Forecasting Suspicious Account Activity at Large-Scale Online Service Providers

Hassan Halawa[1](✉), Konstantin Beznosov[1], Baris Coskun[2], Meizhu Liu[3], and Matei Ripeanu[1]

[1] University of British Columbia, Vancouver, Canada
{hhalawa,beznosov,matei}@ece.ubc.ca
[2] Amazon Web Services, New York, USA
barisco@amazon.com
[3] Yahoo! Research, New York, USA
meizhu@oath.com

Abstract. In the face of large-scale automated social engineering attacks to large online services, fast detection and remediation of compromised accounts are crucial to limit the spread of the attack and to mitigate the overall damage to users, companies, and the public at large. We advocate a fully automated approach based on machine learning: we develop an early warning system that harnesses account activity traces to predict which accounts are likely to be compromised in the future. We demonstrate the feasibility and applicability of the system through an experiment at a large-scale online service provider using four months of real-world production data encompassing hundreds of millions of users. We show that—even limiting ourselves to login data only in order to derive features with low computational cost, and a basic model selection approach—our classifier can be tuned to achieve good classification precision when used for forecasting. Our system correctly identifies *up to one month in advance* the accounts later flagged as suspicious with precision, recall, and false positive rates that indicate the mechanism is likely to prove valuable in operational settings to support additional layers of defense.

Keywords: Forecasting · Machine learning for security · Big data analytics for security · Large-scale cyberattacks · Cloud security

1 Introduction

Online services are an integral part of our personal and professional lives. To support widespread adoption and improve usability, large-scale online service providers (LSOSPs) have made it simple for users to access any of the provided services using a single credential. Such "single sign-on" systems make it much

This work was done when Baris Coskun was with Yahoo! Research.

I. Goldberg and T. Moore (Eds.): FC 2019, LNCS 11598, pp. 1–19, 2019.
https://doi.org/10.1007/978-3-030-32101-7_33

easier for users to manage their interactions through a single account and sign-in interface. As users become more invested in the platform, the single login credential becomes a valuable key to a whole set of services, as well as the ‘key’ to their digital identity and the personal information stored on the platform. As a consequence, these credentials are highly attractive targets to attackers.

As LSOSPs improve their defense systems to protect their user base, attackers have shifted their efforts to social engineering attacks: e.g, attacks that exploit incorrect decisions made by individual users to trick them into disclosing their login credentials [14]. Once an account is compromised, the attackers hijack the account from its legitimate owner and, typically, use it for their own purposes [19]: for example, to evade detection while perpetuating an attack (e.g., multi-stage phishing, or malware distribution campaigns) or to carry out other fraudulent activity (e.g., sending out spam email).

Thus, detecting compromised accounts early and giving back control to their legitimate owners quickly, as well as designing defense mechanisms that add additional layers of defense to protect users likely to fall prey to social engineering attacks, is crucial. Doing so can mitigate the damage an attacker can do while in control of a compromised account, protect the account owner’s digital identity, and reduce the damage inflicted by an automated large-scale social-engineering attack to a LSOSP and its user community. It should be noted that, detecting compromised accounts is much more challenging than just identifying fake ones (i.e., those created by an attacker) since, in the former case, suspicious activity is typically interleaved with the account owner’s legitimate activity [8].

This paper tests the hypothesis that it is feasible to identify likely future victims of mass-scale social-engineering attacks. In a nutshell, we postulate that the behavioral patterns of the users that have little incentives or low ability to fend off social-engineering attacks can be learned. To this end we propose an early warning system based on a completely automated pipeline using machine learning (ML) to identify the accounts with similar behavioral patterns to those that have been flagged as suspicious in the past.

Predicting accounts that are more likely to be compromised in the future can be used to develop new defenses, to fine-tune and better target existing defense mechanisms, as well as to better protect vulnerable users [10]. While we briefly discuss the intuition behind some of these defense mechanisms in the discussion section (Sect. 7), their design and evaluation, however, is beyond the scope of this paper and we focus here solely on evaluating our conjecture that predicting which accounts are more likely to be compromised is feasible.

We have tested our hypothesis using real-world data from a large LSOSP (i.e., at the scale of Amazon, Facebook, Google, or Yahoo). Throughout this paper we will refer to it as a *LSOSP* (in italics, the non-italicized LSOSP refers to a generic Large-Scale Online Service Provider). Our experiments were carried out over four months of production data covering hundreds of millions of users generating hundreds of billions of login events to *LSOSP*’s platform. Due to space constraints we omit some relevant in-depth descriptions from this paper, and we refer interested readers to our more exhaustive technical report [11] for more information.

This paper makes the following contributions:

- We formulate the hypothesis that it is feasible to identify the users more likely to fall prey to mass-scale social-engineering attacks (Sect. 2), propose an approach to identify their accounts, and design an early warning system (Sect. 3).
- We demonstrate the feasibility and applicability of the proposed approach on real-world production data (Sect. 5). We show that, even using low-cost features extracted from two basic datasets (Sect. 4) and a simple model selection approach (Sect. 3) leading to acceptable training runtime, the proposed classifier can be tuned to achieve good classification quality based on the recall, precision, and false positive rate metrics (Sect. 5). For example (CE_C in Sect. 5), using only *one week of login event history and predicting one month in advance*, our classifier predicts more than half of the accounts later flagged as having suspicious behaviour (i.e., achieves a recall of 50.62%) and, at the same time, around one in five of the predicted accounts is actually labeled as suspicious at *LSOSP* within a 30-day prediction horizon (i.e., precision of 18.33%, with a low false positive rate of 0.49%). While, overall, our results indicate that it is feasible to achieve good classification quality, we stress two important points: first, it is important to note that our results should be seen as a lower bound of the achievable classification performance: this can likely be further improved by using richer data or additional computational resources (e.g., to support more sophisticated learning methods). Second, efficient defense mechanisms can be developed based on future victim predictors as, for example, Boshmaf et al. [5] demonstrate in the context of a social-bot infiltration attack. We expand on these points in the discussion Sect. 7.

