Time-Window Based Group-Behavior Supported Method for Accurate Detection of Anomalous Users

Lun-Pin Yuan
Pennsylvania State University
lunpin@psu.edu

Euijin Choo
Qatar Computing Research Institute
echoo@hbku.edu.qa

Ting Yu
Qatar Computing Research Institute
tyu@hbku.edu.qa

Issa Khalil
Qatar Computing Research Institute
ikhalil@hbku.edu.qa

Sencun Zhu
Pennsylvania State University
sxz16@psu.edu

***Abstract*—Autoencoder-based anomaly detection methods have been used in identifying anomalous users from large-scale enterprise logs with the assumption that adversarial activities do not follow past habitual patterns. Most existing approaches typically build models by reconstructing single-day and individual-user behaviors. However, without capturing long-term signals and group-correlation signals, the models cannot identify low-signal yet long-lasting threats, and will wrongly report many normal users as anomalies on busy days, which, in turn, lead to high false positive rate. In this paper, we propose *ACOBE*, an *A*nomaly detection method based on *CO*mpound *BE*havior, which takes into consideration long-term patterns and group behaviors. *ACOBE* leverages a novel behavior representation and an ensemble of deep autoencoders and produces an ordered investigation list. Our evaluation shows that *ACOBE* outperforms prior work by a large margin in terms of precision and recall, and our case study demonstrates that *ACOBE* is applicable in practice for cyberattack detection.**

***Keywords*-Computer Security, Anomaly Detection, Machine Learning**

I. Introduction

Emerging cyber threats such as data breaches, data exfiltration, botnets, and ransomware have caused serious concerns in the security of enterprise infrastructures [1], [2]. The root cause of a cyber threat could be a disgruntled insider or a newly-developed malware. What is worse, emerging cyber threats are more difficult to be identified by signature-based detection methods, because more and more evasive techniques are available to adversaries. To identify the emerging cyber threats before they can cause greater damage, anomaly detection upon user behaviors has attracted focuses from large-scale enterprises [3]–[9], as anomaly detection enables security analysts to find suspicious activities that could be aftermath of cyber threats (including cyberattacks and insider threats). Adversarial activities often manifest themselves in abnormal behavioral changes compared to past habitual patterns. Our goal is to find such abnormal behavioral changes by learning past habitual patterns from organizational audit logs.

Autoencoders are one of the most well-known anomaly detection techniques [10]–[12]. They are attractive to security analysts because of their robustness to domain knowledge. Briefly speaking, an autoencoder-based anomaly detection model only learns how to reconstruct normal data; hence, in events of poor reconstruction, the input data is highly likely to be abnormal. Detailed domain knowledge is no longer required, because the complex correlation among different activities is captured by learning the normal distribution from normal activities. Furthermore, autoencoder learns features from unlabeled data in an unsupervised manner, where human interference is reduced.

However, similar to other typical anomaly detection methodologies, the autoencoder-based approaches also suffer from the overwhelming number of false positive cases. The challenge is that, while an anomaly detection model is required to be sensitive to abnormal events in order to raise anomaly alerts, the model can also be so sensitive to normal behavioral deviation that it often wrongly reports normal deviations as anomalies (hence a false positives). This challenge leads to the question: *how to further differentiate abnormal events from normal events*. While prior approaches work with data that meets the data-quality requirements (e.g., completeness, availability, consistency) for cybersecurity applications [13], important factors such as misconduct timeliness and institutional environment are not considered. To reduce false positives, we argue that it is important to also examine *long-term* signals and *group-correlation* signals, as opposed to typical autoencoder approaches that examine only *single-day* and *individual-user* signals.

The limitation of only examining *single-day* and *individual-user* signals is twofold. First, without capturing *long-term* signals, a model cannot identify low-signal yet long-lasting threats. Certain cyber-threat scenarios do not cause immediate behavioral deviation, but progressively cause small yet long-lasting behavioral deviation; for example, an insider threat is a scenario where a disgruntled employee stealthily leaks sensitive data piece-by-piece over

time [14], [15]. What is worse, without *long-term* signals, a model will likely wrongly report many normal users as anomalies on busy days (e.g., working Mondays after holidays) due to the massive burst of events in a short term. This further leads to the second limitation: without capturing *group-correlation* signals, a model cannot reduce its false positive rate in occasions (e.g., environmental change) where many users have common burst of events. For example, common traffic bursts occur among many users when there is a new service or a service outage. By examining behavioral *group-correlation*, a model may figure out that the common behavioral burst is indeed normal, assuming the more behavioral correlation a user has with the group, the less likely the user is abnormal.

To address these fundamental limitations, we propose a different methodology, which involves a novel behavioral representation. We refer to this representation as a *compound behavioral deviation matrix*. Each matrix encloses individual behaviors and group behaviors across multiple days. Having compound behaviors, we then apply our anomaly detection method, which is implemented with an ensemble of deep fully-connected autoencoders. We propose ***ACOBE***, an ***A**nomaly detection method based on **CO**mpound **BE**havior*. ACOBE has three steps in its workflow (Figure 1): it first derives compound behavioral deviation matrices from organizational audit logs; then, for each user, it calculates anomaly scores in different behavioral aspects using an ensemble of autoencoders; lastly, having anomaly scores, ACOBE produces an ordered list of the most anomalous users that need further investigation. In summary, we make the following contributions.

1) We propose a novel behavioral representation which we refer to as the *compound behavioral deviation matrix*. It profiles individual and group behavior over a time window (e.g., several days). We further apply deep fully-connected autoencoders upon such representation in order to find anomalous users. Our model outputs an ordered list of anomalous users that need to be orderly investigated.
2) We evaluate ACOBE upon a synthesized and labeled dataset which illustrates insider threats. With four abnormal users out of 929 users, ACOBE achieves 99.99% area under the ROC curve; that is, ACOBE effectively puts abnormal users on top of the investigation list ahead of normal users. Furthermore, ACOBE also outperforms our re-implementation of a prior work in terms of precision-recall curve.
3) We demonstrate with case studies that ACOBE can be applied in practice for realistic cyber threats, including botnets and ransomware attacks.

II. Related Work

Most anomaly detection models are zero-positive machine learning models that are trained by only normal (i.e., negative) data. These models are then used in testing whether an observation is normal or abnormal, assuming unforeseen anomalies do not follow the learned patterns. Kanaza et al. [3] integrated supports vector data description and clustering algorithms, and Liu et al. [4] integrated K-prototype clustering and k-NN classification algorithms to detect anomalous data points, assuming anomalies are rare or accidental events. When prior domain knowledge is available for linking causal or dependency relations among subjects, objects and operations, graph-based anomaly detection methods (such as Elicit [9], Log2Vec [16], Oprea et al. [17]) could be powerful. When little prior domain knowledge is available, Principal Component Analysis (PCA) based anomaly detection methods (for example, Hu et al. [8] proposed an anomaly detection model for heterogeneous logs using singular value decomposition) could be powerful. Contrary to zero-positive anomaly detection are semi-supervised or online learning anomaly detection, in which some anomalies will be available over time for training [18].

