathbf{W}$ reflects equal feature weighting. They propose a convex objective to learn $\mathbf{W}$ such that intra-set distances in $\mathcal{S}$ and $\mathcal{D}$ are minimized and maximized, respectively. Our task of identifying culpable events suggests the use of such a learned metric sensitive to expert input, but between complex *events* rather than points. Recall that events are defined over numerical, categorical or textual features, such that $e_i = (f_{(i,1)} \dots f_{(i,d)})$ where $f_{i,d} \in F_d$. Then, our goal is to formulate a weighted metric between events, and learn the weights. In doing so, our metric must obey the aforementioned properties to allow sensible notions of proximity; without these, our concept of inter-event distance may be unintuitive.

We can show that in fact, we can devise such a metric obeying the required properties by construction. Specifically,

Lemma 4 (Event-based Distance Metric)**.** *If $\mathcal{M}_1 \dots \mathcal{M}_d$ define distance metrics on feature spaces $F_1 \dots F_d$ respectively, then for any two events $e_i = (a_1 \dots a_d)$ and $e_j = (b_1 \dots b_d)$ where $a_k, b_k \in F_k$, $\mathbf{W} = diag(w_1 \dots w_d) \succeq 0$ and $\vec{h} = \left[\sqrt{\mathcal{M}(a_1,b_1)} \dots \sqrt{\mathcal{M}(a_d,b_d)}\right]$*

$$\Delta_{\mathbf{W}}(e_i, e_j) = \vec{h}^T\mathbf{W}\vec{h}$$

defines a distance metric on the event-space $F_1 \times F_2 \dots \times F_d$.

With this, we propose a novel objective to learn $\mathbf{W}$:

$$\min_{\mathbf{W}} \sum_{(e_i,e_j)\in\mathcal{S}} \Delta_{\mathbf{W}}(e_i,e_j)^2 - \gamma \log\left(\sum_{(e_i,e_j)\in\mathcal{D}} \Delta_{\mathbf{W}}(e_i,e_j)\right) + \beta\|\mathbf{W}\|_F \text{ subject to } \mathbf{W} \succeq 0 \text{ and } \mathrm{Tr}(\mathbf{W}) = 1$$

This objective is solvable globally optimally due to its convexity. We square the first term to avoid producing deficient solutions. We can tune γ to increase weight of maximizing dissimilarity and, β to avoid feature over-reliance. In practice, this enables practitioners to specify $\mathcal{S}$ and $\mathcal{D}$ in order to guide the clustering to respect expert intuition (see Figure 4a). From the inferred metric $\Delta_{\mathbf{W}}$, we define a similarity function $\sigma_{\mathbf{W}} = 1 - \Delta_{\mathbf{W}}$, which we threshold to establish excessive, suspicious similarity. In practice, $\mathcal{D}$ can be chosen to consist of random event pairs, while $\mathcal{S}$ can be selected from previously caught abusive events, or curated from events from newly discovered anomalous resources. Moreover, multiple curated instances of $\mathcal{S}$ can be used to learn multiple notions of similarity between events, enabling interpretable categorization of downstream discovered event clusters as different abuse signatures (e.g. $\sigma_{\mathbf{W}}$ may uncover a cluster of events associated with targeted account hijacking, whereas $\sigma_{\mathbf{W}'}$ may uncover another cluster associated with distributed account hijacking). If $\mathcal{S}$ and $\mathcal{D}$ are unavailable, we can defer to equal weighting using an identity $\mathbf{W}$.

Note that given the above result, we can utilize *any* suitable metric on a particular space F_k. This lets us flexibly determine the suitability of choices according to the appropriateness on the relevant data. In this work, we use a modified ℓ_1 metric $\frac{|a_k - b_k|}{1+|a_k-b_k|}$ on numerical features, the discrete metric $a_k \neq b_k$ (0 if equal, 1 if not) on categorical features and the Jaccard metric $1 - \frac{|a_k \cap b_k|}{|a_k \cup b_k|}$ on textual features. Under these definitions, each metric is bounded on $(0, 1)$, which is intuitive and advantageous for optimization.

Extracting suspicious event clusters. Given a similarity function $\sigma_{\mathbf{W}}$, we next aim to cluster excessively similar events together for each abnormal resource $r_i \in \mathcal{R}_A$. To do this, we construct a cross-event similarity graph G_i for each r_i, where an edge between events e_i and e_j indicates that $\sigma_{\mathbf{W}}(e_i, e_j) \geq \eta$, where η is an empirically tuned parameter which controls clustering aggressiveness. Note that $\eta = 0$ and 1.0 correspond to full density or sparsity, respectively.

