A User and Entity Behavior Analytics Log Data Set for Anomaly Detection in Cloud Computing

Max Landauer, Florian Skopik, Georg Höld, Markus Wurzenberger
Digital Safety & Security
Austrian Institute of Technology
Vienna, Austria
firstname.lastname@ait.ac.at

***Abstract*—Cyber criminals utilize compromised user accounts to gain access into otherwise protected systems without the need for technical exploits. User and Entity Behavior Analytics (UEBA) leverages anomaly detection techniques to recognize such intrusions by comparing user behavior patterns against profiles derived from historical log data. Unfortunately, hardly any real log data sets suitable for UEBA are publicly available, which prevents objective comparison and reproducibility of approaches. Synthetic data sets are only able to alleviate this problem to some extent, because simulations are unable to adequately induce the dynamic and unstable nature of real user behavior in generated log data. We therefore present a real system log data set from a cloud computing platform involving more than 5000 users and spanning over more than five years. To evaluate our data set for the scenario of account hijacking, we outline a method for attack injection and subsequently disclose the resulting manifestations with an adaptive anomaly detection mechanism.**

***Index Terms*—log data, anomaly detection, user and entity behavior analytics**

I. Introduction

Humans are often considered the weakest link in cyber security. This claim is corroborated by recent studies suggesting that around 82% of data breaches involve a human factor, for example, through stolen credentials and subsequently compromised accounts [1]. Today's omnipresence of cloud computing exacerbates this problematic situation as both external adversaries as well as insider attackers have easy access to a large amount of possibly confidential information [2], [3]. Unfortunately, commonly applied signature-based detection approaches that search for known indicators of compromise are often unable to recognize this kind of intrusion as no technical vulnerabilities are exploited and only legitimate actions are executed, such as viewing or modifying files.

As a consequence, security researchers and analysts resort to User and Entity Behavior Analytics (UEBA), which aims to disclose malicious users that exhibit inherently different behavior patterns than legitimate system users and can thus be detected even when the exact attack scenario is not known beforehand [4]. Thereby, UEBA usually relies on the assumption that the majority of users consistently follows similar routines over an extended period of time so that it is possible to derive behavior profiles for their regular activities. Intrusion detection approaches compare these baselines with the continuously monitored user behavior and report anomalies when users diverge from their usual behavior patterns to inform operators about potential misuse of the respective accounts [5].

Log data permanently preserves a chronological list of almost all activities carried out by users on monitored systems and is thus one of the main sources for generating and evaluating user behavior profiles [6]. Thereby, commonly available parameters in log events include timestamps, types of action (e.g., opening a file), user identifiers (e.g., mail addresses), and even contextual information such as readings from sensors within devices (e.g., global positioning system) [3]. Given that these are highly personal data, it is not surprising that organizations are reluctant to make any real log data sets publicly accessible to avoid privacy violations. Accordingly, evaluations are frequently carried out on private data sets that prevent reproducibility of the presented results.

To overcome this obstacle, researchers therefore resort to synthetic log data generated by simulations in testbeds. Synthetic data generation has some benefits over real infrastructures when it comes to scientific evaluations, including a more flexible technical environment, the possibility to launch arbitrary attacks, and a reliable ground truth as there are no unknown activities [7]. However, we argue that comprehensive evaluations should additionally consider real data as a means of validation, because simulations may not sufficiently represent the intricacy and diversity of real system behavior. In particular, simulated normal user behavior is often modeled in a simplified way using scripts that automatically carry out similar actions indefinitely, thus false alarm rates achieved on synthetic data may be disproportionately low considering the more erratic user behavior that is sometimes encountered in real systems [8]. In addition, real log data is affected by long-term changes such as modifications of system infrastructures and logging configurations that are neglected or not relevant when generating log data in short-term simulations; however, these sources of data instability need to be taken into account when designing anomaly detection systems. In the past, authors manually modified existing log data sets to adequately reflect such dynamic changes [9]. We summarize the benefits and issues with synthetic and real log data in Table I.

We present CLUE-LDS (CLoud-based User Entity behavior analytics Log Data Set) that is published alongside with

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

TABLE I: Differences between synthetic and real log data sets

Aspect	Synthetic log data	Real log data
Generation	(−) Time-consuming	(+) Low effort
Normal/benign user behavior	(−) Limited by scope of simulation	(+) Realistic
False alarm rate	(−) Possibly too low	(+) Accurate
Attack Injection	(+) Execution of any attacks possible	(−) Restricted in productive systems
Ground truth/ Labels	(+) Reliable and (semi-)automatic	(−) Unreliable and manual/unavailable
Publication of data set	(+) Simple as no sensitive data involved	(−) Needs authorization and anonymization
Flexibility	(+) Easy to configure and extend testbeds	(−) Limited ways to change infrastructure

this paper. The data set spans over five years and contains legitimate log events generated by more than 5000 distinct user entities. We overcome the issues arising from real log data sets (cf. Table I) by anonymizing sensitive data without loss of information and proposing a strategy to inject and label anomalies representing compromised accounts in the data set. We also present an illustrative evaluation of an anomaly detection approach that is capable of handling the dynamic nature of the data. We summarize our contributions as follows:

- A new open log data set for UEBA[1],
- an analytical investigation of the presented data set,
- a method to inject anomalies into the data set, and
- a benchmark evaluation for dynamic anomaly detection[2].

The remainder of the paper is structured as follows. Section II reviews existing log data sets. Section III describes how the data set was collected and explains the overall structure of the data. We provide an analysis of user behavior patterns in Sect. IV. In Sect. V we explain our approaches to inject and detect anomalies in the data. We discuss our main findings in Sect. VI. Finally, Sect. VII concludes the paper.

II. Related Work

There is a great need for publicly available log data sets in the research community centered around anomaly detection and UEBA for cyber security applications, however, only few data sets are publicly available [8], [10]. For example, Kholidy et al. [11] describe the generation of a data set for intrusion detection in cloud computing, but unfortunately do not make their data accessible. As a consequence, anomaly detection approaches such as the one by Ganfure et al. [12], which uses count vectors on user-specific system operations (e.g., creating, modifying, or deleting files) to differentiate malware behavior from otherwise normal user activities, frequently rely on private data sets that prevent reproducibility. Others utilize outdated web log data sets that may not be representative for modern system environments anymore [13].

