Who Are You? A Statistical Approach to Measuring User Authenticity

David Mandell Freeman
and Sakshi Jain
LinkedIn Corporation
{dfreeman,sjain2}@linkedin.com

Markus Dürmuth
Ruhr-Universität Bochum
markus.duermuth@rub.de

Battista Biggio
and Giorgio Giacinto
Università di Cagliari
{battista.biggio,giacinto}@diee.unica.it

***Abstract*—Passwords are used for user authentication by almost every Internet service today, despite a number of well-known weaknesses. Numerous attempts to replace passwords have failed, in part because changing users' behavior has proven to be difficult. One approach to strengthening password-based authentication without changing user experience is to classify login attempts into normal and suspicious activity based on a number of parameters such as source IP, geo-location, browser configuration, and time of day. For the suspicious attempts, the service can then require additional verification, e.g., by an additional phone-based authentication step. Systems working along these principles have been deployed by a number of Internet services but have never been studied publicly. In this work, we perform the first public evaluation of a classification system for user authentication. In particular:**

(i) We develop a statistical framework for identifying suspicious login attempts.
(ii) We develop a fully functional prototype implementation that can be evaluated efficiently on large datasets.
(iii) We validate our system on a sample of real-life login data from LinkedIn as well as simulated attacks, and demonstrate that a majority of attacks can be prevented by imposing additional verification steps on only a small fraction of users.
(iv) We provide a systematic study of possible attackers against such a system, including attackers targeting the classifier itself.

I. Introduction

Almost every Internet service today authenticates its users using passwords: each user is associated with a short block of text that is supposedly known only to that user. Advantages to this system are that passwords are nearly universally understood by users and that they are well supported by current infrastructures. However, passwords in practice have numerous security flaws that have been well documented in both the research literature and the popular press: users choose simple and/or easy-to-guess passwords; users reuse passwords across services, meaning that a compromise of accounts on one service leads to compromise of accounts on many other services; users are often tricked into revealing their passwords to attackers (e.g., via "phishing"); and modern password-cracking tools have become very powerful—the best has been reported to guess up to 2.7 billion passwords per second on a single GPU [52].

Many innovative techniques have been proposed to deal with these problems, and several have been implemented in practice. One common proposal is *two-factor authentication*, by which the user must confirm that she has possession of another credential linked to the account. This second factor is typically a hardware token, an authentication app (e.g. [30]), or, with reduced security benefits, a mobile phone number or an email address. Most major websites (e.g. Google, Facebook, LinkedIn, and Twitter) now offer a two-factor authentication solution. However, two-factor authentication, being an opt-in process, suffers from low adoption rates and does little to thwart a large-scale attack on an Internet service.

Biometric authentication techniques, including fingerprint and face recognition [39], [34], and typing dynamics [28], [41], [32], [19], have also been investigated as an alternative to password-based authentication, but limited performance on very large numbers of users and risks for privacy leaks have actually slowed down its adoption in large online services.

Instead of replacing passwords, more recently there has been significant effort to increase the security of password-based authentication. Examples include several methods for increasing the entropy of users' passwords [35], [33], [11] as well as methods for discouraging reuse across websites [35], [27]. Most of these proposals (discussed in more detail in Sect. VII) require changing user behavior, and to date none has achieved widespread adoption.

Given the difficulty of changing users' behavior, in practice one must assume that any password can easily fall into the hands of an attacker. Many websites thus use a different approach: to impose extra friction on authentication attempts they determine to be suspicious. For example, between entering a correct password and proceeding into the site, a service can require a user to solve a CAPTCHA,[1] verify an email address, receive an SMS message, or answer security questions. For maximum security a site could pose such "challenges" on every login attempt; however, this level of friction would be highly detrimental to the site's level of engagement, as a large

NDSS '16, 21-24 February 2016, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2016.23240

[1] A CAPTCHA, i.e., a *Completely Automated Public Turing test to tell Computers and Humans Apart*, is a challenge-response test used to verify whether the user is a human being or a bot.

percentage of legitimate users will be unwilling or unable to solve these challenges. There is thus a need to classify the level of suspiciousness of any authentication attempt and only block or challenge the most suspicious.

The classification of login attempts (e.g., into three tiers good/suspicious/bad) is derived from the data available at the time of the login, including source IP address, geo-location, operating system and browser configuration, the account's patterns of usage, and more. However, the details of these derivations are generally not disclosed by the sites employing such classifiers. Consequently, any public evaluation and comparison is missing. In addition, these defenses have not been tested for their resistance to attacks targeting the classifiers themselves, nor does there exist a systematic treatment of attacks against such a login service. The lack of public benchmarks, and even any baseline performance, makes it difficult to compare and evaluate systems.