Upon constructing such a graph for each resource, we can apply any suitable subgraph-mining algorithm; we use connected components, which extracts coherent clusters in which each event e_i has $\geq \eta$ similarity to at least one other event e_j. Our intuition is that each such cluster corresponds to the signature of a synchronized attack. For example, a re-

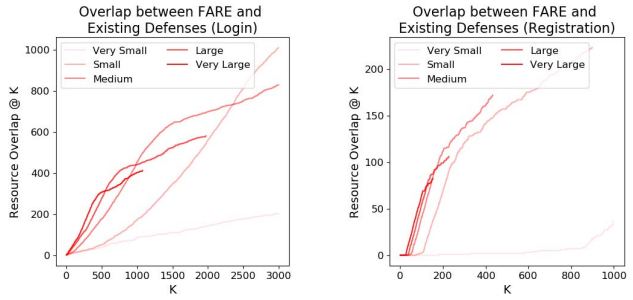


Fig. 5: FARE discovers many anomalous resources previously uncaught by existing production-level login/registration defenses. Reduced overlap with existing systems for "small" resources indicates our improved ability to detect subtle attacks.

source might have two resulting connected components, which reflect two brute-force authentication attacks corresponding to different username patterns. Conversely, if σ_W does not highly weight the username feature, then perhaps we would observe only a single component which consists of the merged attack events. Further yet, if we find that the input graph has zero density, then no two events seem similar enough to be constituted as an attack. Figure 4b shows a toy example of an inferred attack. All clustered events produced from applying this approach to resources in $\mathcal{R}_A$ populate the suspected event set $\mathcal{E}_A$. Once such event clusters are flagged, they can be triaged into manual review tickets, discredited from metrics, and negative associated actions can be reversed.

IV. Evaluation

Our evaluation aims to answer three main questions:

- **Q1. Accuracy:** Can FARE accurately detect resource/event anomalies on multiple types of log data?
- **Q2. Robustness:** How does FARE's performance shift with varying parameter choices?
- **Q3. Scalability:** Does FARE scale amenably in practice?

We discuss these after detailing our experimental setup.

A. Experimental Setup

We first discuss our dataset and computing environment.

Datasets. We consider two production, industrial-scale event log datasets from the Snapchat social platform, reflecting (a) *31 million* account authentication attempts from *9 million* IPs (`LOGIN`) over 4 features (username, whether the login succeeded, whether the target user had an active login, and a proprietary risk score), and (b) *1.1 million* account registrations (`REGN`) from *900 thousand* resources over 7 features and (c) modified registration logs with simulated, synthetic attacks (`SYNTH`) over 7 features (username, display name, e-mail handle/domain, and birth date/month/year).

`SYNTH` datasets were simulated with a 95/5% split of 5000 total designated normal/abnormal resources, with resource event counts $Zipf(\alpha = 2.0)$ distributed. All resources' events were initially randomly sampled from real `REGN` events (amortizing any inherent abnormalities). Anomalous resources were then "corrupted" by imputing events as follows (executed independently for each resource):

1) Randomly generate field value for a template "attacker" event.
2) Replace `atk_scale` ratio of benign with attacker events.
3) For each attacker event besides the template, randomly generate values for `atk_hetero` of the other fields.

This let us control for both (a) the "attack scale" and (b) "attack heterogeneity" for each resource. These simulations reflect real abusive behavior signatures previously discovered by domain experts, which suggest that obvious abuse has low heterogeneity and high blatancy, but intelligently executed abuse often has high heterogeneity and low-scale. Section VI-B1 gives more details on simulation parameters and their implications on attack patterns.

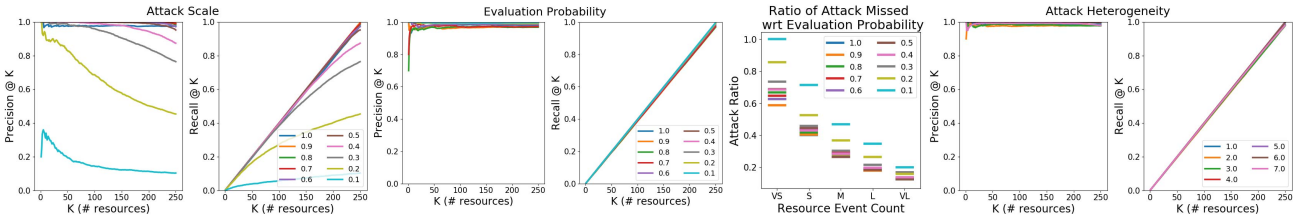
Environment. All experiments were executed using a Google Compute Engine instance with 64 cores and 400GB RAM. Prototyping/experimentation was done in Python. Accuracy and runtime results were averaged over 5 iterations, and reflect offline evaluation.

B. Accuracy

We discuss FARE's accuracy in two experiments on both `LOGIN` and `REGN`: (a) manual validation, and (b) comparison with existing, state-of-the-art production defenses.