To overcome these problems, researchers recently focused on synthetically generating log data sets in test environments where arbitrary attacks can be launched. For example, Uetz et al. [10] use automated scripts to simulate normal user web browsing and collect log data from services and operating systems of client computers. Thereby, they model relevant parameters and distributions after real observations. Similarly, Landauer et al. [7], [8] use state-machines to simulate normal user behavior and compare the generated system log data with logs from real infrastructures. To generate the ADFA log data set, Creech et al. [14] launch attacks during normal system utilization, including browsing and document preparation, and collect the generated system call traces. Sharafaldin et al. use abstracted profiles from real user behavior to generate and publish multiple data sets for network traffic analysis, including CICDDoS2019 [15] and CICIDS2017 [16]. These synthetic log data sets are usually sufficient when analysis focuses on attacks and normal user behavior only serves as background noise. However, the data sets are only of limited use for UEBA, which relies much stronger on the presence of realistic user behavior patterns in the logs.

Some of the most widely used data sets for log-based anomaly detection are the HDFS [17], BlueGene/L (BGL) [18], Thunderbird [18], OpenStack [19], Hadoop [20], and the data sets from Loghub [21]. Other than most synthetic data sets that usually only span over a few days, they are collected in lab settings and sometimes have significantly longer time spans of around 200 days, thus facilitating analysis of long-term system behavior. However, the problem with these data sets is that they do not involve user identifiers, but instead focus on files or processes. Accordingly, they have little value for UEBA. To overcome this gap, we outline the collection of an UEBA data set in the following section.

III. Log Data Generation & Structure

This section explains collection and processing of the log data set, and describes the structure and contents of the events.

A. Collection

The data set was generated at the premises of Huemer Group, a midsize IT service provider located in Vienna, Austria. Huemer Group offers Infrastructure-as-a-Service solutions for enterprises, including cloud computing and storage. In particular, their cloud storage solution called hBOX[3] enables customers to upload their data, synchronize them with multiple devices, share files with others, create versions and backups of their documents, collaborate with team members in shared data spaces, and query the stored documents. The hBOX extends the open-source project Nextcloud[4] with interfaces and functionalities tailored to the requirements of customers.

The log data set was collected by technical staff responsible for maintaining the hBOX. At the time of collection, more than five years of log data produced by the hBOX hosted within the organization were available - precisely, a total of 50,522,931 log events that span from 2017-07-07 to 2022-09-29 (1910 days). The logs were not altered or filtered in any way during collection; however, the data had to be anonymized for publication as described in the following section.

[1]The log data set is available at https://zenodo.org/record/7119953

[2]Scripts for injecting and detecting anomalies are available as open-source code at https://github.com/ait-aecid/clue-lds

[3]Huemer Group website, https://www.huemer-group.com/hbox/

[4]Nextcloud website, https://nextcloud.com

B. Anonymization

The original log data set contains sensitive information, such as names of users and files, that prevent publishing for data privacy reasons. These attributes were therefore anonymized by replacing them with combinations of words randomly selected from dictionaries. Thereby, we ensured that the replacement of values is consistent across all attributes so that the same pseudonyms are used for identical values no matter where they occur. This ensures that no information is lost when relating activities to specific users, which is an often criticized factor in data sets that suffer from heavy anonymization [22].

We split paths at slashes before anonymizing their parts individually so that directory hierarchies are preserved. For example, the paths "/home/user/example" and "/home/user/-folder/example" are anonymized as "/A/B/C" and "/A/B/D/C" respectively. Note that some attributes in the data set were already anonymized as they use an internal numeric identifier rather than actual names. These attributes were left unchanged. Some events also involve the geolocation of the user based on the IP address associated with the event[5]. Since location accuracy is too low to identify individual users, the corresponding attributes were also left in the data without modifications. The following section shows a complete list of all event attributes.

C. Event Attributes

The log data are a chronological sequence of semi-structured events in JSON format. Table II shows all attributes that occur on the first level of the JSON objects as well as a sample value for each attribute and a brief description. The main attributes relevant for forensic analysis of user behavior are "type", "time", and "uid", as they describe what action was carried out, when it took place, and who is responsible for it. Attribute "params" is a nested object that contains many additional attributes that are specific for each event type, which we describe in more detail in the following section.

Some of these attributes are optional and do not necessarily occur in each event. In particular, this concerns the geolocation information which is only available for around 0.25% of all events. Figure 1 shows geolocations present in the data set and indicates their respective occurrence frequencies by colors. Given that Huemer Group and many of their partners are located in Austria it is not surprising that a majority of the users access the hBOX from cities in Austria and specifically its capital Vienna. Note that the plot shows central Europe as this is the origin of most accesses and that there are also other geolocations present in the data set not visible in the plot.

D. User actions

As outlined in the previous section, each log event involves a "type" that corresponds to a specific action carried out by a user. Table III provides an overview of all 49 unique event types that we group into (i) *Administration* for events such as changing settings that are usually carried out by privileged users, (ii) *Comments* for adding and deleting comments, (iii) *Files* for all events that handle files, (iv) *Groups* for creating or changing user groups, (v) *Authentication* for logging in and out, (vi) *Sharing* for events where links are shared between users, and (vii) *Users* for creating or changing user accounts.

[5]MaxMind was used for IP localization, https://dev.maxmind.com/geoip

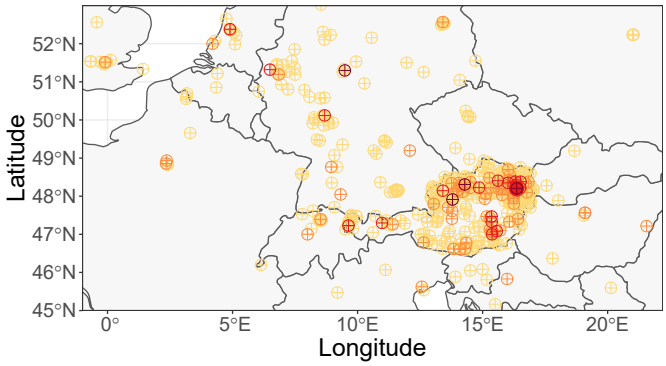


Fig. 1: Geolocations of users generating events. Colors indicate event frequencies (yellow: <50, orange: 50-499, red: 500-4999, dark red: >5000).

The table also shows the attributes of the nested "params" object (cf. Table II) that are specific for each event type. We mark optional attributes with an asterisk. Moreover, we use italics to indicate attributes that are anonymized as described in Sect. III-B. Most of the non-anonymized attributes are numeric representations of categories, for example, "itemSource" or "permissions". Table III furthermore provides examples for attributes that contain values with semantic meaning in brackets next to the parameters. Note that there are also a few special attributes such as "date", which contains a formatted date string, or "deletedShares", which is a nested object comprising several more attributes similar to those of the *Sharing* events.

IV. Data Analysis

In the previous section we stated general properties of the log data set. This section explores user behavior patterns that are relevant for profiling and anomaly detection.