The autoencoder framework is a PCA approach that is widely used in anomaly detection. A typical autoencoder-based anomaly detection method learns how to reconstruct normal data. It then detects anomalies by checking whether the reconstruction error of an observation has exceeded a threshold. To detect anomalies, Zong et al. proposed deep autoencoding Gaussian mixture models [19] and Chiba et al. proposed autoencoders with back propagation [20]. Sakurada and Yairi proposed autoencoders with nonlinear dimensionality reduction [21]. Lu et al. proposed an autoencoder constrained by embedding manifold learning, MC-AEN [22]. Nguyen et al. proposed a variational autoencoder with gradient-based anomaly explanation, GEE [23]. Wang et al. proposed a self-adversarial variational autoencoder with Gaussian anomaly prior assumption, adVAE [24]. Alam et al. proposed an ensemble of autoencoders accompanied by K-mean clustering algorithm, AutoPerf [25]. Mirsky et al. proposed an ensemble of lightweight autoencoders, Kitsune [5]. Liu et al. proposed an ensemble of autoencoders for multi-sourced heterogeneous logs [6], [7]. Chalapathy et al. [26] and Zhou et al. [27] proposed robust autoencoders. For other anomaly detection methods, detail surveys can be found in [10]–[12], [28]–[31].

Each of the above autoencoder work focuses on either the optimization of a particular learning algorithm without cybersecurity context [19], [21], [22], [24]–[27], or the optimization of a learning framework with very specific cybersecurity context (i.e., network intrusion detection system) [5], [20], [23]. However, though they provide fruitful insights in their particular domains, it is hard to apply their model or framework to the other anomaly detection problems (including ours—***detection of abnormal users in a large-scale organization***) due to the requirement difference for the input data. For example, with statistical network-traffic features for individual transactions, it is difficult to

discover a disgruntled insider who exfiltrates proprietary secrets piece-by-piece in a long run.

Regardless of model and framework design, any input data that does not meet the requirement may fail the detection methodology, and this is commonly known as the *data quality challenge* [13]. While Sundararajan et al. [13] discussed six data-quality requirements (i.e., completeness, measurement accuracy, attribute accuracy, plausibility and availability, origination, and logical consistency) for cyber-security applications, important factors such as misconduct timeliness and institutional environment are not considered. Hence, in this paper, we utilize two additional signals—*long-term signals* and *group-correlation* signals—to address the anomaly detection problem challenges and limitations of the existing work (e.g., high false-positive rate).

III. Motivation

Anomaly detection upon user behaviors enables security analysts to find suspicious activities caused by cyber threats including insider threats (Section V) and cyber attacks (Section VI). Typical anomaly detection methods often suffer from the overwhelming number of false positives due to the sensitivity to normal behavioral deviation. Moreover, such methods often only output anomaly labels (i.e., either *normal* or *abnormal*). However, if a model provides only anomaly labels, security analysts will be overwhelmed by heavy workload of investigation. Hence, in practice, it is more preferable to have an ordered list of users that need to be investigated [32]. We thus define an anomaly detection problem as follows: ***given a set of per-user activities, provide an investigation list of usernames that need to be orderly investigated***. On the top of such a list are the most abnormal users.

Autoencoders have been widely used in solving such an anomaly detection problem, as it is capable of learning *what are normal* in order to find *what are abnormal*. Among the aforementioned anomaly detection methods, we find Liu et al. [6] and Hu et al. [8]'s works are representative. They are similar anomaly detection methods that reconstruct single-day individual-user behaviors. However, single-day user-behavior reconstruction is not an ideal solution for identifying cyber threats. We argue that it is important to examine *long-term* signals and *group-correlation* signals, because of the following reasons.

First, certain cyber compromise scenarios do not cause immediate behavioral deviation, but progressively cause small long-lasting behavioral deviation in different behavioral aspects across multiple days. Take Zeus botnet malware as an example, once triggered, it modifies registry values (deviation in system configuration). After a few days, it communicates with the C&C server and acts maliciously (deviation in network traffic). Since these two behavioral deviations occur on different days, only models that examine long-term behaviors can identify such an anomaly. In contrast, single-day user-behavior reconstruction may fail to identify the threats, and it may also wrongly give high anomaly scores to busy days (e.g., working Mondays and make-up days).

The granularity of feature measurements is also an important factor in building profiles to accurately capture normal user behavior. Concretely, users often tend to have more human-initiated activities (e.g., sending emails, browsing web pages, writing documents) during working hours, but more computer-initiated activities (e.g., system updates, system backups, network retries due to no password input) during off hours. Therefore, our approach also captures behaviors over multiple time-frames (e.g., *working hours* and *off hours*) within each day of the measurement window.

Second, there often exists certain behavioral correlation between an individual user and its group due to environmental change or locale change. Take environmental change as an example, when there is a new service or service outage, one can expect correlated unrecognized traffics or correlated retry traffics, respectively. Take locale change as another example, one can expect more human-initiated activities during working hours on working days, as opposed to during off hours or holidays. Based on this observation, we make the following hypothesis: the greater behavioral correlation a user has with the group, the less likely the user is compromised. A model without incorporating the group behavior may not only be ineffective in identifying anomalies, but also wrongly give high anomaly scores to individual users in events of environmental or locale changes. Hence, we propose to incorporate both individual-user behaviors and group behaviors to better capture normal behavior and avoid obvious false positives that may result from unusual yet common activities of users.

IV. Our Methodology

To address the challenges discussed in the previous subsection, we present ***ACOBE***, an ***A****nomaly detection method based on* ***CO****mpound* ***BE****havior*, where a compound behavior encloses individual-user behaviors and group behaviors across multiple time-frames and a time window in days. Having compound behavior, we then apply anomaly detection implemented with deep fully-connected autoencoders. Figure 1 illustrates the workflow: (1) ACOBE first derives compound behavioral deviation matrices from organizational audit logs, (2) for each user, ACOBE then calculates anomaly scores in different behavioral aspects using an ensemble of autoencoders, and (3) having anomaly scores, ACOBE finally produces an ordered list of users that need further investigation.

A. Compound Behavioral Deviation Matrix

A compound behavioral deviation matrix encloses deviation of individual user behavior and group behavior across multiple time-frames and multi-day time window. Figure 2

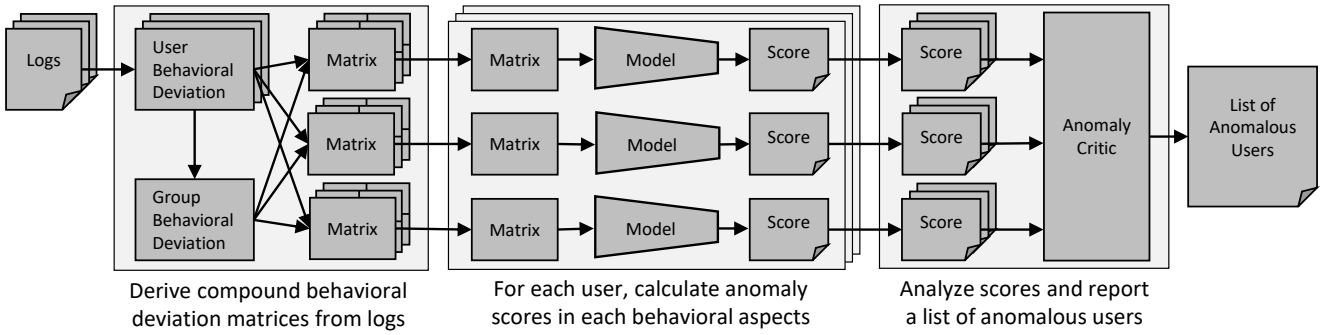


Figure 1. ACOBE Workflow

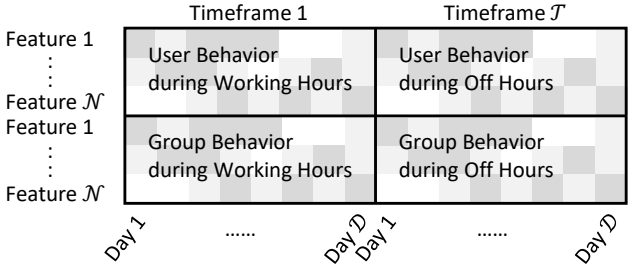


Figure 2. Compound Behavioral Deviation Matrix

illustrates an example of a compound behavioral deviation matrix with $\mathcal{F}$ features, $\mathcal{D}$ days, and $\mathcal{T} = 2$ time-frames (i.e., working hours 6am-6pm and off hours 6pm-6am). Each feature in user behavior represents a normalized characteristic of an aggregated behavior, including but not limited to, the numbers of successful logons, file accesses, failure HTTP queries (during the specific time-frame on specific day indicated by columns). Since feature selection is domain-specific, we leave the details of features in the evaluation (Section V) and case-study sections (Section VI). Features in group behavior are derived by averaging the corresponding features of all users in the group. Note that, how these four components are stacked together is not important (alternative stackings are applicable), because matrices will be flattened before going through the anomaly detection models.