2 Problem Formulation

We present an overview of our problem (Sect. 2.1) by abstracting away from all company- and experiment-specific details which we describe in detail in Sects. 3, 4 and 5. Here, we go over the assumptions and objectives that influenced our approach, we elaborate on the datasets required to carry out the classification task (Sect. 2.2), and we introduce our *classification exercises (CEs)*, which are the means by which we organize our *experiments* (Sect. 2.3).

2.1 Overview

Our goal is to develop an early warning system that can be used by LSOSPs to harness observable user behavior to identify accounts likely to be compromised in the future. Our intuition is the following: over the course of everyday use, the history of user interactions encapsulates information from which one can infer whether an account is more likely to be compromised in the future (e.g., because the user does not have the interest or the ability to fend off social-engineering attacks); eventually (some of) these accounts are compromised, generate suspicious activity, and are later flagged. In other words, to forecast future suspicious activity, we aim for features that approximate user behavioral patterns to infer similarity to accounts that are later flagged as suspicious and develop a binary classifier to act as an *early warning system*. We chose a supervised learning approach as, over the past few years, it has been shown to achieve good performance for a variety of classification tasks [3,6,16,21,22].

2.2 Assumptions, Objectives, and Datasets

Assumptions. We treat the prediction of suspicious accounts as a binary classification problem (suspicious vs. non-suspicious). We assume that only a small subset of the overall population is likely to exhibit suspicious activity. We believe that this is true for large providers that offer services to a large number of users around the world (up to billions of users) and dedicate resources to maintain a "healthy" user population. The direct implication is that the ML techniques used, the data selection for the training of the classifiers, and the success metrics used are all tuned for imbalanced data.

Objectives. We aim to meet the following objectives when designing and tuning the binary classifier. First, a low rate of false positives: accounts incorrectly predicted as suspicious (i.e., false positives) should be minimized even at the cost of decreasing the number of correctly predicted suspicious accounts (i.e., true positives). This trade-off can be controlled by tuning the classifier's prediction threshold when generating the final binary classification. We also discuss tuning for a low rate of false negatives in (Sect. 7).

Second, and crucially for deployment at a LSOSP, with hundreds of millions of users and tens of billions of user activity events per day (or more!), the classifier should be optimized for runtime efficiency during both training (feature extraction and model building) and testing/use (prediction and classification). This can be accomplished by employing features that can be easily extracted/computed from the raw data, and by choosing ML models that offer a good trade-off between the quality of prediction and performance. Balancing this trade-off is crucial for timely forecasting of suspicious activity and thus faster remediation (as well as adoption in realistic settings).

Required Datasets. We assume that the LSOSP has access to at least two types of data. First, data that can be mined to extract behavioral patterns. Second, a sample of accounts previously flagged as suspicious that can be used as ground truth. We detail the data we use from *LSOSP* in Sect. 4.

2.3 Experiment Organization

Here we establish the terminology we use for the rest of this paper. We define the means by which we organize our experiments (*Classification Exercises*), and we detail the categories of accounts that can be observed in the datasets and how we use them.

Classification Exercises (CEs). CEs are our way of grouping together all parameters of a binary classification experiment (e.g., training time interval, testing time interval, ML model hyperparameters) and the associated results. As with any typical supervised ML approach, a CE is divided into two distinct phases: *training* and *testing* (Figure. 1 provides an overview). During training, our goal is to fit a model that learns user behavioral patterns that can be used as early predictors of suspicious account activity. During testing, the fitted model is applied to new data *not seen during training* and the classifier's performance is evaluated against a labeled ground truth.

Categories of Accounts. We consider U as the set of all accounts registered with the LSOSP. Depending on the scale and popularity of the LSOSP, U can be extremely large potentially exceeding a billion users. We use days as a coarse-grain measure of time. We consider L_d as the set of users with login activity on day d. For the set L_d, we extract easy-to-compute low-cost features representing the users' login behavior on day d. We aim to learn the behavioral patterns of legitimate accounts prior to them being flagged as suspicious. We denote with S_d the set of user accounts flagged as suspicious on day d. Existence of an account in set S_d on day d is a clear indication that the account exhibited some suspicious activity prior to or on day d. However, it is important to note that the opposite is not true: if an account is absent from the set S_d on day d that does not imply that it did not exhibit any irregular activity prior to or on day d. The reason for this is that the pipeline used for detecting suspicious accounts at the LSOSP is expected to have some lag. In other words, it takes time for an account to be flagged as suspicious after it first starts exhibiting irregular behavior.

Avoiding Attacker-Controlled Accounts. The set L_d contains not only legitimate user accounts but also those that are under the control of an attacker (the set A_d). These include fake as well as compromised accounts (considered as sets F_d and C_d respectively). *We implement several heuristics to prune such accounts and avoid learning user behavioral patterns from accounts that may be under attacker control.* Thus, we do not use the sets L_d and S_d directly. Instead, to avoid learning the behavior of accounts under attacker control (A_d), we prune both L_d and S_d in order to eliminate accounts that may be under attacker control. We discuss this preprocessing step in detail in Sect. 3.3.

3 Proposed Approach

This section outlines our proposed approach: the details of our classification exercises (Sect. 3.1), the proposed supervised ML pipeline (Sect. 3.2), and the heuristics we implement to avoid learning from accounts under the control of an attacker and to reduce bias when evaluating our approach (Sect. 3.3). The following sections describe our datasets (Sect. 4) and the evaluation results (Sect. 5).

3.1 Classification Exercise Composition

We organize our classification exercises (CEs) as outlined in Fig. 1. During *training*, we attempt to fit a model (M) that learns which behavioral patterns during the training Data Window (training-DW[1]) correlated to the account being labeled as suspicious later in the Label Window (LW). We introduce a Buffer Window (BW) between the DW and LW, to account for any lag (delay) in the suspicious account flagging pipeline used to generate the ground truth of suspicious accounts. The reason is that, in the absence of the BW, a lag in the pipeline will cause the fitted model to learn user behavioral patterns from accounts that are already under the control of an attacker. In Sect. 3.3, we present our heuristics to estimate the width of the Buffer Window (BW).