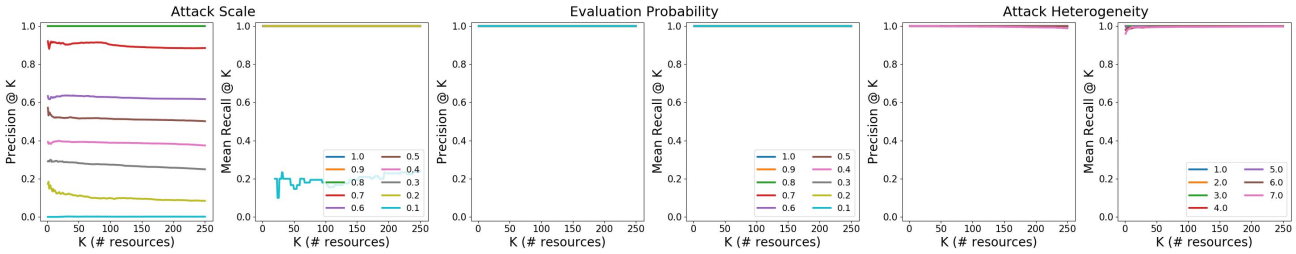
Manual validation. Given that labeling is a labor-intensive task, we limit our manual validation to the top-K ($K = 250$) resources reported by FARE on each dataset. We obtained a strong 100% precision@250 on both `LOGIN` and `REGN` datasets, with all top-ranked anomalous resources behaving abusively according to domain experts. Resources were labeled according to perceived event similarities, examination of other (non-processed) account behaviors, too-short inter-event times, and other proprietary signals. On `LOGIN` data, we observed a gamut of abusive behaviors, including brute-force attempts (high failure rate over many repeated attempts), account compromises (login attempts from unfamiliar environments and failed 2FA challenges), and third-party application usage (frequent account switching or successful same-account logins every few seconds). On `REGN` data, we observed attackers creating fake accounts to send spam or to use at a later time for nefarious purposes. The accounts were often created with highly similar features such as names and contact details.

Comparison with existing defenses. Prior work has not directly tackled the setting we discuss, hence producing no apparent baselines. In lieu, we compared FARE's anomalous resource detection performance with defense logs from existing, state-of-the-art industrial detection systems, which include (a) highly specialized and expert-curated detection rules which leverage numerous additional signals not consumed in our approach, (b) custom-tailored rate limits, and (c) IP filters for cloud environments, known botnets and more. Specifically, we compared the overlap@K between FARE's ranked results and the entire (unranked) set of flagged IPs from which we



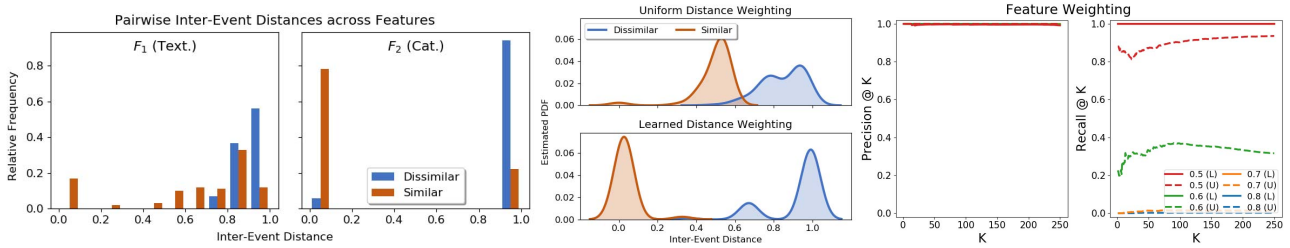
(a) Attack Scale (b) Evaluation Probability (c) Attack Heterogeneity

Fig. 6: FARE demonstrates strong precision/recall on anomalous resources in synthetic attacks, which are unaffected by evaluation probability (b, left) and attack heterogeneity (c), and demonstrates intuitive decline as labeled anomalous resources become more "normal" given smaller attack scale (a). (b, right) shows the "liveness" of resource detection decreases with evaluation probability.



(a) Attack Scale (b) Evaluation Probability (c) Attack Heterogeneity

Fig. 7: FARE identifies culpable events in synthetic attacks with strong precision/recall, which are robust to evaluation probability (b) and attack heterogeneity (c). Moreover, (a) shows that event identification performance is very strong for large-scale attacks, and intuitively decreases for low-scale attacks, since anomalous and normal resources become too alike, and hence similar and dissimilar events can become indiscriminable (see Section IV-C for details). Careful similarity thresholding can greatly improve performance in practice.