A. Static Analysis

Each event in the log data set can be attributed to a specific user through the attribute "uid". Accordingly, it is simple to derive usage statistics for single users and thus form profiles. For example, users may be differentiated between employees who use many functions of the hBOX over a long time span and thus produce higher numbers and more diverse log events, and external users who are just accessing a shared file once. Note that it is also possible to correlate user names with IP addresses as both of these anonymized values are present in events of type "login_attempt"; however, we consider distinct users only by their "uid" in the following for simplicity.

Figure 2 shows aforementioned characteristics measured by the number of unique event types, total number of events, number of days where at least one event is generated, and "role" attribute, where each point represents a single user. The plot shows that most users generate less than 1000 events with less than 10 distinct event types, while comparatively few users

TABLE II: Attributes of the semi-structured log events

Attribute	Example	Description
id	1	Unique log line identifier that starts at 1 and increases incrementally.
time	2021-01-01T00:00:02Z	Time stamp of the event in ISO format.
uid	old-pink-crane-sharedealer	Unique anonymized identifier for the user generating the event.
uidType	name	Specifier for uid, which is either the user name or IP address for logged out users.
type	file_accessed	The type of the event. Table III lists all event types present in the data set.
params	Diverse parameters	Dictionary containing event parameters. See Table III for a detailed enumeration.
isLocalIP	true	Optional flag for event origin, which is either internal (true) or external (false).
role	technical	Optional user role: consulting, administration, management, sales, technical, or external.
location.city	Vienna	Name of the city where the request originates from.
location.countryName	Austria	Name of the country where the city is located.
location.countryCode	AT	Abbreviated name of the country.
location.region	9	Regional code of the location.
location.regionName	Burgenland	Name of the region (e.g., state or province) of the location.
location.isInEuropeanUnion	true	Specifies whether the location lies in the European Union (true) or not (false).
location.continent	Europe	Name of the continent where the country is located.
location.postalCode	1040	Postal code of the city or district.
location.metroCode	754	Metro code of cities within the United States.
location.latitude	48.1535	Latitude of the location information.
location.longitude	16.3855	Longitude of the location information.
location.accuracyRadius	50	Accuracy of the location estimation in kilometers.
location.timezone	Europe/Vienna	Name of the time zone of the location.

TABLE III: Overview of log event types and their respective parameters

Group	Event type	Parameters
Administr.	admin_settings_changed	Diverse parameters (e.g., "value": {"level": 1110} , "key": "loglevel", "app": "logger")
	app_disabled	app (e.g., "quota_warning")
	app_enabled	app
	command_executed	arguments* (e.g., "files:scan <*path*>")
	email_changed	*email**, *user*
	fullname_changed	*fullname*, *user*
	password_changed	*user*
	permission_changed	itemSource, itemType, *path*, permissions, shareType, *shareWith**, *uidOwner*
	public_share_accessed	errorCode (e.g., "404"), errorMessage* (e.g., "Wrong password"), *fileTarget**, itemSource*, itemType*, *token**, *uidOwner**
	public_share_expiration_date_changed	date*, itemSource, itemType, *uidOwner*
	public_share_password_changed	disabled (e.g., "False"), itemSource, itemType, *token*, *uidOwner*
	public_share_update_permission	itemSource, itemType, *path*, permissions, shareType, *shareWith**, *uidOwner*
	version_deleted	*path*, trigger
	version_restored	node*, *path*, revision
	quota_changed	quota (e.g., "1 GB"), *user*
Comments	comment_added	id, *message*, object, *path*
	comment_deleted	id, *message*, object, *path*
Files	file_accessed	*path*
	file_copied	*newpath*, *oldpath*
	file_created	*path**, run*
	file_deleted	*path*, run
	file_renamed	*newpath*, *oldpath*
	file_updated	*path*, run*
	file_written	*path**, run*
	deleted_from_trashbin	*path*
	restored_from_trashbin	*filePath*, *trashPath*
Groups	group_created	*group*
	group_deleted	*group*
	user_became_groupadmin	*group*, *user*
	groupadmin_removed_from_group	*group*, *user*
Auth.	login_attempt	*loginName**, successful* (e.g., "False"), *user**
	login_failed	*ip*, *user**
	login_successful	*user*
	logout_occurred	-
Sharing	shared_email	fileSource, *fileTarget*, id, itemSource, *itemTarget*, itemType, permissions, shareType, *shareWith*, *token*, *uidOwner*
	shared_group	fileSource, *fileTarget*, id, itemSource, *itemTarget*, itemType, permissions, shareType, *shareWith*, *uidOwner*
	shared_link	expiration*, fileSource, *fileTarget*, id, itemSource, *itemTarget*, itemType, permissions, shareType, *token*, *uidOwner*
	shared_user	fileSource, *fileTarget*, id, itemSource, *itemTarget*, itemType, permissions, shareType, *shareWith*, *uidOwner*
	unshared_email	deletedShares, fileSource, id, itemSource, itemType, shareType, *uidOwner*
	unshared_group	deletedShares*, fileSource, *fileTarget*, id, itemSource, *itemTarget**, itemType, itemparent*, shareType, *shareWith*, *uidOwner*, unsharedItems*
	unshared_link	deletedShares, fileSource, *fileTarget*, id, itemParent*, itemSource, itemType, shareType, *uidOwner*
	unshared_user	deletedShares*, fileSource, *fileTarget*, id, itemSource, *itemTarget**, itemType, shareType, *shareWith*, *uidOwner*, unsharedItems*
Users	user_added_to_group	*group*, *user*
	user_assign	*user*
	user_created	*user*
	user_data_viewed	-
	user_deleted	*user*
	user_disabled	*user*
	user_removed_from_group	*group*, *user*

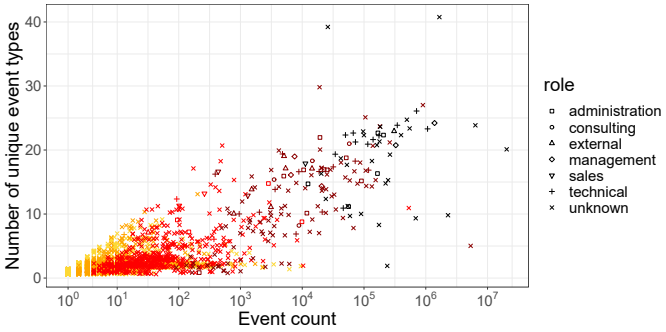


Fig. 2: Users behavior profiles displayed by their number of unique events (vertical axis), total number of events (horizontal axis), role (symbol), and number of active days (yellow: 1, orange: 2-4, red: 5-49, dark red: 50-499, black: >500).