During *testing*, the fitted model (M) obtained during training, is applied during the *testing-DW* to forecast the set of accounts that are likely to have suspicious behaviour (P_{CE}). The quality of those predictions is then evaluated against the ground truth of labeled suspicious accounts extracted from the *testing-LW*.

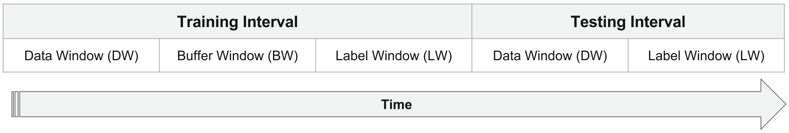

Fig. 1. Overview of a Classification Exercise (CE). Each exercise is divided into two broad phases: *training*, during which the classifier is fitted, and *testing*, during which the classifier predictions are evaluated. Each phase is subdivided into smaller non-overlapping time windows: *Data Window* (DW), *Buffer Window* (BW) and *Label Window* (LW). The DW is the period of time over which behavioural features are mined. The BW is a period of time introduced to avoid learning from accounts that may already be compromised but not yet labeled as such. The LW is the period over which labels are extracted.

3.2 The Early Warning Pipeline

Our system is composed of a pipeline that can be easily integrated into existing systems. We note that our pipeline design stresses efficiency, scalability, and, ultimately, achieving a practical training runtime sometimes even to the detriment of the learned classifiers (e.g., using simple low-cost features as opposed to sophisticated feature extraction). With production data, similar in scale to what we have access to at *LSOSP*, our pipeline is designed to extract behavioral patterns and to train in reasonable time on log traces from hundreds of millions of accounts leading to hundreds of billions of log entries over the duration of each CE. We developed our pipeline in Scala 2.11, employed SparkML for all our developed classifiers, and ran our CEs on Spark 2.0.2 [28].

Data Preprocessing. We preprocess the datasets from which we extract the user behavioral patterns (e.g., login activity dataset) as well as the ground truth (e.g., accounts flagged as suspicious). Importantly, *we also carry out a series of*

[1] Where the context makes the notation unambiguous, we skip the prefix and use *DW* only for *training-DW* or *testing-DW*. Similarly for *LW*.

pruning operations in order to exclude accounts that may bias either learning or evaluation as discussed in Sect. 3.3. During this stage, for each account, we extract features at the day level and aggregate them for the intervals associated with the classification exercise. There is an inherent trade-off here: extracting and computing a large number of features over a long duration of time could potentially include more behavioral information thereby increasing the prediction accuracy. However, this comes at the cost of longer runtime and might affect prediction timeliness. At *LSOSP*, we find that extracting only a relatively small set of low-cost features that are both simple and quick enough to compute is both sufficient and also more practical from a performance perspective in a production environment (details in Sect. 4).

Preprocessing Imbalanced Data. Typically at LSOSPs, suspicious accounts (the positive class) are a minority compared to the overall population. Naively training an ML classifier on such imbalanced data will typically result in a classifier that always predicts the dominant class (the negative class in our case) to achieve the highest accuracy [20]. Approaches to mitigate this problem include simple preprocessing techniques such as undersampling the majority class or oversampling the minority class [12], or Cost-Sensitive Learning [17] that attempts to minimize the cost of misclassifications by assigning asymmetrical costs during the training process. At LSOSP, given the scale of the data and our focus on building a practical pipeline with good balance between runtime and classification performance, we use undersampling during training (however, we test on the whole set of labeled data in the test set).

Classifier Tuning. Second, during the hyperparameter optimization stage, model selection is carried out in order to find the best model (or set of parameters) for the classification task. This only needs to be done once during training (or periodically, with low frequency and offline, to learn new user behavioral patterns) and is not carried out during inference using the fitted model in production. We use a Random Forest (RF) classifier considering the good trade-off it offers between runtime and classification accuracy [9]. We carry out the hyperparameter optimization on an independent dataset extracted from the available history and specifically reserved for this purpose (CE_A in Sect. 5). The extracted model parameters are then fixed for all the subsequent CEs.

Model Fitting and Inference. Third, after data preprocessing and hyperparameter tuning, a ML model M is fitted and later applied to make predictions on new data (i.e., inference). On the one hand, this data could be one for which there already exists labeled ground truth. In that case, the goal is to evaluate the performance of the developed classifier. On the other hand, this could be new data from production for which no ground truth exists (i.e., during the real-world deployment) and in this case, the goal is to put the classifier into practice to predict accounts likely to generate suspicious activity in the future based on their recent behavioral patterns.

Model Evaluation. Finally, we obtain the confusion matrix based on the resulting predictions and collect statistical measures of the classifier's performance.

3.3 Heuristics

Our goal is to learn behaviour from legitimate accounts (i.e., that are not attacker-controlled: fake and compromised accounts—$A_d\{d|d \in$ *Training Interval*$\}$) and predict which legitimate accounts may later get compromised and get labeled as suspicious. To this end we use a number of heuristics. We also implement additional heuristics to increase the confidence in our evaluation.

Heuristics to increase the chance that we capture only the behaviour of accounts under the control of legitimate users. During training, we attempt to exclude all accounts that are potentially under the control of an attacker. In practice the set of accounts A_d is unknown, even for historical data for which there is collected ground truth, as this set may include not-yet-detected fakes and compromised accounts. We take advantage of having an extremely large dataset to carry out aggressive exclusions that reduce the chance that we capture behaviour from attacker-controlled accounts. We use three heuristics:

- First, we exclude any account flagged as suspicious during the training DW or at a later point of time within the Buffer Window (BW). By excluding these accounts, we reduce the likelihood that our classifier learns behavioral patterns stemming from detected compromised accounts.
- Second, to the same end, for the classification exercises where there is available data before the start of the training interval (CE_C in Sect. 5), we exclude accounts flagged as suspicious before the start of training (as they are more likely to be compromised in the future).
- Finally, to eliminate fakes, one of our classification exercises (CE_C in Sect. 5) attempts to eliminate all recently-created or dormant fakes by selecting for training only accounts that are older than two months and have at least one month of activity (our assumption is that once fakes generate enough activity the LSOSP can detect them through existing techniques [25,27] as detecting fakes is easier than detecting compromised accounts [8]).