(a) Imbalanced inter-event distances across features (b) Metric learning effect (c) Improved culpable event identification

Fig. 8: FARE flexibly handles differences in feature importance for event clustering. (a) shows that pairwise distances within similar/dissimilar sets differ. (b) demonstrates that our learned weighting scheme polarizes distances between similar and dissimilar events, unlike uniform weighting. (c) exhibits that learned weights lead to better recall in similar event detection.

rejected endpoint requests (events) on suspicion of abuse from *any of the above defenses* over the same time period. Moreover, we computed this metric over ranked lists of resources, conditioned on varying event count profiles, from very small to very large (specifics excluded for security reasons), to examine sensitivity of existing systems and FARE to resource size. Our results, shown in Figure 5 indicate that while our approach's results are correlated with existing defenses at all resource sizes, we discover nearly 60-100% more suspicious resources in LOGIN (REGN) data at conservative values of $K = 500$ (200), respectively. Moreover, FARE is able to detect smaller-scale attacks much better than existing defenses, as evidenced by the lesser overlap with existing systems at smaller resource size profiles, and our manually validated observations on resource suspiciousness.

C. Robustness

We next discuss FARE's robustness to varying data and detection settings. We first study FARE's performance against different attack types and evaluation settings. Next, we demonstrate how performance is improved by event-based metric

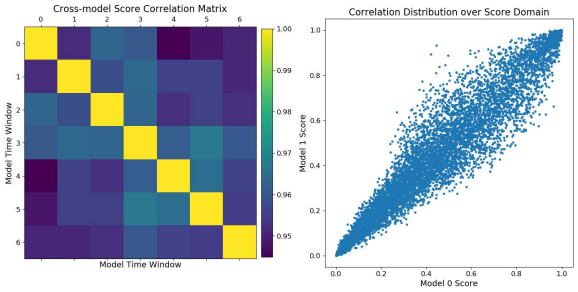


Fig. 9: FARE is able to amortize model training costs over time by exploiting high temporal correlation in resource usage behaviors (a), and hence anomaly detection results (b).

learning. Lastly, we discuss observations about model retraining.

Performance on SYNTH. We use a suite of SYNTH datasets with varying atk_scale and atk_hetero to study performance against varying attack factors. We also vary p_{eval} (detection stochasticity) to demonstrate the influence of stochasticity on detection performance and responsiveness. By default, we use atk_scale= 1.0, $p_{eval} = 0.1$ and atk_hetero= 1; while varying each parameter, the other two are kept fixed.

Figure 6 summarizes results for FARE's anomalous resource detection component. 6a shows that performance decreases when atk_scale decreases; intuitively, tiny attacks do not sufficiently discern designated anomalous resources. 6b shows that decreasing p_{eval} does not impact detector performance, but leads to delayed detection (i.e. most attack events occur before detection). However, if staleness is tolerable, reducing p_{eval} can be advantageous computationally. 6c shows that FARE's is able to detect both simple/diverse attacks with strong precision and recall.

Figure 7 summarizes the downstream performance of these parameter settings on FARE's culpable event identification component. In all cases, $\mathcal{S}$ is populated with 100 randomly chosen within-resource pairs from the top 5 anomalous resources' events, and $\mathcal{D}$ is populated with 1000 completely randomly chosen pairs; η is chosen as the median pairwise similarity score in $\mathcal{S}$. Note that these settings are chosen to promote reproducibility given the attack model; in practice, curating $\mathcal{S}$ and $\mathcal{D}$ carefully would greatly improve discriminability and resulting precision and recall. 7a demonstrates commensurately decreasing performance as atk_scale decreases; precision declines because of (a) worsened abilities to estimate a good similarity threshold ($\mathcal{S}$ and $\mathcal{D}$ become very alike given the above selection strategy), and (b) little discriminability between anomalous and normal resources. η can of course be adjusted for higher precision and lower recall, but we show results here only for a single setting given space constraints. 7b shows that, as expected, p_{eval} does not influence culpable event identification beyond its influence on anomalous resource detection. Finally, 7c shows that regardless of attack heterogeneity, FARE can maintain near-perfect performance due to a suitably learned metric.

Effects of event-based metric learning. One may wonder what exactly the advantage of learning a suitable event-based metric is. Figure 8 substantiates this choice. 8a shows the distribution of pairwise distances between REGN events across two different features F_1 (left, textual) and F_2 (right, categorical) from $\mathcal{S}$ and $\mathcal{D}$ chosen as above. Note the distributional asymmetry, due to varying distance definitions between the two feature types, as well as inherent feature variety (e.g. many "dissimilar" events may still have the same birth year); specifically, F_2 seems to cluster the chosen similar events together better than F_1. 8b (top) shows that by ignoring this intuition and uniformly weighting the features, pairwise distances within $\mathcal{S}$ and $\mathcal{D}$ overlap notably, leading to potential false positives. However, 8b (bottom) shows by learning the appropriate weights, we can much more easily discriminate the desired events. The result is shown in 8c, in which we vary $\eta \in \{.5, .6, .7, .8\}$ and compare learned and uniform weighting; we observe that while precision is hardly affected, recall is drastically higher when using a learned metric.