We derive our deviation measurement $\sigma_{f,t,d}$ for feature f in time-frame t on day d with the below equations. $m_{f,t,d}$ denotes the numeric measurements (of feature f in time-frame t on day d). $\vec{h}_{f,t,d}$ denotes the vector of history numeric measurements (in $\omega-1$ days before day d, where ω is the window size in days). $std(\vec{h}_{f,t,d})$ denotes the standard deviation of history measurements (std is set to ϵ if it is less than ϵ to avoid divide-by-zero exception). $\delta_{f,t,d}$ denotes the variance of numeric measurement and $\sigma_{f,t,d}$ denotes the final behavioral deviation which is bounded by Δ. We bound $\sigma_{f,t,d}$ by a large Δ, as it is equivalently anomalous when $|\delta_{f,t,d}| \geq \Delta$. For example, variances larger than $\Delta = 3$ are equivalently *very abnormal*, assuming the numeric measurements follow Gaussian distribution. Note that, in events when users slowly shift their normal behavioral patterns over time, their compound behavioral deviation matrices will not show increasing deviation over time, as the history $\vec{h}_{f,t,d}$ (from which deviations are derived) slides through time and will always cover the recent shift.

$$m_{f,t,d} = \text{numeric measurements of feature } f \text{ in timeframe } t \text{ on day } d$$

$$\vec{h}_{f,t,d} = [m_{f,t,i} | i : d - \omega + 1 \leq i < d]$$

$$std(\vec{h}_{f,t,d}) = \begin{cases} \epsilon, & \text{if standard-deviation } (\vec{h}_{f,t,d}) < \epsilon \\ \text{standard-deviation } (\vec{h}_{f,t,d}), & \text{otherwise} \end{cases}$$

$$\delta_{f,t,d} = \frac{m_{f,t,d} - \text{mean } (\vec{h}_{f,t,d})}{\text{std } (\vec{h}_{f,t,d})}$$

$$\sigma_{f,t,d} = \begin{cases} \Delta, & \text{if } \delta_{f,t,d} > \Delta \\ -\Delta, & \text{if } \delta_{f,t,d} < -\Delta \\ \delta_{f,t,d}, & \text{otherwise} \end{cases}$$

Weights are applied to features, as different behaviors have different importance in capturing user behavior; for example, frequent and chaotic *file-read* activities are often less critical than rarer *file-write* activities. Since the anomaly scores are essentially the reconstruction errors of features, applying weights to features can scale-down unimportant features and thus the partial errors introduced by unimportant features. Consequently, it makes ACOBE to focus only on reconstructing important features while being more resilient to noise introduced by unimportant features. To automatically reflect the relative importance without conducting empirical studies upon individual users, Hu et al. [8] applied weights to features based on Term Frequency-Inverse Document Frequency (TF-IDF) measurements, which was originally designed for measuring the amount of information a particular text subject provides based on text frequency. Similarly, ACOBE offers the option of introducing the following feature weights $w_{f,t,d}$.

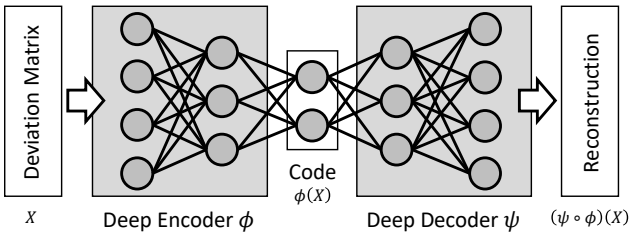


Figure 3. Deep Fully-Connected Autoencoder

$$w_{f,t,d} = \frac{1}{\log_2\left(\max\left(std(\vec{h}_{f,t,d}), 2\right)\right)} \quad (1)$$

The equation is based on the *log-normalized TF weight*, which is defined by $TF_x = \log(1 + \text{frequency of a term } x)$; that is, the less the frequency, the less information a term x could provide. Yet, unlike terms, the higher std (or equivalently more chaotic), the less information a feature f could provide. To serve our need, we inverse the equation TF_x and substitute standard deviation for frequency, so that the weights are lower for chaotic features but higher for consistent features by design. However, it cannot be infinitely high for constantly static features with very small std $(\vec{h}_{f,t,d})$, or otherwise ACOBE would be overly sensitive to small changes of static features. Therefore, a minimum value of two is given to the logarithm function, and we change the base to two so that the maximum value of weights is bounded to one. In other words, activities with small standard deviation less than two shall have equal weights of value one.

B. Anomalous Deviation Detection Model

As shown in Figure 1, we leverage an ensemble of autoencoders, each of which identifies behavioral anomalies in terms of a designated ***behavioral aspect***, where a behavioral aspect is a set of relevant behavioral features; for example, (1) *file-access* aspect includes *file-read*, *file-write*, and *file-delete* activities, (2) *network-access* aspect includes *visit*, *download*, and *upload* activities, and (3) *configuration* aspect includes *registry-modification*, *password-modification*, and *user/group-modification*.

We leverage fully-connected autoencoders in identifying anomalous compound behavioral deviation matrices. Briefly speaking, an autoencoder is an unsupervised learning method whose goal includes (1) to encode an input matrix into a representation code, and (2) to reconstruct the input matrix purely based on the representation code. Trained by normal compound behavioral deviation matrices, an autoencoder is capable of encoding and reconstructing only the seen normal behaviors; hence, poor reconstructions result from behaviors that have not yet been seen. Conceptually, an autoencoder is capable of learning *what are normal* in order to find *what are abnormal*.

To be specific, Figure 3 illustrates an example of a deep fully-connected autoencoder. It is trained by minimizing $\phi, \psi = \arg\min_{\phi,\psi} \|X - (\psi \circ \phi)(X)\|$, where ϕ is a multi-layer fully-connected encoder, ψ is a multi-layer fully-connected decoder, X is the input matrix, and $(\psi \circ \phi)(X)$ is the reconstructed matrix. In between, the code $\phi(X)$ essentially encloses the characteristics of the input matrix. Trained with only normal matrices, an autoencoder is able to reconstruct only normal matrices with minimal reconstruction errors. In events of high reconstruction errors, the abnormal matrices are likely aftermath of compromised users or unusual legitimate user activity.

C. Anomaly Detection Critic

After retrieving anomaly scores (which essentially are reconstruction errors) from an ensemble of autoencoders, our anomaly detection critic then produces a list of users that need to be orderly investigated. Recall that, we do not assign anomaly labels (e.g., *normal* or *abnormal*) to users, because providing labels without ordering is not really helpful in the context of anomaly detection, which typically has overwhelming numbers of false-positive cases. It is more preferable to have an ordered list of users that need to be orderly investigated [32].