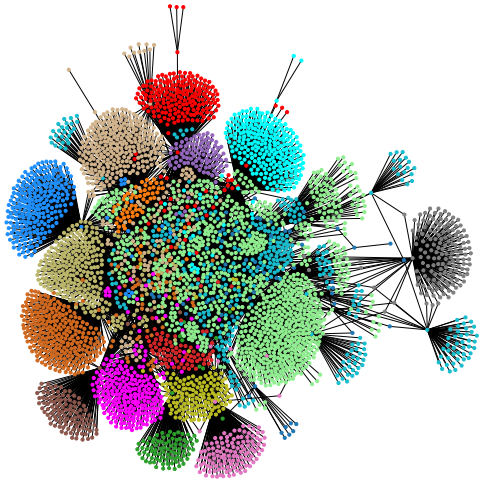


Fig. 3: User interaction graph showing groups of related users.

account for a much higher number of events and make use of more event types. Not surprisingly, the latter group of users is active for a longer amount of time while users who carry out less events usually do so within few days, possibly a single day. Overall, out of the 5389 total users, 3907 are only active on a single day, 768 on 2-4 days, 539 on 5-49 days, 128 on 50-499 days, and 47 on 500 or more days, where counted days of user activity are not necessarily consecutive.

As the hBOX is a collaborative tool, it is reasonable to analyze interactions and collective behavior patterns in addition to individual user profiles. We therefore use the attributes "user" (e.g., in group *Users*) and "uidOwner" (e.g., in group *Sharing*) to form connections between related users. Figure 3 shows the resulting graph, where each node represents one user and an edge between two users indicates that there exists an event generated by one of the users that references the other user. Note that we omit edges that directly connect users to themselves and also exclude pairs of users that only connect to each other. As visible in the figure, several clusters of varying sizes emerge where all users within a cluster share connections to the same user, which we refer to as hub. To emphasize these clusters we plot hubs with more than 100 adjacent nodes with a specific color and apply the color of the largest connected hub on all other nodes. While the majority of the nodes is located in one of these clusters, the nodes in the center of the graph have a much higher number of adjacent nodes, indicating that these users interact with many distinct users.

B. Dynamic Analysis

The previous section presented static properties of the data set without considering timestamps of events. However, user behavior, in particular when observed over a long time, constitutes a highly dynamic process that usually exhibits long-term shifts as well as seasonal patterns that need to be taken into account when applying anomaly detection. To obtain an overview of the stability of event occurrences, we first generate a count matrix for all event type occurrences in each month. Figure 4 visualizes the resulting time-series of the 14 most common event types. The plot shows that even though different event types appear with frequencies across several orders of magnitude, there is indeed some continuity over time, for example, the "login_attempt" event type occurs between 10^4 and 10^5 times in almost all months, except for March 2021 when it suddenly peaks to 10^7. Another interesting observation is that some event types correlate with other, for example, events "login_attempt" and "login_successful" have a strong correlation due to the fact that most login attempts are successful. Moreover, the appearances of some event types depend on changes of the system logging configuration. For example, "login_failed" events are not generated until the beginning of 2019 and suddenly stop to appear after mid 2021.

To identify reasons for aforementioned variabilities of event occurrences, we investigate event counts independent of their type with higher granularity. Figure 5a shows event counts in intervals of 2 minutes for every day in the data set. Several interesting patterns appear in this plot. First, vertical lines of varying width depict phases of high activity that usually takes place for some hours, days, or even weeks. Many of these events are generated by client synchronization tools that query the system in short time intervals (e.g., multiple login attempts occur in each second). Second, horizontal lines originating from scheduled programs generate events at specific times of a day (e.g., every full hour) over several weeks or months. This includes tools for version management, file scanning, and recycling. Third, normal user activities that are generally more frequent during daytime (i.e., 7:00-17:00) form a region with increased event density in the vertical center of the plot.

The overall system utilization is obviously affected by a large number of independently acting users and thus the aggregated system behavior frequently undergoes changes. However, it turns out that the behavior of many users is also highly unstable, consisting of a mix of regular and irregular patterns, and apparently changing their behavior profiles several times. We arbitrarily picked a user representing this tendency and visualized their activities in Fig. 5b, where each point represents one event generated by that user. The artifacts are similar to the aggregated plot: the figure clearly shows horizontal lines around January 2018, vertical lines starting

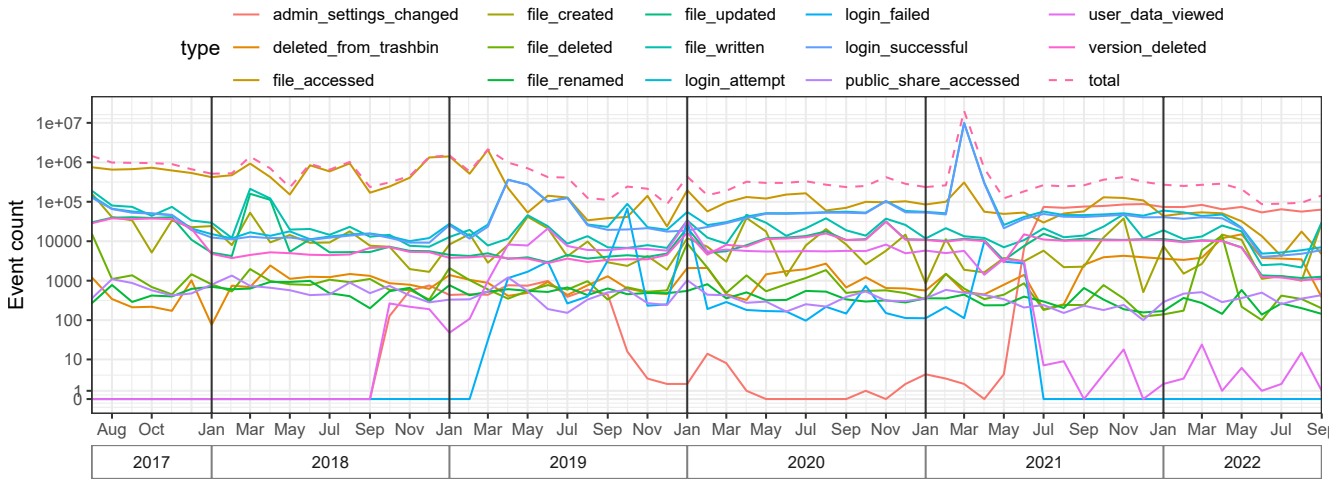
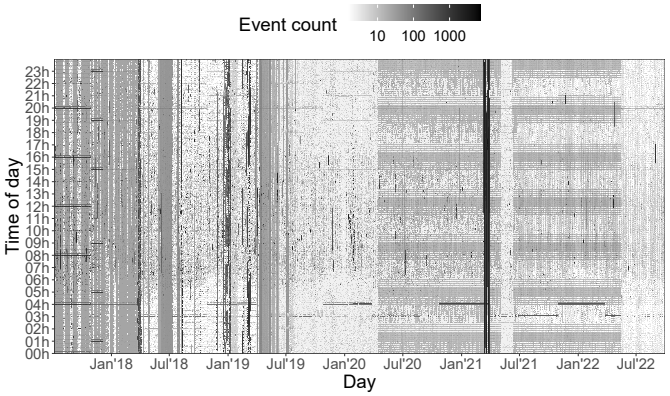
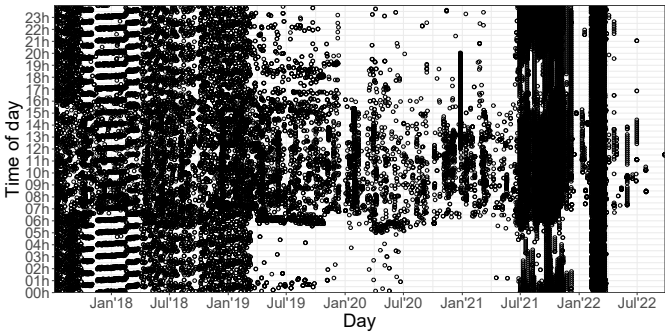