Model consistency. Section III-B1 mentions that we need not vary the model window $\mathcal{M}$ and detector $\mathcal{D}_{\mathcal{M}}$ often if aggregate behaviors are roughly the same. Figure 9 shows this is true in practice; models over consecutive windows are highly correlated. 9 (left) shows a correlation matrix between anomaly scores of the same resources over 7 windows $\mathcal{M}_{(t_0,t_1)} \ldots \mathcal{M}_{(t_6,t_7)}$ and their respective detectors. Correlations are all $\geq .95$, indicating strong relationships; 9 (right) shows a scatterplot over $D_{\mathcal{M}_{(t_0,t_1)}}$ and $D_{\mathcal{M}_{(t_1,t_2)}}$; note that the most anomalous resources (near 0) are consistently so.

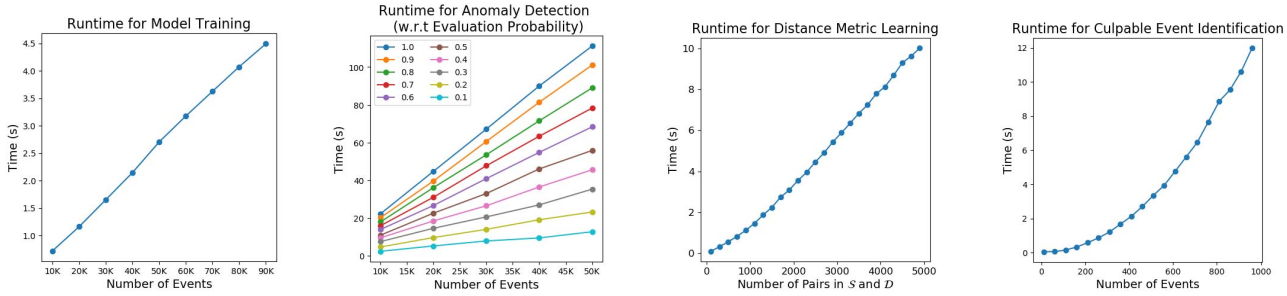
D. Scalability

Below, we discuss runtime complexity of our approach.

Model training. Figure 10a shows linear scaling for model training on SYNTH. The cost of model training is primarily driven by maintaining appropriate data structures in memory during window $\mathcal{M}$, generating the resource representations at training time, and lastly training the detector. Dropping constants, the terms are $O(|\mathcal{E}|d)$, $O(|\mathcal{E}|d)$ and $O(t)$, for $|\mathcal{E}|$ events and d features, and t isolation trees (assuming a constant samples per tree) used in Isolation Forest. The runtime is dominated by the first two steps.

Anomalous resource detection. Figure 10b shows linear scaling in anomalous resource detection on SYNTH, and demonstrates the linear reduction of resource evaluation cost w.r.t p_{eval} shown theoretically in Lemma 3. Runtime consists of maintaining data structures during window $\mathcal{C}$, generating the resource representations at a rate of p_{eval}, and evaluating the anomaly score using the eCDF of scores observed in training (scoring using $\mathcal{D}_{\mathcal{M}}$ and computing percentile). Dropping constants, the terms are $O(|\mathcal{E}|d)$, $O(|\mathcal{E}|d)$ and $O(t+k)$ assuming $|\mathcal{E}|$ events, d features, t isolation trees and k resources trained on in $\mathcal{D}_{\mathcal{M}}$. The heaviest cost is the detector evaluation.

Event-based metric learning. Figure 10c shows that this step exhibits near-linear runtime, using events from SYNTH. Time complexity is ill-defined in numerical optimization, but is



(a) Model training (b) Anomalous resource detection (c) Event metric learning (d) Culpable event identification

Fig. 10: FARE scales linearly in modeling and detecting resources w.r.t. total stream event count (a) and evaluation stochasticity (b). (c) shows the near-linear runtime induced by solving our convex event-based metric learning objective, and (d) shows quadratic runtime in the offline culpable event identification step, which is alleviated by heavy resource event-count skew and embarrassing parallelism.

associated with problem conditioning and the solver used. We used a splitting conic solver [40] which utilizes the ADMM method. In practice, the cost is small and amortizes over detection windows.

Culpable event identification. Figure 10d shows that our culpable event identification step admits cost quadratic in event count on SYNTH. This cost is $O(n^2)$, dominated by pairwise distance computation and Kosaraju's connected components algorithm, given n events (nodes). However, given that (a) most resources have few events due to heavy skew, (b) only the few resources in $\mathcal{R}_A$ are considered for this step, and (c) this task can be embarrassingly parallelized across resources and CPUs, the time cost in practice is much lower.