The investigation priority of a user is derived based on the N-th highest rank of the user in different behavioral aspects (that is, in more aspects is a user top anomalous, the more anomalous the user is). For example, say $N = 2$ and a user is ranked at 3rd, 5th, 4th in terms of in-total three behavioral aspects, since 4th is the 2nd highest rank of this user, this user has a investigation priority of 4. The investigation list is sorted based on these priorities (the smaller the number, the higher the priority); if the 4 is the highest priority in the list, then this example user will be put on top of the list. A simpler way to understand N is to imagine that N is the number of votes required from each behavioral aspects. Having the investigation list, security analysts may decide how many users they want to investigate. For example, they can investigate only top 1% of the users, or stop even earlier if they have checked a certain number of users and found nothing suspicious.

V. Evaluation Upon Synthesized Data

We evaluate our proposed work upon the CERT Division Insider Threat Test Dataset [14], [15], which is widely used in the literature [33]. It is a synthesized dataset that simulates a large-scale organizational internet.

Implementation: We implement the autoencoder model with Tensorflow 2.0.0. Each fully connected layer is implemented with *tensorflow.keras. layers.Dense* activated by ReLU. The numbers of hidden units at each layer in the

Algorithm 1: Anomaly Detection Critic

Input: the number N, and a set of users $\mathbb{U} = \{\mathcal{U}_1, \mathcal{U}_2, \dots\}$, where each user has ranks $\mathcal{U}_i.\mathbb{R} = \{\mathcal{R}_1, \mathcal{R}_2, \dots\}$ in different aspects

Output: a list of users ordered by priority

$\mathbb{P} \leftarrow$ a list that stores $(user, priority)$ tuples
foreach $\mathcal{U}_i \in \mathbb{U}$ **do**
  $ranks \leftarrow$ sort $\mathcal{U}_i.\mathbb{R}$
  $priority \leftarrow ranks[N-1]$ *// index starts from 0*
  append tuple $(\mathcal{U}_i, priority)$ into $\mathbb{P}$
end
$list \leftarrow$ sort $\mathbb{P}$ based on priority
return $list$

encoder are 512, 256, 128, and 64; the numbers of hidden units in the decoder are 64, 128, 256, and 512. Between layers, Batch Normalization [34] is implemented with *tensorflow.keras.layers.Batch-Normalization*; batch normalization serves the purpose of optimizing training procedure. To train the model, Adadelta optimizer is used in minimizing Mean-Squared-Error (MSE) loss function. Before feeding compound behavioral deviation matrices, we flatten the matrices into vectors, and transform the deviations from close-interval $[-\Delta, \Delta]$ to $[0, 1]$.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\Big(X_i - (\psi \circ \phi)(X_i)\Big)^2$$

A. Dataset Description and Pre-processing

1) ***Insider Threat Scenarios****:* The dataset provides anomaly labels for five pre-defined threat scenarios. Among them, however, we evaluate our approach for the below scenarios, as we are particularly interested in user-based anomaly detection.

1) User who did not previously use removable drives or work during off-hours begins logging in off-hours, using a removable drive, and uploading data to Wikileaks.org. Leaves the organization shortly thereafter.
2) User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data.

2) ***Training Sets and Testing Sets****:* The dataset has two sub-datasets, namely, r6.1 and r6.2. Both r6.1 and r6.2 span from 2010-01-02 to 2011-05-31. Each subset contains one instance of each threat scenario; hence, there are four abnormal users in the four corresponding groups. We define their groups by their organizational departments (i.e., the third-tier organizational unit) listed in the LDAP logs. There are in total 925 normal users in these four groups. Since the four threat scenarios occur in different times, we select the training sets and the testing sets for each scenario accordingly; yet, the detection metrics in terms of true positives, false positives, and false negatives are put together into derivation of F_1 scores. For each scenario, the training set includes the data from the first collection day until roughly one month before the date of the labeled anomalies, and the testing set includes the dates from then until roughly one month after the labeled anomalies. Take r6.1 Scenario 2 as example, since the anomalies span from 2011-01-07 to 2011-03-07, we build the training set from 2010-01-02 to 2010-11-30, and the testing set from 2010-12-01 to 2011-03-30. The window size (ω) is set to 30 days.

3) ***Behavioral Feature Extraction****:* The dataset encloses a few types of logs, including device accesses, file accesses, HTTP accesses, email accesses, logon-and-logoffs, and LDAP. For presentation purpose, we only present the logs and features that are strongly related to this evaluation. For each log type, we split the log entries by user ID, and then for each user ID we extract a set of behavioral deviations $\sigma_{f,t,d}$, each of which represents the number of instances of feature f during the time-frame t on the day d. Feature weights $w_{f,t,d}$ are applied.

1) ***Device Accesses:*** This category encloses the usage of thumb drives. Each log entry includes an *activity* (either *connect* or *disconnect*) and a *host ID* to where a thumb drive is connected. There are in total two deviation features in this category: (f1) ***connection***, the number of connections and (f2) ***new-host-connection***, the number of connections to a new host that the user never had connected to before day d.
2) ***File Accesses:*** Each log entry in this category includes an *activity* (e.g., open, copy, write), a *file ID*, and a *dataflow direction*. There are in total seven features in this category: (f1) ***open-from-local***, (f2) ***open-from-remote***, (f3) ***write-to-local***, (f4) ***write-to-remote***, (f5) ***copy-from-local-to-remote***, (f6) ***copy-from-remote-to-local***, and (f7) ***new-op***. The value of each feature is computed as the number of operation in terms of $(\textit{feature}, \textit{file-ID})$ pair that the user never had conducted before day d.
3) ***HTTP Accesses:*** Each log entry in this category includes an *activity* (e.g., visit, download, upload), a *domain*, and a *filetype* that is being downloaded or uploaded. We do not take *visit* and *download* into considerationThere are in total seven features in this category: (f1) ***upload-doc***, (f2) ***upload-exe***, (f3) ***upload-jpg***, (f4) ***upload-pdf***, (f5) ***upload-txt***, (f6) ***upload-zip***, and ***http-new-op***, (f7). The value of each feature is computed as the number of operation in terms of $(\textit{feature}, \textit{domain})$ pair that the user never had conducted before day d).

Figure 4 depicts the behavioral deviation matrices of the abnormal user *JPH1910*. The upper two sub-figures are behavioral deviation in the *device-access* aspect (with two fea-

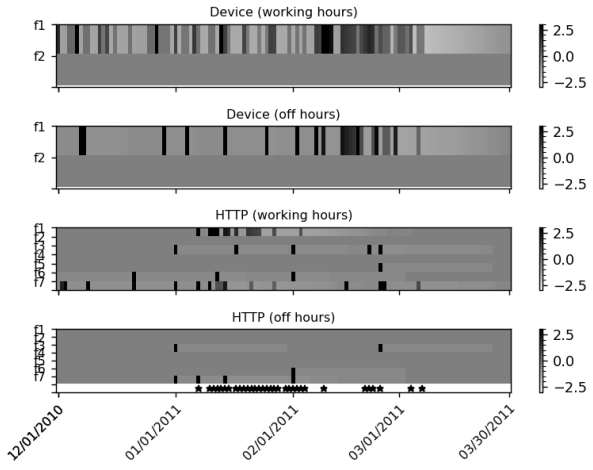


Figure 4. Example of Abnormal Behavioral Deviations

tures), during working hours and off hours, respectively. The lower two sub-figures are behavioral deviation in the *HTTP-access* aspect (with seven features), during working hours and off hours, respectively. The star markers at the bottom indicate the labeled abnormal days; however, unlabeled days are not necessarily normal, as we observed identical events being both labeled and unlabeled. Behavioral deviations $\sigma_{f,t,d}$ are in range $[-\Delta, \Delta] = [-3, 3]$. We can see in that this user *JPH1910* has abnormal deviation pattern in the *HTTP upload-doc* feature (first row) starting from January. These deviations are caused by uploading *"resume.doc"* to several websites, and these events also cause noticeable deviation in the *HTTP new-op* feature (last row). Dark deviations have white tails, because sliding history window is applied (as they change $mean$ and std that were used in deriving latter $\sigma_{f,t,d}$; note, the length of tails depends on the window size).