Fig. 4: Event occurrence counts per month for 14 most common event types and the sum of all event types (dashed line).



(a) All users.



(b) User "spectacular-copper-cheetah-postman".

Fig. 5: Daily event occurrences plotted over time.

from July 2021, and normal user behavior during day times.

These instabilities of normal user behavior have a strong impact on anomaly detection applied on the data set. The reason for this is that anomaly detection systems are usually designed to recognize such rapid changes of user behavior and thus many of the reported anomalies have to be regarded as false positives. To further investigate the extent of this problem, we measure how long it takes until a user executes a new event type that they have not carried out before. The bottom part of Fig. 6 shows these anomalies as red circles, where the vertical axis lists all users sorted by their total active duration (i.e., the duration between the first and last event recorded for that user), the horizontal axis shows the days after which the anomalies were detected, and the blue triangles indicate the last event recorded by the user. Note that we only display users that are active for at least five days. As expected, most anomalies accumulate on the left side of the plot; however, the plot also reveals that users execute a significant number of new events even after several months of activity. In particular, we observe anomalies from 567 out of 1206 users after 10 days (47%), 360 out of 846 users after 100 days (42.6%), 162 out of 445 users after 500 days (36.4%), and 45 out of 196 users after 1000 days (23%).

The top plot of Fig. 6 shows the total number of anomalies per day aggregated over all users. Corresponding to the bottom plot, the number of anomalies is enormous with around 10,000 anomalies on the first day, but rapidly decreases to around 100 anomalies on the second day, and stabilizes at less than 10 anomalies per day after 50 days. We emphasize that these numbers depict an ideal situation where all users start at the same point in time, however, as new users appear arbitrarily these anomalies spread out evenly through the data set. Moreover, the depicted scenario neglects that a conventional anomaly detection system trained only for a limited amount of time (say, 100 days) would report repeated occurrences of event types not seen in the training phase multiple times and thus significantly increase the total number of anomalies. These findings align with previous studies that indicate limited applicability of conventional anomaly detection approaches on user-generated log data [23]. We therefore conclude that anomaly detection with fixed training sizes is not feasible for this data and that dynamic learning techniques with continuous re-training are required to compensate for long-term changes of user behavior patterns. We present and evaluate such an anomaly detection technique in the following section.

V. Illustrative Scenario

This section presents an illustrative scenario for anomaly detection. We first outline how we inject anomalies in the data set and then describe and evaluate our detection approach.

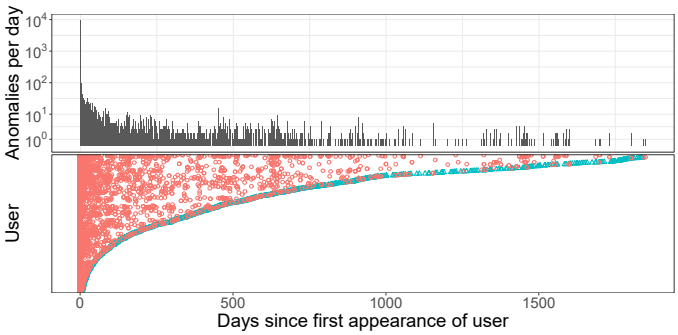


Fig. 6: Anomaly detection based on event types. Bottom: Anomalies per user. Top: Aggregated anomaly counts.

A. Attack Injection

We showed in the previous section that the user behavior patterns derived from the data set are subject to dynamic changes. Given that the data records real user behavior it is difficult to determine the exact reasons behind these artifacts and thus no comprehensive ground truth can be established for the data set. As a solution, we propose to purposefully alter the behavior patterns of some users at specific points in time and to use that information as the ground truth for evaluation of anomaly detection systems. Note that this implies that detection of normally occurring system or user behavior changes counts as false positives even though they may be valid changes from a purely data-driven perspective.

Our idea is to focus on an attack where one user account is hijacked by another, i.e., starting from one point in time the behavior patterns of a certain user does not reflect their previous behavior anymore, but instead entails new patterns that belong to some other user. We simulate this situation by randomly selecting two users u_1, u_2 and interchanging their respective identifiers (attribute "uid") when they fulfill the following requirements (Table IV summarizes all used variables). First, both users must be active for a certain number of days before switching their identifiers at time t_s so that an anomaly detection system is capable of learning their respective behavior profiles. We therefore require that $|\{t_i \in d(u) : t_i < t_s\}| > d_{min}$ for both $u1$ and $u2$, where $d(u)$ is the set of days where user u produces at least one event. Second, both users must be active for at least one day after switching their identifiers to ensure that an anomaly detection system has sufficient time to recognize the change. Third, we only consider users that produce at least c_{min} total events with at least a_{min} unique event types, i.e., we require that both u_1 and u_2 fulfill $c(u) > c_{min}$ and $|a(u)| > a_{min}$, where $c(u)$ is the event count and $a(u)$ is the set of all event types executed by user u. Fourth, the behavior patterns of the two users should not be too different so that detection becomes trivial, but also not too similar so that switching their identifiers has no apparent effect. We address this problem by proposing a similarity metric in Eq. 1 that yields high scores for users that exhibit similar system utilization rates and system activities. The first part of the equation computes the ratio between the average number of events produced per day and the second part computes the ratio of common event types carried out by both users. Furthermore, we use parameters ω_1, ω_2 with $\omega_1 + \omega_2 = 1$ to weight the terms.