Heuristics to Reduce Bias During Classifier Evaluation. Our preliminary experiments suggest that user accounts that have been flagged as suspicious in the past are more likely to be flagged again in the future (a possible indication that their users are more vulnerable to attacks than the general user population). To provide a conservative (lower-bound) evaluation of the developed classifier's performance, we exclude all accounts that have been previously labeled as suspicious during training (i.e., flagged at any point during the training-LW or before). Moreover, one of our classification exercises (CE_C in Sect. 5), also excludes any accounts flagged as suspicious during the first month of the data collection. As a result, the classifier is evaluated on never seen before true positives.

Heuristics to Size the Buffer Window (BW). It is expected that, at any LSOSP, detection of suspicious activity is not instantaneous, thus accounts may be under the control of an attacker for a while before they are flagged. We developed an experiment to estimate how aggressive is *LSOSP*'s suspicious activity flagging pipeline. For this experiment, we only rely on two types of events: flagging events for accounts marked as suspicious on day d (extracted from set S_d) and login events for these accounts (extracted from set L_d). We include only user accounts that have at least one login event and at least one flagging event within the period of time over which we run the experiment. We define the *lag* per flagged user as the number of days between the first time that account is flagged and the most recent previous login event. Over a period of 30 days, the results showed that 90% of accounts flagged within that period have a lag of at most one week and 98.6% have a lag of less than three weeks. As such, we decided on a 1-week buffer window (BW) for most of our CEs, yet we also experimented with a 3-week BW (CE_D in Sect. 5).

Table 1. Summary of low-cost features. (from login traces)

Brief description	Type
# Login Attempts	Numeric
# Unique Login Sources (e.g., Web Login, Mobile Login, etc.)	
# Unique Login Types (e.g., Password Login, Account Switch, etc.)	
# Unique Login Statuses (e.g., Success, Session Extension, etc.)	
# Unique Password Login Statuses (e.g., Success, Invalid Password, etc.)	
# Unique Actions (e.g., Login/Logout, Device Authentication, etc.)	
# Unique Login Geographical Locations	
# Unique Login Geographical Location Statuses (e.g., Neutral Location, White-listed Location, etc.)	
# Unique Login Autonomous Systems (ASNs)	
# Unique Login User Agents (e.g., Browser, Mobile App, etc.)	
# Successful Logins	
# Unsuccessful Logins	
User has a "verified" mobile number	2-Categorical

4 Datasets

Overall, we had access to 118 days (≈4 months or ≈16 weeks) worth of production data collected from September 1st, 2016 to the December 27th, 2016 across two datasets which were updated daily. Overall, these datasets are representative of any LSOSP with a global user base, an extensive set of offered online services, as well as the latest techniques to identify potentially compromised accounts.

4.1 Extracting Features

The first dataset includes features associated with all login events. Whenever a user logs-in to a service offered by *LSOSP* or has their session re-authenticated, a login event is recorded into this dataset with all relevant features that can be associated with the event at that time. We use this dataset to extract a minimal set of 13 basic and easy to compute features that reflect users' login behavioral patterns (summarized in Table 1) from login traces at a day-level granularity, and then aggregate them for each user account as a way of characterizing its behavioral pattern over the DW. It is important to note that we do not have access to any fine-grained account features such as account/user details. Importantly, we do not have access to any personally identifiable information. Moreover, given the diversity of the login methods as well as the services offered at *LSOSP*, the features extracted for each login event are not uniform and the set of features extracted for each user is sparse.

4.2 Groundtruth: Suspicious Account Flagging

The second dataset includes events from which we extract our groundtruth. At *LSOSP*, a list of accounts flagged as suspicious is generated daily by combining information from various sources that include human content moderators, manual reports from internal teams, user reporting, in addition to automated systems employing heuristics (which include clustering techniques to identify anomalies, and regression models to identify spammers). We used this daily list of accounts flagged as suspicious as our ground truth.

For this study, we had access to this daily list of accounts flagged as suspicious and a high-level description of the system. The detailed internals of the flagging pipeline were not available. As a consequence, we are neither able to distinguish between the different classes of suspicious accounts nor to identify the reason why a particular account had been flagged. We believe that, the lack of such fine-grained information poses only limited threats to the validity of our findings: on the one side we have developed heuristics to exclude attacker-controlled accounts from training (see Sect. 3.3), and, on the other side, at this point our machine learning model aims to provide only predictive power (will an account be flagged as suspicious?) rather than explanatory power (why will the account be flagged?). We extend this discussion in Sect. 7.

5 Evaluation Results

The Objectives of our Classification Exercises. We present four of the classification exercises (CEs) carried out at *LSOSP* labeled CE_A, CE_B, CE_C, and CE_D in Table 2. The table outlines the Training and Testing intervals assigned to each CE and their respective Data Window (DW), Buffer Window (BW), and Label Window (LW). For each CE, we have a specific objective:

- CE_A: evaluating the feasibility of our proposed pipeline, its applicability at *LSOSP*, and optimizing hyperparameters.
- CE_B: testing the tuned model on new data to ensure that no overfitting occurred in CE_A.
- CE_C: investigating how the performance of our classifier changes when excluding accounts previously flagged as suspicious (higher chance to be flagged again) or accounts that have little previous activity (lower chance to include fakes).

- CE_D: evaluating the impact of more training data (longer data and label windows) and more aggressive exclusion of potentially not-yet-flagged attacker-controlled accounts (longer buffer window).