V. CONCLUSION

In this work, we tackle the general problem of mining anomalies corresponding to suspicious behaviors in social event logs with multiple feature types. Our contributions begin with a novel formulation of schema-agnostic anomaly detection (Problem 1) in event logs, and its dissection into two components: anomalous resource detection (Problem 2) and culpable event identification (Problem 3). Next, we propose FARE, a holistic, unsupervised framework for tackling these problems by combining online resource detection and offline event identification modules, capable of mitigating attacks in near-realtime, and reversing suspected actions in a later phase. Finally, we demonstrate empirical success in (a) detecting anomalous resources and events with 100% P@250 in two classes of industrial-scale Snapchat event data (account login and registrations), where we detect as many as 50% of anomalies associated with abusive behaviors that were uncaught by state-of-the-art, tailored platform defenses, (b) showed robustness across various synthetic attack modes and detector settings and (c) conducted runtime analysis which shows that FARE scales suitably to large datasets.

REFERENCES

[1] S. Kumar and N. Shah, "False information on web and social media: A survey," *arXiv preprint 1804.08559*, 2018.

[2] G. Ho, A. S. M. Javed, V. Paxson, and D. Wagner, "Detecting credential spearphishing attacks in enterprise settings," in *USENIX Security*, 2017.

[3] D. H. P. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining and inference for malware detection," in *SDM*. SIAM, 2011, pp. 131–142.

[4] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Comm. Surv. & Tut.*, vol. 18, no. 2, pp. 1153–1176, 2016.

[5] N. Shah, "Flock: Combating astroturfing on livestreaming platforms," in *WWW*. IW3C2, 2017, pp. 1083–1091.

[6] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *CCS*. ACM, 2014.

[7] C. Xiao, D. M. Freeman, and T. Hwa, "Detecting clusters of fake accounts in online social networks," in *WAIS*. ACM, 2015, pp. 91–101.

[8] K. Thomas, F. Li, C. Grier, and V. Paxson, "Consequences of connectivity: Characterizing account hijacking on twitter," in *CCS*. ACM, 2014, pp. 489–500.

[9] W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *TKDE*, 2004.

[10] F. Bezerra, J. Wainer, and W. M. van der Aalst, "Anomaly detection using process mining," in *BPMDS*. Springer, 2009.

[11] L. Mariani and F. Pastore, "Automated identification of failure causes in system logs," in *ISSRE*. IEEE, 2008, pp. 117–126.

[12] E. Chuah, A. Jhumka, S. Narasimhamurthy, J. Hammond, J. C. Browne, and B. Barth, "Linking resource usage anomalies with system failures from cluster log data," in *SRDS*. IEEE, 2013, pp. 111–120.

[13] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *SOSP*. ACM, 2009, pp. 117–132.

[14] A. Gainaru, F. Cappello, S. Trausan-Matu, and B. Kramer, "Event log mining tool for large scale HPC systems," in *EuroPar*. Springer, 2011.

[15] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in *KDD*. ACM, 2009, pp. 1255–1264.

[16] R. Vaarandi, "A breadth-first algorithm for mining frequent patterns from event logs," in *ICICS*. Springer, 2004, pp. 293–308.

[17] B. Hooi, N. Shah, A. Beutel, S. Günnemann, L. Akoglu, M. Kumar, D. Makhija, and C. Faloutsos, "Birdnest: Bayesian inference for ratings-fraud detection," in *SDM*. SIAM, 2016, pp. 495–503.

[18] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Gunnemann, D. Makhija, M. Kumar, and C. Faloutsos, "Edgecentric: Anomaly detection in edge-attributed networks," in *ICDMW*. IEEE, 2016, pp. 327–334.

[19] D. M. Freeman, "Using naive bayes to detect spammy names in social networks," in *WAIS*. ACM, 2013, pp. 3–12.

[20] S. Kumar, F. Spezzano, and V. Subrahmanian, "Vews: A wikipedia vandal early warning system," in *KDD*. ACM, 2015, pp. 607–616.

[21] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *KDD*. ACM, 2015, pp. 985–994.