TABLE IV: Symbol definitions

Symbol	Definition	Symbol	Definition
t	Specific day (time stamp).	$d(u)$	Days where u is active.
$\mathcal{U}_t$	Set of users known at t.	d_{min}	Min. req. active days.
u	Specific user, $u \in \mathcal{U}$.	t_s	Time where users are switched.
$a(u)$	Event types carried out by u.	ω_1, ω_2	Weights for sim. computation.
α	Specific event type, $\alpha \in a(u)$.	s_{min}	Min. req. similarity.
a_{min}	Min. req. unique event types.	s_{max}	Max. req. similarity.
$c(u)$	Event counts of user u.	r	Min. number of re-training days.
$c_{\alpha,t}(u)$	Counts of α at day t for u.	q	Max. size of trained model.
c_{min}	Min. req. total events.	θ	Min. sim. detection threshold.

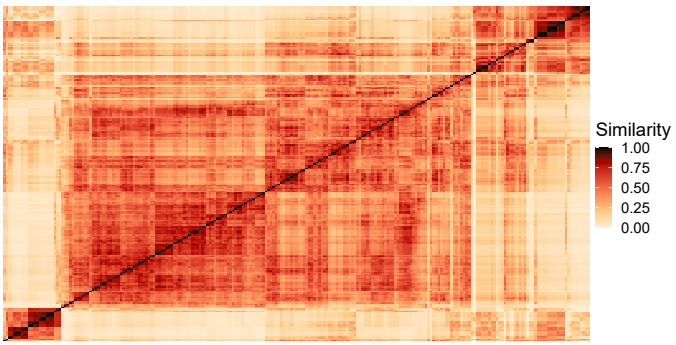


Fig. 7: Heatmap visualizing similarities of user pairs.

$$sim(u_1, u_2) = \omega_1 \cdot \frac{\min\left(\frac{c(u_1)}{|d(u_1)|}, \frac{c(u_2)}{|d(u_2)|}\right)}{\max\left(\frac{c(u_1)}{|d(u_1)|}, \frac{c(u_2)}{|d(u_2)|}\right)} + \omega_2 \cdot \frac{|a(u_1) \cap a(u_2)|}{|a(u_1) \cup a(u_2)|} \quad (1)$$

Figure 7 visualizes the similarity scores between pairs of users in a heatmap using $\omega_1 = 0.3$, and $\omega_2 = 0.7$. Note that for this plot we only select users that are active for at least 25 days but do not require that their activity periods overlap so that it is possible to switch their identifiers. The plot shows that our similarity metric allows to identify groups of users from which two users can be picked randomly so that requirement $s_{min} < sim(u_1, u_2) < s_{max}$ is fulfilled.

B. Dynamic Learning

As outlined in Sect. IV-B, the data set presented in this paper is unstable as it involves dynamically changing user behavior patterns. Accordingly, it is not simply possible to obtain training and test files that are commonly available when evaluating anomaly detection systems based on user and entity behavior analysis, e.g., by separating the data set at a specific point in time or by selecting a subset of users [3]. Instead, the nature of this data set (and most real-world data sets) necessitates the application of dynamic learning mechanisms that enable incremental and continuous adaptation of trained models to comply with newly observed patterns.

Our key observation from Sect. IV-B is that changes of user behavior are usually not manifesting themselves as slow processes that gradually affect the prevalent patterns; they occur rapidly and subsequently remain stable for some time. Our goal is therefore to detect these change points as anomalies and immediately start re-training by adapting the already

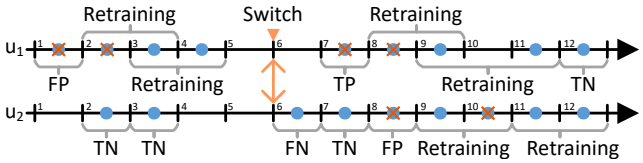


Fig. 8: Detected events and subsequent re-training phases.

trained models until they are sufficiently approximating the new behavior patterns. Thereby, we define only the minimum duration of the re-training phase r but allow the detector to dynamically extend that duration based on how many anomalies are detected. After the model stabilizes and no more anomalies occur for r days, re-training is stopped and the anomaly detection system switches back to normal operation.

For simplicity, we consider a day where one specific user generated at least one event as the unit of detection in the following, i.e., the whole day is regarded either as anomalous or benign rather than its individual events. Figure 8 shows exemplary timelines for users u_1 (top) and u_2 (bottom), where each tick marks the start of a new day, blue circles indicate that at least one event was generated on the respective day, red crosses indicate that one or more of these events were detected as anomalous, and the orange triangle indicates the attack injection time. In the figure, day 1 of u_1 is detected as anomalous and therefore triggers the re-training phase for days 2 and 3. However, day 2 is also considered as anomalous and therefore re-training is extended to day 4. As no more anomalies occur on day 3 and 4, the re-training phase ends. Note that we only count days with user activity, e.g., the re-training phase triggered by the anomaly on day 8 takes three days as there are no events produced by u_1 on day 10.

C. Anomaly Detection

To determine whether the daily activities of a user are considered anomalous or not, we propose a metric based on the frequencies and diversities of involved event types in accordance with our user similarity metric from Sect. V-A. The main idea is to compare each new observation of daily generated events with a set of previous observations that act as a baseline of normal behavior, and classify the new observation as anomalous if it is not sufficiently similar to any of the previous ones. This allows incremental processing of events as new observations are immediately added to the model after classification. To enable comparison of generated events, we first compute daily count vectors comprising elements $c_{\alpha,t}(u)$ for each user u, where α is the respective event type and t is the observed day. The trained model thus consists of a list of count vectors from previous observations. To phase out old count vectors that may not be representative for the current user behavior anymore, we design the list as a queue of length q where new or matching vectors are always moved to the beginning of the queue. We measure the difference between count vectors of any new day and all count vectors available in the model for that user with the l_1 norm since it is more suitable for high-dimensional data than higher-order norms

TABLE V: Sample computation of the anomaly score

Event types	Model		Test	t_1, t_3		t_2, t_3	
α	c_{α,t_1}	c_{α,t_2}	c_{α,t_3}	Diff.	Max.	Diff.	Max.
file_accessed	2	-	-	2	2	0	0
file_created	1	-	2	1	2	2	2
file_updated	5	15	14	9	14	1	15
file_written	6	15	16	10	16	1	16
login_attempt	-	4	4	4	4	0	4
login_successful	-	4	4	4	4	0	4
Sum	-	-	-	30	42	4	41
Score	-	-	-	0.71		0.1	

[24]. We also normalize the l_1 norm by dividing by the sum of the maximum of each vector element according to Eq. 2.

$$score_{t_j}(u) = \min_{i=j-q}^{j-1} \left\{ \frac{\sum_{\alpha \in a(u)} \left| c_{\alpha,t_j}(u) - c_{\alpha,t_i}(u) \right|}{\sum_{\alpha \in a(u)} \max\left(c_{\alpha,t_j}(u), c_{\alpha,t_i}(u) \right)} \right\} \quad (2)$$