Summary of Results. Tables 3 and 4 summarize the results for all CEs carried out (their setup is outlined in Table 2). For conciseness, we focus here only on the most relevant metrics we collected. The two tables highlight how several metrics are impacted by the selected operating threshold T of the classifier as well as by the duration of the prediction horizon (presented as Test-LW and Extended-Test-LW in Table 2 and whose combined size in days is denoted as *the prediction horizon*: H). The tables present results for operating thresholds of $T = 0.5$ and $T = 0.9$ and prediction horizons of $H = 7, 21, 30, 34,\ and\ 90\ \ days$, in separate columns. Note that the minimum and maximum values of H depend on the CE.

In summary, these results show:

- High accuracy (ACC) ≈99.9% and low false positive rate (FPR) <0.01% for an operating threshold $T = 0.9$,
- Good evidence for the absence of overfitting (CE_B),
- Good balance between precision (PRE) and recall (REC): ≈18.33% and ≈50.62% respectively, when forecasting with a Horizon H = 30 days and Operating Threshold T = 0.5 (CE_C),
- A small improvement after excluding recent/no activity accounts (more likely to be fakes) and those flagged as suspicious before training (Comparing CE_B and CE_C),
- As the Horizon (H) increases, precision increases while recall stays roughly constant (We expand on this in Sect. 5.1),
- High AUC as shown in Fig. 2 (≈0.947 for CE_D), and
- More training data and a more aggressive exclusion of not-yet-flagged attacker-controlled accounts do not significantly impact classification performance (CE_D).

5.1 The Impact of the Prediction Horizon

Our classifier's precision markedly improves with the depth of the prediction horizon H (Fig. 3). Some of the accounts that are false positives for a small precision window then become true positives as the prediction window increases. We speculate that those accounts are owned by users that do not have the ability or the interest to fend off social engineering attacks, and thus a longer horizon increases the chance that they fall victim to an attack, and then generate suspicious activity which gets them flagged during the longer prediction horizon.

6 Related Work

Statistical methods (including ML) have achieved widespread adoption within LSOSPs not only to provide rich business features (e.g., product recommendations) but also for cybersecurity purposes. For instance, such approaches have

Table 2. Timeline of the four classification exercises (CEs) Performed CE_A, CE_B, CE_C, and CE_D. Notation: DW - Data Window, BW - Buffer Window, LW - Label Window, H - Prediction Horizon.

<table>
<tr><th rowspan="2">CE</th><th colspan="16">Week</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th></tr>
<tr><td rowspan="2"><b>A</b></td><td colspan="3">Train</td><td colspan="2">Test</td><td colspan="11">Extended Test</td></tr>
<tr><td>DW</td><td>BW</td><td>LW</td><td>DW</td><td>LW</td><td colspan="11">Extended LW <i>(H = [7, 90] days)</i></td></tr>
<tr><td rowspan="2"><b>B</b></td><td colspan="8">Unused</td><td colspan="3">Train</td><td colspan="2">Test</td><td colspan="3">Extended Test</td></tr>
<tr><td colspan="8">Unused</td><td>DW</td><td>BW</td><td>LW</td><td>DW</td><td>LW</td><td colspan="3">Extended LW</td></tr>
<tr><td rowspan="2"><b>C</b></td><td colspan="4">Preprocess</td><td colspan="4">Unused</td><td colspan="3">Train</td><td colspan="2">Test</td><td colspan="3">Extended Test</td></tr>
<tr><td colspan="4">Preprocess</td><td colspan="4">Unused</td><td>DW</td><td>BW</td><td>LW</td><td>DW</td><td>LW</td><td colspan="3">Extended LW</td></tr>
<tr><td rowspan="2"><b>D</b></td><td colspan="9">Train</td><td colspan="6">Test</td><td>Ext. Test</td></tr>
<tr><td colspan="3">DW</td><td colspan="3">BW</td><td colspan="3">LW</td><td colspan="3">DW</td><td colspan="3">LW</td><td>Ext. LW</td></tr>
</table>

Table 3. Summary of results using an operating threshold (T) = 0.5 for different prediction horizons (H days). Notation used: AUC-Area Under Receiver Operating Characteristic Curve, BTR-%-tile better than a random classifier, PRE-Precision, REC-Recall, ACC-Accuracy, FPR-False Positive Rate. Values in bold represent the best result for that performance metric.

CE			**Performance Evaluation Metrics**											
	H_{Min}	H_{Max}	$H=H_{Min}$		H=7				H=21		H=30		$H=H_{Max}$	
			AUC	BTR	PRE	REC	ACC	FPR	PRE	REC	PRE	REC	PRE	REC
A	7	90	0.928	85.61%	**6.38%**	**46.87%**	99.43%	0.52%	**19.79%**	45.81%	**20.14%**	43.81%	**24.99%**	31.02%
B	7	30	0.910	82.14%	3.78%	41.26%	**99.50%**	**0.46%**	18.18%	46.82%	19.98%	42.28%	19.98%	42.28%
C	7	30	0.922	84.42%	3.18%	42.96%	99.38%	0.58%	16.58%	57.32%	18.33%	**50.62%**	18.33%	**50.62%**
D	21	34	**0.947**	**89.41%**	$H < H_{Min}$				10.64%	**57.42%**	11.68%	48.96%	12.34%	48.13%

Table 4. Summary of results using an operating threshold (T) = 0.9 for different prediction horizons (H days). Notation used: AUC-Area Under Receiver Operating Characteristic Curve, BTR-%-tile better than a random classifier, PRE-Precision, REC-Recall, ACC-Accuracy, FPR-False Positive Rate. Values in bold represent the best result for that performance metric.