B. Research Questions

ACOBE is different from prior work (e.g., [6], [8]) in designs, and we are interested in the following questions:

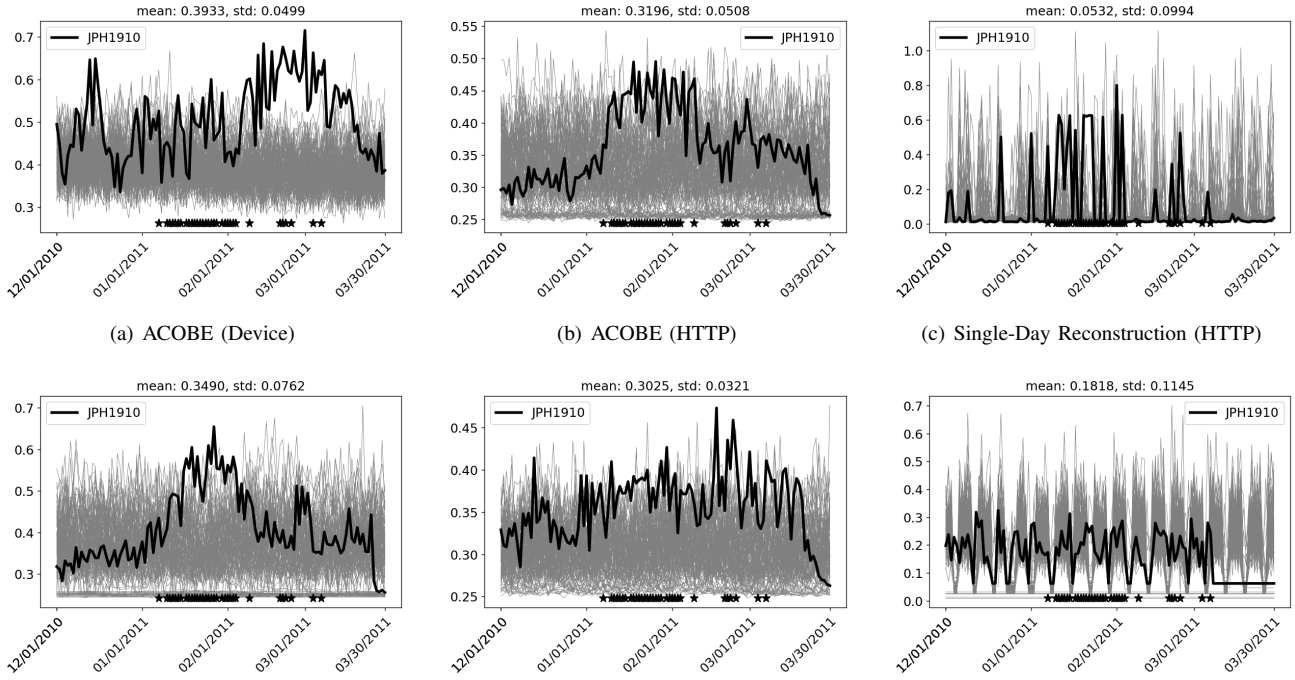
1) How does reconstructing multiple days help with detection of abnormal users in a large-scale organization?
2) How does including group deviations help?
3) How does splitting behavioral aspect help?

To answer these questions, we present r6.1 Scenario 2 in Figure 5. Each sub-figure depicts the trends of anomaly scores of 114 users in the department, to which *JPH1910* belongs. The black line depicts the score trend of the abnormal user, and the grey lines depict the score trends of 113 normal users. The star markers at the bottom indicate the labeled abnormal days. Mean and standard deviation (*std*) on top of each sub-figures are derived by all data points in each sub-figure. From Figure 5(a) and Figure 5(b), we can see that *JPH1910* has higher anomaly scores on the dates when we observe the abnormal patterns shown in Figure 4.

*1) **Long-term vs. Single-Day Reconstruction**:* We prefer long-term reconstruction rather than single-day reconstruction, because typical cyber threats (including insider threats) do not start and end within one day. Figure 5(c) depicts the anomaly scores derived by a single-day reconstruction model, which is similar to ACOBE except that the features are normalized occurrences of activities (as it no longer has history window for derived behavioral deviations). We can see that, although there are abnormal raises on the labeled days, the score waveform of the abnormal user is not distinguishable from the waveforms of normal users (peaks on weekdays and troughs on weekends); that is, single-day reconstruction cannot identify long-lasting threats that span multiple days. In contrast, as shown in Figure 5(b), ACOBE with long-term reconstruction can demonstrate the score waveform of the abnormal user, and on some dates the anomaly score stands out on top of all users. The score gets higher as the abnormal behavior patterns continue to appear in the compound behavioral deviation matrix. The anomaly scores shortly remain high and then decreases as the abnormal patterns gradually slides out of the matrix.

*2) **With Group Deviations vs. Without Group Deviation**:* Figure 5(d) depicts the trends of anomaly scores of matrices without group deviations (the other configurations are the same as in ACOBE). While we can see that the score waveform of the abnormal user still seems alike and distinguishable as in Figure 5(b), we argue that this model is less ideal. Without considering behavioral correlation between a user and the group, this model overly emphasizes self-deviating users, and thus mis-ranks normal users before abnormal users. In addition, we can see that the average of anomaly scores on top of Figure 5(d) is higher (which means reconstruction errors are higher), despite that the size of behavioral matrices is cut in half due to the absence of group deviations. In contrast, ACOBE, with such correlations, reduces not only the mis-rankings, but also the average of anomaly scores, meaning that group behavior indeed help with reducing reconstruction errors of normal behavioral matrices. In the following section, we further show that ACOBE outperforms the corresponding long-term model without group deviations (denoted as *No-Group* model).

*3) **An Ensemble of Autoencoders vs. One Autoencoder**:* The drawback of deploying just one autoencoder for all features is that, the model may be too sensitive to noise introduced by irrelevant features. ACOBE suffers from the common limitation among all other anomaly detection methods: if a set of features cannot describe a cyber threat, ACOBE may not be able to identify cyber compromises. Figure 5(e) depicts the anomaly scores derived by all-in-one model. The included features are the same features in *device-access*, *file-accesses*, and *HTTP-accesses* aspects, and the model configurations are identical to the one for ACOBE. Considering the usage of thumb drives is critically abnormal under this scenario, this model is not ideal,



(a) ACOBE (Device) (b) ACOBE (HTTP) (c) Single-Day Reconstruction (HTTP)

(d) Excluding Group Deviations (HTTP) (e) All-in-One Autoencoder (f) Baseline

Figure 5. Trends of anomaly scores of 114 users in r6.1 Scenario 2 under different model configuration