Our contribution. In this work we provide the first public study of the classification of login attempts, which we call *reinforced authentication.* We advocate a "security by design" approach, avoiding basing the security of the classifier on the secrecy of its design. This is important because recent work on the behavior of classifiers in the presence of attacks (so-called *adversarial machine learning*) has shown that the state of a classifier can be reconstructed by repeatedly querying the classifier, questioning the validity of an approach based on "security by obscurity" [3], [5], [6], [31]. These results can also be used to harden classifiers against these attacks, without requiring secrecy of the design.

More concretely, our contributions are as follows:

- We develop a statistical framework for identifying suspicious authentication attempts. We begin with a likelihood ratio test and show how the desired ratio can be approximated in the presence of sparse data. Our main formula (7) can be computed even in the absence of labeled account takeover data. When such labeled data is present, we can use the terms in the formula as features in a logistic regression model to achieve greater accuracy.
- We develop a prototype implementation of the described system. The main contribution here is smoothing techniques to handle cases where there is no data, such as when a user logs in from a previously unseen IP address. Our solution supports the intuition that the level of suspiciousness should increase as the user gets farther away from a previously seen location. We demonstrate that the proposed system is efficient enough for a practical large-scale deployment.
- We validate our proposal on a sample of real-life user login data from LinkedIn. When using six months of history and only two features (IP address and useragent), we achieve AUC 0.96 and find that the most suspicious 10% of login attempts include 89% of labeled account takeover attempts.
- Inspired by recent work in the area of adversarial machine learning [3], [5], [6], [31], we classify potential attacks against such a classifier, taking into account the specific challenges of an authentication system. This classification allows us to assess our system's security by hypothesizing potential attack scenarios and simulating attacks. In a number of experiments, we demonstrate the efficiency of various feature combinations and evaluate the effectiveness of attackers trying to evade the classifier.

This work is not intended to replace alternative authentication methods such as two-factor authentication, but is orthogonal and can be applied, in principle, to any kind of login procedure. We expect it to be particularly helpful in securing the large majority of accounts that are unable or unwilling to use stronger authentication mechanisms.

II. Reinforced Authentication

The underlying idea behind reinforcing user authentication is to exploit complementary information beyond the validation of user credentials. This information can be extracted from the HTTP session logs of authentication sessions established between the user and the web server (including, e.g., the IP address, useragent, timestamp, and cookies) and then compared against data available from the user's login history through a carefully designed statistical machine-learning approach. The principal result of this section is Eq. (7), which gives a scoring function that can be computed using only per-user and global login history and asset reputation systems; in particular, labeled account compromise data is not required for the basic scoring function.

Let us denote with $u \in \mathcal{U}$ a given *user account*, with $\boldsymbol{x} = (x^1, \ldots, x^d) \in \mathcal{X}$ a d-dimensional set of *feature values* characterizing a login attempt (e.g., timestamp, IP address, browser, etc.), and with $y \in \mathcal{Y} = \{\mathrm{L}, \mathrm{A}\}$ the class label of legitimate login (L) or attacks (A). In the sequel, uppercase letters will be used to denote random variables (r.v.), and lowercase letters to denote the corresponding realizations; for example, if the r.v. X denotes the "IP address", then x will correspond to a specific IP address (e.g., 127.0.0.1). We assume that for each account, login samples of either class are generated according to an underlying (though unknown) probability distribution $p(\boldsymbol{X}, U, Y)$, for which we are only given a set of (i.i.d.) samples $\mathcal{D} = \{\boldsymbol{x}_i, u_i, y_i\}_{i=1}^n$ representing the available login history for each user. We also assume that in all cases the provided credentials are correct. If the provided credentials are wrong, then the login attempt is rejected regardless of the output of the reinforced authentication module.

Given this notation, reinforced authentication can be formulated as the task of learning a classification function $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{Y}$ that, for each set of features $\boldsymbol{x}$ and user u, accurately predicts whether the corresponding login attempt is a *legitimate* login, or an *account takeover*. For compactness, we will denote this function as $f_u(\boldsymbol{x}) \in \{\mathrm{L}, \mathrm{A}\}$.