Daily event occurrences are considered anomalous when their resulting anomaly scores exceed a pre-defined detection threshold, i.e., $score_{t_j}(u) > \theta$. We provide an exemplary computation of the anomaly score in Table V, which shows three daily count vectors with six distinct event types taken from user "ethical-lavender-clownfish-stagemover", where t_1 and t_2 are assumed to be part of the baseline and t_3 is a new observation. We state the differences and maxima of elements in the count vectors of t_1 and t_2, which correspond to the numerator and denominator of Eq. 2 respectively. The row containing the sums shows that the vectors differ in 30 out of 42 possible positions, thus yielding an anomaly score of $30/42 = 0.71$. Considering only these two count vectors and assuming a threshold of $\theta = 0.5$, t_3 would be reported as an anomaly; however, since t_2, t_3 yields a score of only 0.1, the new observation is similar enough to one of the count vectors already contained in the baseline model to be regarded as normal behavior. As visible from this example, an intuitive aspect of this detection method is that it treats event types that are only present in one of the two compared vectors as zero, which also has the benefit that the set of available event types does not need to be known beforehand and that count vectors of the model do not require updating. Moreover, as the method relies on similarity-based clustering, it is not necessary to assume that event counts follow any statistical distributions.

We recognize that event types that usually occur with higher frequencies have stronger influence on the anomaly score and therefore propose two variants that modify the count vectors. The first variant stated in Eq. 3 normalizes the vectors so that the measure relies on relative rather than absolute frequencies.

$$c^{norm}_{\alpha,t}(u) = \frac{c_{\alpha,t}(u)}{\sum_{\alpha' \in a(u)} c_{\alpha',t}(u)} \quad (3)$$

The second variant assumes that event types used by more users are less relevant for detection, and vice versa. The metric thus assigns weighs to all event types similar to the well-known tf-idf statistic [25]. In particular, Eq. 4 multiplies the elements of the count vectors with the fraction of all users $\mathcal{U}_t$ observed until time t over the number of users that have already used that specific event type. Note that we add 1 in the

numerator to ensure that every event type receives a non-zero weight even if it is generated by all users.

$$c^{idf}_{\alpha,t}(u) = c_{\alpha,t}(u) \cdot log\left(\frac{1+|\mathcal{U}_t|}{|\{u' \in \mathcal{U}_t : \alpha \in a(u')\}|}\right) \quad (4)$$

D. Evaluation

To evaluate our illustrative scenario, we first generate a data set with injected attacks as described in Sect. V-A. In particular, we randomly select 10 pairs of users that satisfy our selection criteria with parameters $d_{min} = 25$, $c_{min} = 100$, $a_{min} = 4$, $s_{min} = 0.1$, $s_{max} = 0.6$, $\omega_1 = 0.3$, and $\omega_2 = 0.7$. We then apply the anomaly detection approach from Sect. V-C using the dynamic learning strategy from Sect. V-B. We count the first day with user activity after switching each user pair as true positive (TP) if the day is detected as an anomaly, and as false negative (FN) otherwise. We count all other days with user activity as false positives (FP) if they are detected as anomalous, and as true negatives (TN) otherwise. Figure 8 provides examples for each of these classes. Note that this evaluation setup only considers days where models are not re-trained, which means that all days that are correctly detected as anomalous but fall in re-training phases due to some false positives occurring before are not counted as true positives. To avoid this issue, we also count these days separately in an adjusted true positive score (TP_{adj}) that keeps track of all correctly detected days independent of training phases. We then compute $TPR = \frac{TP}{TP+FN}$, $TPR_{adj} = \frac{TP_{adj}}{TP_{adj}+FN_{adj}}$, and $FPR = \frac{FP}{FP+TN}$. Moreover, we count the number of days spent for re-training in relation to all days with user activity, because a high fraction of re-training days ($R = \frac{\text{number of training days}}{\text{total number of days}}$) means that comparatively few days are used for detection, which limits practical applicability.

Figures 9a-9c visualize these metrics for various parameter settings. Figure 9a shows how the detection threshold as well as queue size q influence the results. For lower detection thresholds, we obtain a higher TPR_{adj} since more days are detected as anomalous; however, TPR declines for $\theta \leq 0.25$, because the higher amount of FP cause that more correctly detected days appear during re-training. Furthermore, lower queue sizes improve detection of anomalous days but also increase FPR and R as more days are incorrectly detected. Based on these results we select $q = 30$ as well as $\theta = 0.6$ as a trade-off between a high true positive rate ($TPR = 65\%$), low false positive rate ($FPR = 4.6\%$ or less than 2 false alarms per day on average), and few days used for re-training ($R = 6.8\%$). Figure 9b shows that a higher number of days used to re-train the models reduces false positives but at the same time significantly increases R. The plot also shows that detection rates are largely unaffected for $\theta \geq 0.6$. We therefore select $r = 1$ as the best setting for our experiments. Finally, Fig. 9c shows the influences of count vector variations. As visible in the figure, the idf-based variation yields higher TPR for $\theta > 0.7$ than the default case, but also has the disadvantage of higher FPR and R. The main reason why this variant is unable to outperform the default variant is that anomalies often manifest in commonly used event types, which receive lower weights. On the other hand, the normalized variant yields low FPR and R but cannot compete with the other variants in terms of detection rates. Overall, the default variant appears as the best selection for this data set.

VI. Discussion

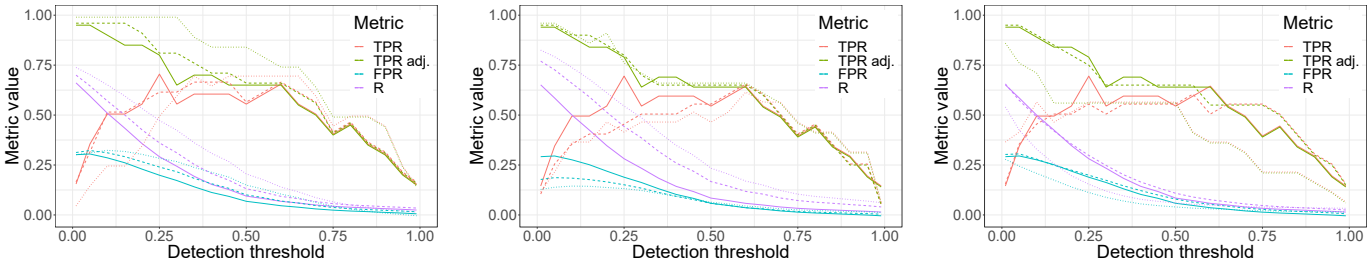
The data set presented in this paper shows how diverse and erratic user behavior can be in real cloud computing environments. Not only do users utilize the system differently in terms of used functions and access frequencies; there are many dynamic changes that are often neglected when evaluating anomaly-based systems, including evolution of the user community (e.g., new users appear and others stop using the system), changes of user behavior patterns (e.g., as consequences of users employing different tools), and technological changes of system infrastructures and configurations (e.g., new or modified log event types). It is particularly interesting to see that user behavior does not just involve direct interactions between the human and the system, but that manifestations of user activity blend with artifacts generated by automatic scripts and tools used to access the system.