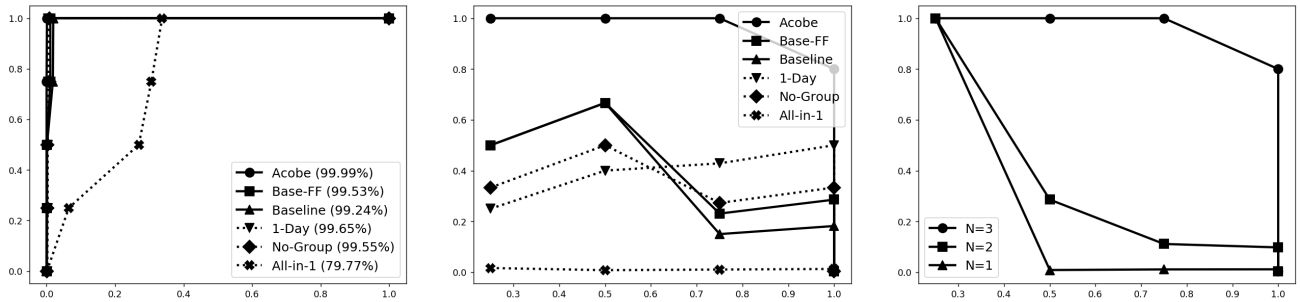
because the waveform is not as outstanding as shown in Figure 5(a). Without emphasis on the *device-accesses* aspect, this model may wrongly report normal users that have slightly chaotic behaviors in the other aspects. A common approach to resolve the drawback is to deploy an ensemble of autoencoders that each takes responsibility in one aspect (related work includes [5]–[7], [25]). This approach gives each autoencoder only necessary features and thus reduces the unwanted noise.

C. Comparison with Prior Work

We compare our work with a state-of-the-art deep autoencoder model that works on the same dataset proposed by Liu et al. [6]. We refer to their model as the ***Baseline*** model. The differences between ACOBE and Baseline include the followings. First, for each user, Baseline builds four autoencoder for coarse-grained unweighted features from the numbers of *activities* (e.g., connect, write, download, logoff) in four aspects (i.e., *device*, *file* and *HTTP*, and *logon*), whereas ACOBE builds three autoencoders for fine-grained weighted behavioral deviations that also include *new-ops* and *file types*. Second, Baseline reconstructs normalized features on individual days, whereas ACOBE reconstructs behavioral deviations across multiple days. Third, Baseline does not take group features into consideration, whereas ACOBE embeds group deviations into compound behavioral deviation matrices. Fourth, Baseline splits one day into 24 time-frames (i.e., 24 hours), whereas ACOBE splits one day into two (i.e., *working hours* and *off hours*); yet, the number of time-frames contribute negligible performance difference for this dataset.

From the discussions in the previous section, we anticipate that Baseline may not perform well due to the above differences. Figure 5(f) depicts the anomaly scores of users under r6.1 Scenario 2, and the scores of the abnormal user do not stand out on any dates. To have a fair comparison, we also build an alternative Baseline model that leverages our fine-grained features, and we refer to the new detection model as the ***Base-FF*** model. We implement Baseline and Base-FF with Tensorflow 2.0.0. Each fully connected layer is implemented with *tensorflow.keras.layers.Dense* activated by ReLU. The numbers of hidden units at each layer in the encoder are 512, 256, 128, and 64; the numbers of hidden units in the decoder are 64, 128, ,256, and 512. Between layers, *tensorflow.keras.layers.Batch-Normalization* is applied; it serves the purpose of optimizing training procedure [34]. To train the model, Adadelta optimizer is used in minimizing the Mean-Squared-Error (MSE) loss function.

We leverage the below metrics to compare the models, where TP, FP, TN, and FN denote the numbers of true positives, false positives, true negatives, and false negatives, respectively. These numbers are determined by the number of *how many user investigations the security analysts can conduct*. For example, say security analysts can investigate 1% of the users, and a user $\mathcal{U}_i$ has its investigation priority ranked at less than 1%, then $\mathcal{U}_i$ is considered to be abnormal



(a) ROC (TP Rate - FP Rate) Curve (b) Precision-Recall Curve (c) Precision-Recall Curve

Figure 6. Comparisons Between Models

and will be investigated. Say, $\mathcal{U}_i$ is however normal, then $\mathcal{U}_i$ is a FP case. Similarly, say $\mathcal{U}_j$ has its priority ranked greater than 1% and thus is marked as normal while $\mathcal{U}_j$ is indeed abnormal, then $\mathcal{U}_j$ is a FN case.

$$\text{TP Rate} = \frac{TP}{TP+FN} \qquad \text{FP Rate} = \frac{FP}{FP+TN}$$
$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Recall} = \frac{TP}{TP+FN}$$

Previous work was evaluated by the area under Receiver Operating Characteristic (ROC) curve, where the X-axis represents the FP Rate, and the Y-axis represents the TP Rate. Figure 6(a) depicts the ROC curves of different models. The ROC curve is useful when security experts conduct orderly investigations upon ordered users, as the curve depicts the expected trade-offs between TP Rate and FP Rate throughout the investigation process. Basically, the larger the area under the curve, the better the anomaly detection approach is. Since we only have four positive (abnormal) cases, we only have four data points on each curve. Note that, if a FP and a TP has the same top N-th rank, the FP is listed before the TP to illustrate the worst-case investigation order. For reference purpose, we also include the models we have discussed in previous subsections in this comparison, namely, Excluding Group Deviations (No-Group), Single-Day Reconstruction (1-Day), and All-in-one Autoencoder (All-in-1).

In Figure 6(a), the curve of ACOBE is almost a right-angle line, and the areas under ROC curves (AUC) is 99.99%. There are in total 0, and 1 FP (out of 925 normal users) listed before the 3rd and 4th TPs, respectively (that is, the users on top of the investigation list are [TP, TP, TP, FP, TP, ...], and recall that there are only four TPs). We can see that ACOBE outperforms Baseline (with AUC of 99.23%) and Base-FF (with AUC of 99.54%), as well as other sub-optimal configurations. Baseline has 1, 1, 17, and 18 FPs listed before its 1st, 2nd, 3rd, and 4th TPs, respectively. Base-FF has 1, 1, 10, and 10 FPs, respectively. If Baseline constantly provides results with $18/925$ FP Rate, security analysts could be overwhelmed in conducting timely investigation depends on the scale of the organization.

We also present the Precision-Recall curve in Figure 6(b), as ROC metric is known to be misleading (overly optimistic) for imbalanced dataset [35] due to the significantly larger number of negative cases. In Figure 6(b), the X-axis represents the recall, and the Y-axis represents the precision. The importance of this curve includes that the calculations do not make use of the number of TNs, and thus the curve is concerned with only the correct prediction of the small number of positive cases. Based on the Precision-Recall curves, we can differentiate ACOBE from Baseline and Base-FF by a large margin. For reference, while ACOBE's works with $N = 3$, we also plot the curves of alternative ACOBE's that work with $N = 2$ and $N = 1$ in Figure 6(c).

Based on the above, we conclude that ACOBE outperforms the Baseline model by a large margin in terms of the Precision-Recall metric, meaning that ACOBE is more effective in correctly ranking abnormal users before normal users when providing an ordered investigation list.