If the underlying distribution p were known, the decision function yielding the minimum probability of wrong predictions (i.e., minimizing *generalization error* or *risk*) would be given by the so-called Maximum-A-Posteriori (MAP) criterion:

$$\arg\max_{y \in \mathcal{Y}} p(Y = y | \boldsymbol{X} = \boldsymbol{x}, U = u)\,. \tag{1}$$

For two classes, the MAP criterion amounts to deciding for

$Y = \mathrm{A}$ if

$$\frac{p(Y = \mathrm{A}|\boldsymbol{X} = \boldsymbol{x}, U = u)}{p(Y = \mathrm{L}|\boldsymbol{X} = \boldsymbol{x}, U = u)} > 1, \tag{2}$$

and for $Y = \mathrm{L}$ otherwise. Applying Bayes' Theorem to the numerator and denominator of (2) gives

$$\frac{p(Y = \mathrm{A}|\boldsymbol{X} = \boldsymbol{x}, U = u)}{p(Y = \mathrm{L}|\boldsymbol{X} = \boldsymbol{x}, U = u)} = \frac{p(\boldsymbol{x}, u|Y = \mathrm{A})}{p(\boldsymbol{x}, u|Y = \mathrm{L})} \frac{p(Y = \mathrm{A})}{p(Y = \mathrm{L})}, \tag{3}$$

and noting that the prior probabilities are not dependent on $\boldsymbol{x}$ and u, the MAP criterion becomes

$$\underbrace{\frac{p(\boldsymbol{x}, u|Y = \mathrm{A})}{p(\boldsymbol{x}, u|Y = \mathrm{L})}}_{g_u(\boldsymbol{x})} \lessgtr \underbrace{\frac{p(Y = \mathrm{L})}{p(Y = \mathrm{A})}}_{\theta}; \tag{4}$$

that is, decide for class $Y = \mathrm{A}$ if $g_u(\boldsymbol{x}) > \theta$, and for $Y = \mathrm{L}$ otherwise. The threshold θ can be generally adjusted on a validation set to balance the trade-off between the rate of misclassified legitimate logins (false positives, FP) and the attack detection rate (true positives, TP). This rule is is widely-known in biometric identity verification as *likelihood ratio*, or *Neyman-Pearson criterion* [43]. According to the Neyman-Pearson lemma, for a given FP rate, one can select θ such that the likelihood ratio test maximizes the detection rate (TP). If the probability distributions $p(\mathrm{A}|\boldsymbol{x}, u)$ and $p(\mathrm{L}|\boldsymbol{x}, u)$ are exactly known, this rule is optimal, i.e., it yields the maximum detection rate (and no other test outperforms this one in terms of statistical power). However, in practice the aforementioned probability distributions are not known, and they have to be estimated from the available data $\mathcal{D}$. The performance of the likelihood ratio test will thus depend on how well such distributions are estimated.

Finally, if the confidence score $g_u(\boldsymbol{x})$ is too close to the threshold, i.e., the prediction is not very confident, then one may consider requesting additional information from the user to validate the login. For example, we may request the user to verify a phone number or email address or to provide some personal information that can be validated by the system (e.g., date of birth). In this case, one should consider two different thresholds $\theta^{\mathrm{u}} > \theta^{\mathrm{d}}$ such that:

$$f_u(\boldsymbol{x}) = \begin{cases} \mathrm{A}\,, \text{ if } g_u(\boldsymbol{x}) > \theta^{\mathrm{u}}\,, \\ \mathrm{F}\,, \text{ if } \theta^{\mathrm{d}} \leq g_u(\boldsymbol{x}) \leq \theta^{\mathrm{u}}\,, \\ \mathrm{L}\,, \text{ if } g_u(\boldsymbol{x}) < \theta^{\mathrm{d}}\,, \end{cases} \tag{5}$$

where F represents the case in which we should request further information. This scenario is usually referred to as *classification with reject option* [20].

A. Working Assumptions

As our system has to deal with large amounts of *sparse* data, i.e., a large number of login attempts per user with different, rare combinations of features, like IP address and useragent, we make here some simplifying assumptions to keep the computational complexity of our approach manageable, and allow exploiting information provided from third-party services, e.g., IP reputation systems.