[22] G. B. Guacho, S. Abdali, N. Shah, and E. Papalexakis, "Semi-supervised content-based detection of misinformation via tensor embeddings," in *ASONAM*, 2018.
[23] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson, "Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse." in *USENIX Security*, 2013, pp. 195–210.
[24] N. Shah, H. Lamba, A. Beutel, and C. Faloutsos, "The many faces of link fraud," in *ICDM*. IEEE, 2017, pp. 1069–1074.
[25] M. Giatsoglou, D. Chatzakou, N. Shah, A. Beutel, C. Faloutsos, and A. Vakali, "Nd-sync: Detecting synchronized fraud activities," in *PAKDD*. Springer, 2015.
[26] M. Giatsoglou, D. Chatzakou, N. Shah, C. Faloutsos, and A. Vakali, "Retweeting activity on twitter: Signs of deception," in *PAKDD*. Springer, 2015, pp. 122–134.
[27] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *KDD*. ACM, 2016.
[28] K. Shin, B. Hooi, and C. Faloutsos, "M-zoom: Fast dense-block detection in tensors with quality guarantees," in *ECML-PKDD*. Springer, 2016, pp. 264–280.
[29] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *WWW*. ACM, 2013, pp. 119–130.
[30] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos, "Timecrunch: Interpretable dynamic graph summarization," in *KDD*. ACM, 2015.
[31] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, "Spotting suspicious behaviors in multimodal data: A general metric and algorithms," *TKDE*, vol. 28, no. 8, pp. 2187–2200, 2016.
[32] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.
[33] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *SIGCOMM CCR*. ACM, 1999.
[34] C. Buragohain and S. Suri, "Quantiles on streams," in *EDS*. Springer, 2009.
[35] G. Cormode, T. Johnson, F. Korn, S. Muthukrishnan, O. Spatscheck, and D. Srivastava, "Holistic udafs at streaming speeds," in *SIGMOD*. ACM, 2004, pp. 35–46.
[36] J. S. Vitter, "Random sampling with a reservoir," *TOMS*, vol. 11, no. 1, 1985.
[37] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *J. of Algorithms*, 2005.
[38] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*. IEEE, 2008, pp. 413–422.
[39] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *NIPS*, 2003.
[40] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *J. Optim. Theory Appl.*, pp. 1042–1068, June 2016.

VI. Reproducibility

In this section, we discuss aspects of reproducibility which cover (a) proofs for mathematical claims in the paper, and (b) details on experimental settings and implementation choices.

A. Proofs

Below, we include proofs for the lemmas from Sections III-B1 and III-B2. We reproduce the lemmas here for the reader's convenience.

Lemma 3.1 (Categorical Uniques)**.** *Given a size-n sample from a feature with categorical distribution P over j categories, the expected number of unique elements $\mathbb{E}[u] = \sum_{i=1}^{j} 1-(1-p_i)^n$, where p_i denotes the probability mass on the i^{th} category of P.*

Proof of Lemma 1 (Categorical Uniques). Let I_i denote whether an element of the i^{th} category appears in the size-n sample. The probability that it does *not* appear is $(1-p_i)^n$, meaning that we can write $\mathbb{E}[I_i] = 1-(1-p_i)^n$. Since $\mathbb{E}[u] = \mathbb{E}[\sum_{i=1}^{j} I_i]$, by the linearity of expectation we have $\mathbb{E}[u] = \sum_{i=1}^{j} \mathbb{E}[I_i] = \sum_{i=1}^{j} 1-(1-p_i)^n$. □

Lemma 3.2 (Skipped Evaluations)**.** *Given an evaluation probability p_{eval}, the probability of waiting "too long" ($> s$ events beyond the expected) before an evaluation is*

$$P(X - \mathbb{E}[X] > s) = (1-p_{eval})^{s+p_{eval}^{-1}}$$

Proof of Lemma 2 (Skipped Evaluations). Observe that X is a random variable, distributed as $X \sim Geom(p_{eval})$, which has $E[X] = p_{eval}^{-1}$. We can write the CDF as $F_X(k) = 1-(1-p_{eval})^k$. Then, the probability of waiting $\leq \mathbb{E}[X] + s$ events is $F_X(\mathbb{E}[X]+s)$, making the complement $1 - F_X(\mathbb{E}[X]+s) = P(X > \mathbb{E}[X]+s) = P(X - \mathbb{E}[X] > s) = (1-p_{eval})^{s+p_{eval}^{-1}}$. □

Lemma 3.3 (Event Evaluation Ratio)**.** *Given n events across m resources $r_1 \dots r_m$ such that event count $|\mathcal{E}(r_i)|$ is distributed as C, an event count evaluation threshold τ and probability p_{eval}, the expected ratio of evaluated to total events is*

$$\mathbb{E}[n_{eval}] = n^{-1} \cdot m \cdot p_{eval} \cdot \sum_{k=\tau}^{K} [p_k \cdot (k - \tau + 1)]$$

where $K = max(dom(C))$ and $p_k = P(|\mathcal{E}(r_i)| = k)$.