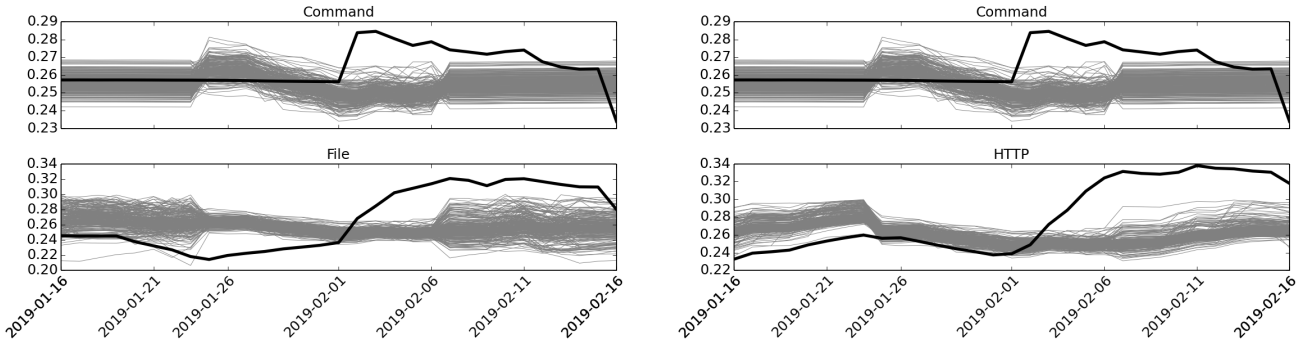
VI. Case Studies Upon Real Data

We applied ACOBE to a real-world enterprise dataset. We gathered a set of audit logs that spans seven months. Audit logs were generated on Windows servers and web proxies, and they were gathered through the ELK stack [36]; note that, we do not have audit logs from endpoints. This dataset spans seven months, including six months of training set and one month of testing set.

Ethics: It is important to note that our enterprise set is anonymized where possible privacy concerns are carefully addressed. All identifiable and personal information (e.g., user names/IDs and email addresses) are replaced with securely hashed pseudo information before use.

A. Audit and Attack Scenarios

Our Windows servers provide logs of the following audit categories: *Windows-Event auditing* (for application, security, setup, and system events), *PowerShell auditing* (*Microsoft-Windows-PowerShell/Operational*), *System-Monitor auditing* (*Microsoft-Windows-Sysmon/Operational*),



(a) Ransomware's Anomaly Score

(b) Zeus-Bot's Anomaly Score

Figure 7. Case Studies: Ranwomware and Zeus-Bot

and DNS-query logs. To reduce daily log size, we discard events of noisy and redundant event types, including *Process Access* (event ID: 10). Web proxies provide its native system and service logs (including syslog and DHCP logs) and informative proxy logs (where each entry includes user, source, destination, resources, and various security verdicts).

For presentation purpose, we present only employee accounts (say, *alice* is an employee acount), but employee accounts are integrated with (have activities from) computer accounts (e.g., *alice$*), email accounts (e.g., *alice@enterprise.com*), and domain accounts (e.g., *ENT\alice*). Note that, we exclude service accounts (e.g., *httpd*), and privileged accounts (e.g., *root*) from this case study, because they do not demonstrate habitual patterns like human. By doing so, we have 246 employees in this dataset.

To work with this dataset, we launched the following two attacks in the same environment under control.

- **Zeus Botnet:** Malicious activities include downloading Zeus from a downloader app, deleting this downloader app, and modifying registry values. Our Zeus bot also made queries to non-existing domains generated by *newGOZ* (a domain generation algorithm found in Gameover Zeus and Peer-to-Peer Zeus [37], [38]).
- **Ransomware:** We use the WannaCry samples available in Malware DB [39]. Malicious activities include modifying registry values and encrypting files.

B. Behavioral Feature Engineering

We design a set of behavioral features for ACOBE to work with this real-world dataset. However, note that we want to emphasize only that *"ACOBE can be applied to real audit logs in practice"*, but not that *"we have fairly working features"*. It is hard to craft good behavioral features by hand for a complex dataset, and we are aware that our feature design is not perfect. For presentation purpose, we split our features into two categories listed below; for simplicity, we present only the features that are related to this case study. That said, we have in total 27 behavioral features, 16 of which from four behavioral aspects (namely, *File*, *Command*, *Config*, and *Resource*) and 11 from statistical aspects (namely *HTTP* and *Logon*). The window size for our compound behavioral deviation matrix is two weeks.

*1) **Predictable Behavioral Aspects**:* When dependency or causality exists among consecutive events, we may predict upcoming events based on a sequence of events. To measure how an event sequence deviates from a user's habitual pattern, we may leverage deep-learning based anomaly detection model for discrete events (e.g.,DeepLog [40]). In this case study, we present two (out of four) predictable aspects for this dataset: ***File*** accesses such as file-handle operations, file shares, and Sysmon file-related events (event IDs: 2, 11, 4656, 4658-4663, 4670, 5140-5145), and ***Command*** executions such as process creation and PowerShell execution (1, 4100-4104, 4688). For simplicity, we present only the following three features.

- f1: the number of events during a period
- f2: the number of unique events during a period
- f3: the number of new events during a period

*2) **Statistical Behavioral Aspects**:* When we cannot easily predict upcoming events, we may craft statistical features for statistically structural behaviors. For example, Siadati et. al [41] built features for structural logon patterns in an enterprise. Similarly, we present one (out of two) statistical behavioral aspect: ***HTTP*** traffic. For simplicity, we present only the following four features.

- f1: # of successful requests during a period $\mathcal{P}$
- f2: # of successful requests to a new domain during $\mathcal{P}$
- f3: # of failure requests during $\mathcal{P}$
- f4: # of failure requests to a new domain during $\mathcal{P}$

C. Detection Results

We showcase how malicious activities impact the victim's behavioral matrix and anomaly scores in the two cyber attack scenario. We have 246 employees, and one of which is under the aforementioned two attacks on Feb 2nd.

Figure 7(a) and Figure 7(b) show examples of how each aspect affects the anomaly scores. We see that the waveforms have significant rises after the attack day (i.e., Feb 2nd). In both attack scenarios, we see positive deviations in the *Command* aspect. These deviation are caused by the newly observed executions of the malware programs. Since the victim barely has any activities in the *Command* aspect, such deviations are significant. Though not shown in this paper, we see the same positive deviation in the *Config* aspect, as two attacks both modified registry values shortly after being triggered. In the ransomware scenario, deviations in the *File* aspect are caused by newly observed *read*, *write*, and *delete* operations conducted by the malware. In the botnet scenario, deviations in the *HTTP* aspect are caused by successful connections to the C&C server and failure connections to *newGOZ* domains.

Considering all aspects together, our victim is ranked at 1st place in ACOBE's investigation list from Feb 3rd to Feb 15th with both ransomware and botnet malware. Security analysts can easily find the attacks if periodic investigation is enforced. In addition, we observe the followings: First, normal users together demonstrate a main stream of score trends. We can see that normal users have rises in *Command* and drops in *HTTP* on Jan 26th due to a environmental change. This again indicates that it is important to examine behavioral correlation between an individual and its group. Second, although the attack day is on Feb 2nd, the waveforms of *File* and *HTTP* do not demonstrate immediate and significant rise for our victim. This indicates that the widely used single-day detection methodology may not be able to identify the attacks. In contrast, with long-term deviation pattern embedded in behavioral matrices, the waveforms rise after the attack day. Based on the above, we argue that long-term signals and group correlations are very effective in signaling abnormal users.

VII. Discussion and Future Work

A. Challenges with Filtering Benign Anomalies

One may be concerned about the importance of group correlation by questioning *"why can't we just filter out common benign anomalies during a specific period?"* We argue that it may not be easy. Indeed, sometimes a security analyst is explicitly informed about certain common benign anomalies (e.g., environmental changes), and thus filtering is sometimes feasible. However, such information is often not readily available (as indicated by the security analysts of a large enterprise that we interacted with). To manually filter out FPs, we need to first investigate what happened during a specific period, and why/how the anomalies are benign. For example, before concluding that the cause of an anomaly is a new service, we need to first deduce that (1) several users accessed this service, and (2) this service did not appear in the audit history. After having the cause, we need to know what behaviors are affected (e.g., does it incur just new HTTP sessions, or does it incur an executable?). Only after then, we know what common behaviors can be safely excluded without incurring FNs. This procedure may not be timely affordable, and this is why we intend to leverage deep learning techniques to efficiently bypass this procedure.