Despite frequent changes of the overall system behavior, there are also remarkable regularities in the data, such as correlations between event types, communities that users interact with, and periods where users generate events with stable frequencies. The latter makes the data set especially interesting for adaptive learning mechanisms that aim to detect points in time where system behavior changes while immediately adapting to the new baselines.

Our proposed strategies for measuring similarities between users as well as detecting anomalies in the data set only makes use of event type frequencies; this decision is motivated by our observation that most long-term users utilize many distinct event types (cf. Sect. IV-A). It may be beneficial to extend our comparisons to other attributes in the data, including geolocation, accessed paths, related users, and times of day when events are generated. Thereby, it is important to adequately adapt the attack injection method to also modify attributes in addition to the user identifier to avoid that anomaly detection becomes too trivial. For example, utilizing attribute "loginName" of event type "login_attempt" likely makes it too easy to detect changed users as the names they use for logging in are rather unique. Other than that, it is also possible to rely on different methods for attack simulation, for example, replacing one user with another (i.e., deleting one of the users after switching), injecting new or modified events for some users, or inserting an artificially generated user. In that regard it could also be interesting to pursue classification of users rather than detection of single anomalous days. We leave these tasks for future work.

VII. Conclusion

We present a real cloud computing log data set for anomaly detection. Thereby, anonymization of sensitive data is carried out in such a way that no information is lost when applying user and entity behavior analysis. The log events comprise



(a) Influence of model size (solid: $q = 30$, dashed: $q = 15$, dotted: $q = 5$).

(b) Influence of re-train duration (solid: $r = 1$, dashed: $r = 2$, dotted: $r = 3$).

(c) Influence of mode (solid: default, dashed: idf, dotted: normalized).

Fig. 9: Evaluation results under varying conditions.

information on the user, type of action (e.g., a user logging in or accessing a file), event timestamp, and context (e.g., system paths or geolocations). Our analysis shows that users exhibit distinct behavior patterns, for example, based on average activity rates, system utilization, and communities. Moreover, we highlight the dynamic nature of the log data set, in particular, the unstable behavior patterns caused by long-term changes of system utilization. Based on these insights we design an adaptive anomaly detection technique that is capable of handling unstable data through repeated re-training of learned models. To evaluate our approach, we also outline a strategy for injecting attack consequences in the data, in particular, we simulate account hijacking by switching identifiers of similar users. For future work, we plan to inject also other types of anomalies in the data.

Acknowledgment

This work was partly funded by the FFG project DECEPT (873980). The authors thank Walter Huemer, Oskar Kruschitz, Kevin Truckenthanner, and Christian Aigner from Huemer Group for supporting the collection of the data set.

References

[1] G. Bassett, C. D. Hylender, P. Langlois, A. Pinto, and S. Widup, "Verizon data breach investigations report," pp. 1–108, 2022.

[2] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *Journal of Network and Computer Applications*, vol. 79, pp. 88–115, 2017.

[3] A. G Martín, A. Fernández-Isabel, I. Martín de Diego, and M. Beltrán, "A survey for user behavior analysis based on machine learning techniques: current models and applications," *Applied Intelligence*, vol. 51, no. 8, pp. 6029–6055, 2021.

[4] M. Shashanka, M.-Y. Shen, and J. Wang, "User and entity behavior analytics for enterprise security," in *International Conference on Big Data*. IEEE, 2016, pp. 1867–1874.

[5] Y. Gao, Y. Ma, and D. Li, "Anomaly detection of malicious users' behaviors for web applications based on web logs," in *International Conference on Communication Technology*. IEEE, 2017, pp. 1352–1355.

[6] F. Skopik, M. Wurzenberger, and M. Landauer, *Smart Log Data Analytics*. Springer, 2021.

[7] M. Landauer, F. Skopik, M. Wurzenberger, W. Hotwagner, and A. Rauber, "Have it your way: generating customized log datasets with a model-driven simulation testbed," *IEEE Transactions on Reliability*, vol. 70, no. 1, pp. 402–415, 2020.

[8] M. Landauer, F. Skopik, M. Frank, W. Hotwagner, M. Wurzenberger, and A. Rauber, "Maintainable log datasets for evaluation of intrusion detection systems," *arXiv preprint arXiv:2203.08580*, 2022.

[9] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, "Robust log-based anomaly detection on unstable log data," in *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 807–817.

[10] R. Uetz, C. Hemminghaus, L. Hackländer, P. Schlipper, and M. Henze, "Reproducible and adaptable log data generation for sound cybersecurity experiments," in *Annual Computer Security Applications Conference*, 2021, pp. 690–705.

[11] H. A. Kholidy and F. Baiardi, "CIDD: A cloud intrusion detection dataset for cloud computing and masquerade attacks," in *International Conference on Information Technology-New Generations*. IEEE, 2012, pp. 397–402.

[12] G. O. Ganfure, C.-F. Wu, Y.-H. Chang, and W.-K. Shih, "Deepguard: Deep generative user-behavior analytics for ransomware detection," in *International Conference on Intelligence and Security Informatics*. IEEE, 2020, pp. 1–6.

[13] K. Singh, P. Singh, and K. Kumar, "User behavior analytics-based classification of application layer http-get flood attacks," *Journal of Network and Computer Applications*, vol. 112, pp. 97–114, 2018.

[14] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the KDD collection," in *Wireless Communications and Networking Conference*. IEEE, 2013, pp. 4487–4492.

[15] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *International Carnahan Conference on Security Technology*. IEEE, 2019, pp. 1–8.

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," 2018.

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

[18] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *International conference on dependable systems and networks*. IEEE, 2007, pp. 575–584.

[19] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *ACM conference on computer and communications security*, 2017, pp. 1285–1298.

[20] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *International Conference on Software Engineering Companion*, 2016, pp. 102–111.

[21] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: A large collection of system log datasets towards automated log analytics," *arXiv preprint arXiv:2008.06448*, 2020.

[22] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Software Networking*, vol. 2018, no. 1, pp. 177–200, 2018.

[23] F. Skopik, M. Wurzenberger, G. Höld, M. Landauer, and W. Kuhn, "Behavior-based anomaly detection in log data of physical access control systems," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[24] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*. Springer, 2001, pp. 420–434.

[25] C. D. Manning, *Introduction to information retrieval*. Syngress Publishing, 2008.