CE			**Performance Evaluation Metrics**									
	H_{Min}	H_{Max}	$H=H_{Min}$		$H=H_{Min}$				$H=H_{Max}$			
			AUC	BTR	PRE	REC	ACC	FPR	PRE	REC	ACC	FPR
A	7	90	0.928	85.61%	12.92%	0.47%	**99.92%**	**0.0024%**	33.99%	0.20%	99.54%	**0.0018%**
B	7	30	0.910	82.14%	7.11%	13.15%	99.88%	0.0760%	**35.96%**	12.90%	99.74%	0.0520%
C	7	30	0.922	84.42%	6.91%	**15.57%**	99.86%	0.0940%	35.33%	**16.29%**	99.75%	0.0650%
D	21	34	**0.947**	**89.41%**	**26.19%**	14.45%	99.86%	0.0430%	28.47%	11.36%	**99.82%**	0.0420%

been used for detecting compromised accounts, fake accounts, spam, and phishing. None of these approaches has focused on evaluating the feasibility of predicting which legitimate accounts are more vulnerable and likely to be compromised in the future (our long term aim). In this section each paragraph focuses on a specific area, surveys related approaches, and outlines the statistical methods and features used.

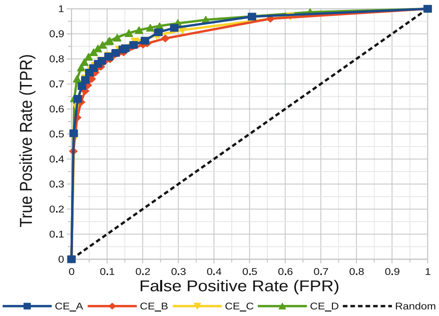

Fig. 2. ROCs for all classification exercises.

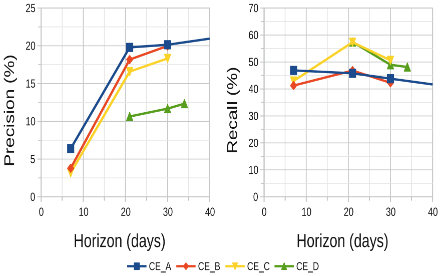

Fig. 3. Impact of the prediction horizon on precision (left) and recall (right) at operating threshold $T = 0.5$

Compromised Accounts. Egele et al. [8] combined statistical modeling and anomaly detection techniques in order to detect compromised accounts on Online Social Networks (OSNs). Their approach was based on identifying sudden changes in user behavioral patterns in addition to observing whether those changes are common to a large group of accounts therefore potentially a result of a malicious campaign. Thomas et al. [24] employed clustering and classification (via logistic regression) in order to detect account hijacking on Twitter. Their approach was based on the observation that legitimate account owners frequently delete tweets posted via their accounts after recognizing the compromise. Those deletions are thus used as a feature to retroactively identify hijacked accounts and clustering is then used to detect similarly compromised accounts. Zhang et al. [29] made use of a ML-based approach to automatically detect compromised accounts at a large academic institution. Their approach employed logistic regression on features extracted from web login and VPN authentication logs.

Fake Accounts. Yang et al. [27] proposed approaches to identify Sybil (i.e., fake) accounts on the Renren OSN. One approach was based on ML and employed Support Vector Machines (SVMs) on basic user-level features (e.g., the frequency of friendship requests and the fraction of accepted incoming friendship requests). Wang el al. [25] instead used clustering to identify fake accounts on Renren. Their approach clustered users with similar behavior based on features extracted from their clickstreams (e.g., the average session length, the average number of clicks per session).

Spam. Benevenuto et al. [2] developed an ML-based approach to identify spammers on Twitter. Their approach was based on a non-linear Support Vector Machine (SVM) classifier with the Radial Basis Function (RBF) kernel and made use of both content- and user-level features (e.g., the age of the user account, the number of followers, the average number of URLs per tweet). Castillo et al. [7] developed a ML-based approach using cost-sensitive decision trees to detect spam pages on the Web. Their approach makes use of content- and link-based features extracted from the Web graph (e.g., the ratio between the average degree of a page and that of its neighbours, number of words in the page/title). In the context of email spam, Blanzieri et al. [4] carried out a survey of many of the approaches to detect email spam proposed in the literature based on statistical methods (including ML).

Phishing. Ludl et al. [18] developed a ML-based approach to identify phishing web pages. Their approach was based on the C4.5 decision tree algorithm and made use of features extracted from a page's content as well as its URL (e.g., the number of forms/fields tags on the page, whether the page is served over HTTPS, whether the URL's domain appears on a Google whitelist). Whittaker et al. [26] developed a scalable ML-based approach to detect phishing websites that is used to maintain Google's phishing blacklist automatically. Their approach is based on a Random Forest (RF) classifier and employed both content-, host- and URL-based features (e.g., PageRank, the host geolocation/ASN).

7 Summary and Discussion

Summary. We explore the feasibility of predicting the legitimate (i.e., not attacker-controlled) accounts more likely to generate suspicious activity in the future, a likely indication that they have fallen for a mass-scale social engineering attack. To this end, we propose an early warning system that employs supervised machine learning to identify the accounts whose behavioral patterns indicate that they are similar to other accounts that have been eventually labeled as suspicious in the past. We implement this early warning system at a Large-Scale Online Service Platform (LSOSP) and evaluate it on four months of real-world production data covering hundreds of millions of users. Our evaluation demonstrates that our approach is not only feasible but that it also offers promising classification performance based on which further defense mechanisms can be developed as we discuss below.

Discussion. We continue by exploring several interrelated topics:
How can a defense system use information about which users are likely to be compromised in the future, and thus more "vulnerable", to enhance its robustness? User vulnerability can be thought of as an additional "signal" that can inform a number of defense mechanisms. For example, it can: *(i)* serve as an indicator to prioritize the allocation of limited defense resources (e.g., use of human analyst time [13], or compute-intensive filters [23]), *(ii)* support differentiated defenses that take into account user vulnerability (e.g., additional CAPTCHAs [1] on login attempts into vulnerable accounts, or imposing rate limits on the outbound messages of vulnerable users to slow-down the spread of multi-stage—and potentially epidemic—phishing attacks), *(iii)* enable faster remediation of compromised accounts (e.g., by enabling more efficient inspection campaigns that focus on the accounts of vulnerable users instead of the entire

user population [15]), *(iv)* facilitate the detection of the origin of an attack (as, in effect, the differentiated response between vulnerable and robust users to similar interactions initiated by the same source can be used as a weak yet effective signal [5]); and *(v)* even facilitate the detection of new attacks (as, in effect, the differentiated response between vulnerable and robust—yet otherwise similar—user groups to the same "stimuli" is an indication of an attack). We explore the use of such information for several cybersecurity domains in [10].