Proof of Lemma 3 (Event Evaluation Ratio). We can write the expected number of k-event resources as $m \cdot P(|\mathcal{E}(r_i)| = k)$. Since only the last $k - \tau + 1$ elements per k-event resource are considered for evaluation, we can write the expected considered event-count from all k-event resources as $m \cdot P(|\mathcal{E}(r_i)| = k) \cdot (k - \tau + 1)$. Thus, across all $\geq k$-event resources, we have an expected considered event-count of $m \cdot \sum_{k=\tau}^{K} P(|\mathcal{E}(r_i)| = k) \cdot (k - \tau + 1)$. Since only p_{eval} of these events are evaluated in expectation out of n total events, we have $\mathbb{E}[m_{eval}] = n^{-1} \cdot m \cdot p_{eval} \cdot \sum_{k=\tau}^{K} [P(|\mathcal{E}(r_i)| = \tau) \cdot (k - \tau + 1)]$. □

Lemma 3.4 (Event-based Distance Metric)**.** *If $\mathcal{M}_1 \dots \mathcal{M}_d$ define distance metrics on feature spaces $F_1 \dots F_d$ respectively, then for any two events $e_i = (a_1 \dots a_d)$ and $e_j = (b_1 \dots b_d)$ where $a_k, b_k \in F_k$, $\mathbf{W} = diag(w_1 \dots w_d) \succeq 0$ and $\vec{h} = \left[\sqrt{\mathcal{M}(a_1, b_1)} \dots \sqrt{\mathcal{M}(a_d, b_d)}\right]$*

$$\Delta_{\mathbf{W}}(e_i, e_j) = \vec{h}^T \mathbf{W} \vec{h}$$

defines a distance metric on the event-space $F_1 \times F_2 \dots \times F_d$.

Proof of Lemma 4 (Event-based Distance Metric). It is well-known that if $\mathcal{M}_1 \dots \mathcal{M}_d$ individually define metrics on spaces $F_1 \dots F_d$, then the product-space $F_1 \times F_2 \dots \times F_d$ is also metrizable with $\sum_i^d \mathcal{M}_i$ defining a metric. Moreover, for any scalar $w > 0$ and metric $\mathcal{M}$, $w\mathcal{M}$ is also a metric. Thus, $\sum_i^d w_i \mathcal{M}_i$ is a metric for $w_1 \dots w_d > 0$. Noting that $\sqrt{\mathcal{M}}$ is also a metric for any metric $\mathcal{M}$, we have that $\vec{h}^T \mathbf{W} \vec{h} = \sum_i^d w_i \mathcal{M}_i$, hence defining a metric. □

B. Experimental settings

1) Parameter values: For all synthetic experiments, we use the following "fixed" parameter values, except when varying a particular parameter to gauge sensitivity.

- We fix $p_{eval} = 0.1$ (evaluation stochasticity), re-running anomaly scoring on final resource representations after processing the generated stream to compute anomaly scores for resources which had not been scored before due to stochasticity. In cases where resources were scored numerous times, we used the maximum anomaly score to rank the resources for measuring detector performance.
- We fix *atk_scale* $= 1.0$ (proportion of anomalous resource attack events), such that all designated, synthetic anomalous resources had only anomalous events, removing the possibility for variation in that some resources become more anomalous than others.
- We fix *atk_hetero* = 1 (number of varying attacker features), to reflect that most attackers are somewhat naïve and only manipulate a single feature while leaving other features constant (i.e. changing their target username, but leaving contact details the same for registration events).
- We fix η (cross-event similarity threshold) to the median of pairwise similarity scores in $\mathcal{S}$, under the assumption that most pairs in $\mathcal{S}$ will have considerably high similarity given the learned metric.

We reiterate that these are simply the standard parameter choices we considered as fixed, such that we could vary each parameter individually and study its own sensitivity on detection performance (reflected in Figures 6 and 7) rather than the combinatorial number of options.

As for our metric learning objective, we specified $\gamma = 1.0$, and $\beta = 1.0$, as we found they were empirically suitable for producing inferred $\mathbf{W}$ which was not rank-deficient, and also not trivial/uniform. We found that increasing γ (dissimilarity penalty) did not greatly influence the learned weights, but increasing β too much (weight regularization) led to uniform weighting (as expected) as the regularization term dominated the objective.

2) Implementation details: We used Python and Jupyter Notebooks to run all experiments, on a high-memory single-node Google Cloud compute engine instance. We conducted the initial simulation experiments on summary statistic selection using `scipy`'s built-in distributions and variate generation functions. In evaluating maintenance and computation of summary statistics, we simulated streaming `LOGIN` and `REGN` data by iterating over Pandas dataframes extracted from structured production-level event logs taken over several days, upon which we wrote custom classes for reservoir sampling and numerical statistic maintenance, and used `numpy`'s `percentile` function which employs an $O(k)$ selection algorithm for quantile computation. We used the Isolation Forest implementation available in `scikit-learn` for unsupervised resource anomaly detection, and `statsmodels` ECDF functionality for detection score normalization into $(0, 1)$. We utilized `cvxpy`, a Python interface into CVX to solve the proposed metric objective; we found the splitting conic solver (SCS) to be the most time-efficient for our setting, with comparable minimization performance to more exact solvers. Finally, to extract suspicious event clusters, we utilized the `networkx` package for inter-event similarity graph construction and connected components extraction.