B. A More Flexible Detection Critic

The anomaly detection critic in ACOBE is simple.We do not incorporate a fancy critic, as we only want to showcase the fundamental idea: an autoencoder-based anomaly-detection approach based on compound behaviors. Nevertheless, our future work includes an advanced detection critic, which may be capable of robustly filtering out some benign anomalies. This future critic shall considers more factors, including but not limited to the followings. First, from Figure 5, we can see that the anomaly score raises significantly once abnormal activities has happened; hence, we could examine *"whether the anomaly score has a recent spike"*. Second, we could examine *"whether the abnormal raise demonstrate a particular waveform"*, as different behavioral changes demonstrate different characteristics upon the anomaly score. For example, a developer starting a new project can incur a bursting raise with long-lasting but smooth decrease, whereas a cyberattack may not show the decrease but chaotic signals, as the malicious behaviors may not be consistent over time. These additional factors could be useful in anomaly detection judgement in our future work.

C. More Evaluations in the Future Work

While Yuan et. al [33] addressed that most recent deep learning-based studies adopt the CERT dataset [15] to evaluate their proposed approaches, we are eager to know how well ACOBE can work on different datasets. Moreover, we would like to compare ACOBE with more related work (not necessarily deep learning-based) in the literature, including but not limited to, for example, Log2Vec [16]. However, we could not do so by the time we wrote this paper because (1) we do not have the essential prior knowledge about specific target domains, and (2) we do not have detailed parameters for re-implementation (if not open-sourced). We need time to study different datasets and different approaches, hence we leave more evaluations in our future work.

VIII. Conclusion

The fundamental limitation of the widely-used anomaly-detection methodology is that it overlooks the importance of long-term cyber threats and behavioral correlation within a group. To this end, we propose a behavioral representation which we refer to as a *compound behavioral deviation matrix*. Having such matrices, we then propose ACOBE, an autoencder-based anomaly detection for compromises. Our evaluation and case studies show that, ACOBE not only outperforms prior related work by a large margin in terms of precision and recall, but also is applicable in practice for discovery of realistic cyber threats.

REFERENCES

[1] G. Belani, "5 cybersecurity threats to be aware of in 2020," https://www.computer.org/publications/tech-news/trends/5-cybersecurity-threats-to-be-aware-of-in-2020.

[2] D. Rafter, "Cyberthreat trends: 15 cybersecurity threats for 2020," https://us.norton.com/internetsecurity-emerging-threats-cyberthreat-trends-cybersecurity-threat-review.html.

[3] T. Kenaza, K. Bennaceur, and A. Labed, "An efficient hybrid svdd/clustering approach for anomaly-based intrusion detection," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC 18. New York, NY, USA: Association for Computing Machinery, 2018, p. 435443. [Online]. Available: https://doi.org/10.1145/3167132.3167180

[4] Z. Liu, T. Qin, X. Guan, H. Jiang, and C. Wang, "An integrated method for anomaly detection from massive system logs," *IEEE Access*, vol. 6, pp. 30 602–30 611, 2018.

[5] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018.

[6] L. Liu, O. De Vel, C. Chen, J. Zhang, and Y. Xiang, "Anomaly-based insider threat detection using deep autoencoders," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov 2018, pp. 39–48.

[7] L. Liu, C. Chen, J. Zhang, O. De Vel, and Y. Xiang, "Unsupervised insider detection through neural feature learning and model optimisation," in *Network and System Security*, J. K. Liu and X. Huang, Eds. Cham: Springer International Publishing, 2019, pp. 18–36.

[8] Q. Hu, B. Tang, and D. Lin, "Anomalous user activity detection in enterprise multi-source logs," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov 2017, pp. 797–803.

[9] M. A. Maloof and G. D. Stephens, "Elicit: A system for detecting insiders who violate need-to-know," in *Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection*, ser. RAID07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 146166.

[10] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, p. 105124, 2020.

[11] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *CoRR*, 2019.

[12] F. Falcão, T. Zoppi, C. B. V. Silva, A. Santos, B. Fonseca, A. Ceccarelli, and A. Bondavalli, "Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 318327. [Online]. Available: https://doi.org/10.1145/3297280.3297314

[13] A. Sundararajan, T. Khan, A. Moghadasi, and A. I. Sarwat, "Survey on synchrophasor data quality and cybersecurity challenges, and evaluation of their interdependencies," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 3, pp. 449–467, 2019.

[14] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *2013 IEEE Security and Privacy Workshops*, 2013, pp. 98–104.

[15] "Insider threat test dataset," https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099.

[16] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 17771794.

[17] A. Oprea, Z. Li, T. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2015, pp. 45–56.

[18] M. Du, Z. Chen, C. Liu, R. Oak, and D. Song, "Lifelong anomaly detection through unlearning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 12831297. [Online]. Available: https://doi.org/10.1145/3319535.3363226

[19] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[20] Z. Chiba, N. Abghour, K. Moussaid, A. E. Omri, and M. Rida, "A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection," *Computers & Security*, vol. 75, pp. 36 – 58, 2018.

[21] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, ser. MLSDA14. New York, NY, USA: Association for Computing Machinery, 2014, p. 411. [Online]. Available: https://doi.org/10.1145/2689746.2689747

[22] X. Lu, W. Zhang, and J. Huang, "Exploiting embedding manifold of autoencoders for hyperspectral anomaly detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 3, pp. 1527–1537, March 2020.

[23] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *2019 IEEE Conference on Communications and Network Security (CNS)*, June 2019, pp. 91–99.

[24] X. Wang, Y. Du, S. Lin, P. Cui, Y. Shen, and Y. Yang, "advae: A self-adversarial variational autoencoder with gaussian anomaly prior knowledge for anomaly detection," *Knowledge-Based Systems*, vol. 190, p. 105187, 2020.

[25] M. Alam, J. Gottschlich, N. Tatbul, J. Turek, T. Mattson, and A. Muzahid, "A zero-positive learning approach for diagnosing software performance regressions," 2017.

[26] R. Chalapathy, A. K. Menon, and S. Chawla, "Robust, deep and inductive anomaly detection," in *Machine Learning and Knowledge Discovery in Databases*, M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, Eds. Cham: Springer International Publishing, 2017, pp. 36–51.

[27] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD 17. New York, NY, USA: Association for Computing Machinery, 2017, p. 665674. [Online]. Available: https://doi.org/10.1145/3097983.3098052

[28] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, May 2012.

[29] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

[30] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *International conference on Data Mining (full paper)*. IEEE, December 2009. [Online]. Available: https://www.microsoft.com/en-us/research/publication/execution-anomaly-detection-in-distributed-systems-through-unstructured-log-analysis/

[31] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, Oct 2016, pp. 207–218.

[32] A. Oprea, Z. Li, R. Norris, and K. Bowers, "Made: Security analytics for enterprise threat detection," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 124–136.

[33] S. Yuan and X. Wu, "Deep learning for insider threat detection: Review, challenges and opportunities," *Computers & Security*, p. 102221, 2021.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[35] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PLOS ONE*, vol. 10, no. 3, pp. 1–21, 03 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0118432

[36] "Elasticsearch: Restful, distributed search and analytics," https://www.elastic.co/.

[37] J. Bacher, "Domain generation algorithms," https://github.com/baderj/domain_generation_algorithms.

[38] "Zeus," https://github.com/Visgean/Zeus.

[39] "thezoo aka malware db - a live malware repository," https://thezoo.morirt.com/.

[40] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1285–1298.

[41] H. Siadati and N. Memon, "Detecting structurally anomalous logins within enterprise networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1273–1284. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134003