Is the prediction quality good enough? Even if defense mechanisms based on vulnerability predictions can be imagined, an immediate subsequent question is whether the classification quality implied by our results (e.g., $PRE \approx$ 15–25%, $REC \approx$ 40–50%, and $FPR \approx$ 0.1–0.5%) is good enough to support such mechanisms. While we have not yet extensively studied such mechanisms, our intuition is that this signal, although noisy, is useful. Consider, for example, defense resource prioritization - it is evident that a heuristic that uses this signal, as weak as it is, to prioritize those resources is better than randomly allocating them (the only alternative when capacity is constrained). Others have also experimented with a heuristic that harnesses the different responses to similar requests between vulnerable and robust users [5] to infer attack source(s) (although in the context of a social network). In this case, even a vulnerability predictor significantly weaker than the one we have obtained here has proven useful, leading to a technique that improves over the state-of-the-art. While the above indicates that even low quality predictions can still be used to improve defenses, we believe that the prediction quality threshold above which these mechanisms become valuable is context specific and we are studying this issue in a related project [10].

Why do we focus on minimizing the false positive rate (FPR)? What if the focus were on maximizing recall instead? We envisage that the predictions made by our early warning system will be used to better target existing defenses. As many of these defenses are not lightweight and may lead to increased friction for users (e.g., rate-limiting outbound emails of vulnerable users to prevent an attack outbreak, delaying incoming suspicious email addressed to vulnerable users to give enough time for more robust users to report mass-phishing emails), or allocating costly resources (e.g., human analyst time), the resulting cost of false positives is high: thus, we have focused on minimizing the FPR at the expense of lower recall. Other situations, however, offer a different cost/benefit balance between the false positive rate and recall. For these situations, our classifier can be tuned by either using lower threshold values (T as highlighted by the ROC across all CEs available in Fig. 2), or by specifically optimizing for recall.

What are the threats to validity? Our study indicates that it is feasible to harness account behaviour to predict the accounts that are more likely to generate suspicious traffic in the future (an indicator that they may be compromised). There are two main concerns regarding the validity of our conclusions. The first one relates to the quality of the ground truth we use—this is a threat to validity common to any study using a methodology based on machine learning.

The second one relates to the accuracy of the heuristics used to avoid learning behavioural patterns from accounts that may be controlled by an attacker (i.e., compromised or fake accounts) detailed in Sect. 3.3. We prune: *(i)* all accounts flagged for suspicious activity in the data window (DW) - as they are highly likely to be compromised, *(ii)* all accounts flagged as suspicious in the buffer window (BW) - as these accounts are more likely to have been compromised but not yet flagged as such (thus contaminating our training data), *(iii)* all accounts which have been labeled as suspicious at any point *before* the training data window - as our experience shows that these accounts are more likely to be compromised again (in experiment CE_C); and, finally *(iv)* new/low activity accounts (for which the system may not have enough history to determine whether the accounts are fakes). We run various experiments that compare the impact of these heuristics - even the most conservative experiments appear to support our conclusions.

It is worth discussing, however, the alternative: assume that our heuristics fail to eliminate a large portion of attacker controlled accounts. Even in this case, we believe that our pipeline provides value through forecasting. Assume, for example, that these accounts are predominantly (dormant) fakes that mimic legitimate user behaviour. In this case, our pipeline predicts the fakes that will likely be "awakened" by the attacker and start generating suspicious activity. Assume, on the other side, that these are compromised accounts not yet exploited by the attacker, then our pipeline predicts which compromised accounts are under the control of the attacker but not yet exploited. In this case as well the forecasting pipeline can give an early sign of the attacker resources and strategy.

A final concern may be that our proposed approach may be learning the heuristics by which some accounts are flagged as suspicious in the ground truth (other accounts in the ground truth are flagged by humans). We believe that this represents a limited threat due to the way we formulated our forecasting problem (i.e., making future predictions) as opposed to the underlying heuristics which operate in real-time by design.

Why are the presented results positioned as lower-bounds? Our goal was to test the feasibility of our proposed approach within constraints related to:

- *Access to Data (i.e., login traces only).* Datasets with additional information that characterizes user behaviour (e.g., email or browsing traces) would likely improve classification performance.
- *Limited Computational Resources (i.e, runtime feasibility for processing billions of events).* More resources enabling additional data preprocessing (e.g., to extract complex aggregate features), model optimization, or sophisticated learning methods (e.g., deep neural networks) would likely improve classification performance.
- *Imperfect Ground Truth (i.e., detection lag as well as the presence of false positives and false negatives).* This impairs the learned models during training, and impacts the evaluation during testing.

- *Aggressive Pruning Heuristics (i.e., extensive pruning of accounts during training as described in* Sect. 3.3*).* This reduces bias during the evaluation of the classifier but leads to more conservative results.

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_18
2. Benevenuto, F., Magno, G., Rodrigues, T., Almeida, V.: Detecting spammers on twitter. In: Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS), vol. 6, p. 12 (2010)
3. Bilge, L., Han, Y., Dell'Amico, M.: Riskteller: predicting the risk of cyber incidents. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, pp. 1299–1311. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134022, https://doi.acm.org/10.1145/3133956.3134022
4. Blanzieri, E., Bryl, A.: A survey of learning-based techniques of email spam filtering. Artif. Intell. Rev. **29**(1), 63–92 (2008). https://doi.org/10.1007/s10462-009-9109-6. https://dx.doi.org/10.1007/s10462-009-9109-6
5. Boshmaf, Y., et al.: Integro: leveraging victim prediction for robust fake account detection in OSNs. In: 22nd Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA, 8–11 February 2015, pp. 1–15. http://www.internetsociety.org/doc/integro-leveraging-victim-prediction-robust-fake-account-detection-osns
6. Canali, D., Bilge, L., Balzarotti, D.: On the effectiveness of risk prediction based on users browsing behavior. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS 2014, pp. 171–182. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2590296.2590347, https://doi.acm.org/10.1145/2590296.2590347
7. Castillo, C., Donato, D., Gionis, A., Murdock, V., Silvestri, F.: Know your neighbors: web spam detection using the web topology. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, pp. 423–430. ACM, New York, NY, USA (2007). https://doi.org/10.1145/1277741.1277814, https://doi.acm.org/10.1145/1277741.1277814
8. Egele, M., Stringhini, G., Kruegel, C., Vigna, G.: COMPA: detecting compromised accounts on social networks. In: Proceedings of the Network & Distributed System Security Symposium, NDSS 2013, ISOC, February 2013
9. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**(1), 3133–3181 (2014). http://dl.acm.org/citation.cfm?id=2627435.2697065
10. Halawa, H., Beznosov, K., Boshmaf, Y., Coskun, B., Ripeanu, M., Santos-Neto, E.: Harvesting the low-hanging fruits: defending against automated large-scale cyber-intrusions by focusing on the vulnerable population. In: Proceedings of the 2016 New Security Paradigms Workshop, NSPW 2016, pp. 11–22. ACM, New York, NY, USA (2016). https://doi.org/10.1145/3011883.3011885, https://doi.acm.org/10.1145/3011883.3011885

11. Halawa, H., Ripeanu, M., Beznosov, K., Coskun, B., Liu, M.: Forecasting suspicious account activity at large-scale online service providers. CoRR abs/1801.08629 (2018). http://arxiv.org/abs/1801.08629
12. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009). https://doi.org/10.1109/TKDE.2008.239
13. Ho, G., Javed, A.S.M., Paxson, V., Wagner, D.: Detecting credential spearphishing attacks in enterprise settings. In: Proceedings of the 26rd USENIX Security Symposium, USENIX Security 2017, pp. 469–485 (2017)
14. Jagatic, T.N., Johnson, N.A., Jakobsson, M., Menczer, F.: Social phishing. Commun. ACM **50**(10), 94–100 (2007)
15. Liu, G., Xiang, G., Pendleton, B.A., Hong, J.I., Liu, W.: Smartening the crowds: computational techniques for improving human verification to fight phishing scams. In: Proceedings of the Seventh Symposium on Usable Privacy and Security, SOUPS 2011, pp. 8:1–8:13. ACM, New York, NY, USA (2011). https://doi.org/10.1145/2078827.2078838, https://doi.acm.org/10.1145/2078827.2078838
16. Liu, Y., et al.: Cloudy with a chance of breach: forecasting cyber security incidents. In: Proceedings of the 24th USENIX Security Symposium, USENIX Security 2015, pp. 1009–1024 (2015)
17. Lomax, S., Vadera, S.: A survey of cost-sensitive decision tree induction algorithms. ACM Comput. Surv. **45**(2), 16:1–16:35 (2013). https://doi.org/10.1145/2431211.2431215. https://doi.acm.org/10.1145/2431211.2431215
18. Ludl, C., McAllister, S., Kirda, E., Kruegel, C.: On the effectiveness of techniques to detect phishing sites. In: M. Hämmerli, B., Sommer, R. (eds.) DIMVA 2007. LNCS, vol. 4579, pp. 20–39. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73614-1_2
19. Moore, T., Clayton, R., Anderson, R.: The economics of online crime. J. Econ. Perspect. **23**(3), 3–20 (2009). https://doi.org/10.1257/jep.23.3.3. https://www.aeaweb.org/articles/?doi=10.1257/jep.23.3.3
20. Provost, F., Fawcett, T.: Robust classification for imprecise environments. Mach. Learn. **42**(3), 203–231 (2001). https://doi.org/10.1023/A:1007601015854. https://dx.doi.org/10.1023/A:1007601015854
21. Shon, T., Moon, J.: A hybrid machine learning approach to network anomaly detection. Inf. Sci. **177**(18), 3799–3821 (2007)
22. Soska, K., Christin, N.: Automatically detecting vulnerable websites before they turn malicious. In: Proceedings of the 23rd USENIX Security Symposium, USENIX Security 2014, pp. 625–640 (2014)
23. Stein, T., Chen, E., Mangla, K.: Facebook immune system. In: Proceedings of the 4th Workshop on Social Network Systems, SNS 2011, pp. 8:1–8:8. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1989656.1989664. https://doi.acm.org/10.1145/1989656.1989664
24. Thomas, K., Li, F., Grier, C., Paxson, V.: Consequences of connectivity: characterizing account hijacking on twitter. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, pp. 489–500. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2660267.2660282. https://doi.acm.org/10.1145/2660267.2660282
25. Wang, G., Konolige, T., Wilson, C., Wang, X., Zheng, H., Zhao, B.Y.: You are how you click: clickstream analysis for sybil detection. In: Proceedings of the 22Nd USENIX Conference on Security, SEC 2013, pp. 241–256. USENIX Association, Berkeley, CA, USA (2013). http://dl.acm.org/citation.cfm?id=2534766.2534788

26. Whittaker, C., Ryner, B., Nazif, M.: Large-scale automatic classification of phishing pages. In: Proceedings of the 17th Annual Network and Distributed System Security Symposium, NDSS Symposium 2010, San Diego, CA, USA (2010)
27. Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B.Y., Dai, Y.: Uncovering social network sybils in the wild. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC 2011, pp. 259–268. ACM, New York, NY, USA (2011). https://doi.org/10.1145/2068816.2068841. https://doi.acm.org/10.1145/2068816.2068841
28. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud 2010, p. 10. USENIX Association, Berkeley, CA, USA (2010). http://dl.acm.org/citation.cfm?id=1863103.1863113
29. Zhang, J., et al.: Safeguarding academic accounts and resources with the university credential abuse auditing system. In: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), pp. 1–8, June 2012. https://doi.org/10.1109/DSN.2012.6263961