First, by noting that $p(\boldsymbol{x}, u|y) = p(\boldsymbol{x}|u, y)p(u|y)$, the left-hand-side term of Eq. 4 can be rewritten as:

$$g_u(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|u, \mathrm{A})}{p(\boldsymbol{x}|u, \mathrm{L})} \frac{p(u|\mathrm{A})}{p(u|\mathrm{L})}. \tag{6}$$

Since data is sparse, we have few samples for a given set of feature values conditioned to each class; e.g., observing the same pair of IP and useragent for a given user is a rare event, even if the login is legitimate. Thus, providing a reliable estimate of the term $p(\boldsymbol{x}|u, y)$ for either class is not always possible. To overcome this limitation, we assume independence among features, i.e., $p(\boldsymbol{x}) = \prod_{k=1}^{d} p(x^k)$. This is a well-known practice when estimating multivariate probability distributions from sparse data, e.g., for text categorization and spam filtering [49]. Furthermore, we assume that, in an attack attempt, the identity u of the user being attacked is *independent* of the value of the feature set $\boldsymbol{x}$ used by the attacker, i.e., $p(\boldsymbol{x}|u, \mathrm{A}) = p(\boldsymbol{x}|\mathrm{A})$. This is reasonable in *indiscriminate* attacks (i.e., attacks not targeting a specific user), as we can assume that the attacker will not be able to determine the values of $\boldsymbol{x}$ used by different users and adjust the values $(\boldsymbol{x}, u)$ accordingly.

According to the aforementioned assumptions, we can write $p(\boldsymbol{x}|u, \mathrm{A}) = \prod_{k=1}^{d} p(x^k|\mathrm{A})$ and similarly for $p(\boldsymbol{x}|u, \mathrm{L})$. Furthermore, by Bayes' Theorem we can substitute $p(x^k|\mathrm{A}) = p(\mathrm{A}|x^k)p(x^k)/p(\mathrm{A})$. This allows us to exploit feature reputation systems to estimate the term $p(x^k|\mathrm{A})$; e.g., an external reputation system that labels certain IP addresses as belonging to botnets, or certain useragents as being command-line tools. Then, by disregarding the terms dependent on $p(\mathrm{A})$ (which can be compensated for by adjusting the decision threshold θ), one yields:

$$g_u(\boldsymbol{x}) = \left(\prod_{k=1}^{d} p(\mathrm{A}|x^k) \frac{p(x^k)}{p(x^k|u, \mathrm{L})} \right) \frac{p(u|\mathrm{A})}{p(u|\mathrm{L})}. \tag{7}$$

B. Probability Estimation

Eq. 7 gives a formula that we can use to compute a score $g_u(\boldsymbol{x})$ for any given login attempt. We now explain how to estimate each term of $g_u(\boldsymbol{x})$ from data.

$p(\mathrm{A}|x^k)$. Ideally, one may obtain this probability for every feature value x^k from the training data. In practice, there are not sufficient examples of attacks for each x^k, so we can use *reputation systems* as proxies. These may be built internally from sitewide abuse data (i.e., not just account takeover attempts) or may be sourced from a third party (see, e.g., [54]). An IP reputation system will assign a "reputation score" to each IP address that indicates how likely the IP address is to be abusive. The reputation score can then be normalized to yield a suitable probability estimate. The system can also use the reputation of the Internet Service Provider (ISP) that the IP belongs to, to improve the reputation score. A similar construction can be applied to other fine-grained features.

$p(u|\mathrm{A})$. This is the probability that user u is attacked, given that we observed an attack. Since we may not have enough data to estimate this probability in a reliable manner, it may be entirely reasonable to assume that all users are equally likely to have their password compromised; e.g., this could be the case in an attack using a username/password list from a third-party site. A more sophisticated approach may consist of estimating which users are most valuable to attackers, i.e., most likely to be targeted by an attack. In a social network signals like

connections, followers, and amount of content posted could be used to inform such a model.

$p(u|\mathrm{L})$. This is simply the probability that user u is logging in to the site, given that the login is legitimate. We can take the proportion of all legitimate logins that belong to user u, possibly with some time decay.

$p(x^k), p(x^k|u, \mathrm{L})$. As a basic estimate we can compute the proportion of times in history that we have seen x^k as the value of feature X^k; this computation can be applied globally or on a per-user, class-conditional basis. For a more sophisticated model we can discount older events, e.g., using an exponential decay model.

We note that under the above assumptions, we do not directly use any labeled account compromise events in the computation of $g_u(\boldsymbol{x})$ beyond what is input to the reputation scores. This property could be useful to a new service that is acquiring users quickly and does not have a reliable system of labeling compromised accounts; such a service would also be likely to use external reputation data.

C. Smoothing

As we deal with sparse data, it is likely that some variable X may never take on a specific value x; for example, a user may log in from an IP address that we have not observed in that user's history or even in the global history. The corresponding maximum likelihood estimates (MLEs) of $p(x|u, \mathrm{L})$ and $p(x)$ (i.e., their relative frequencies) will thus be zero, yielding an undefined value for Eq. (7). To overcome this limitation, we exploit a well-known technique known as *smoothing*, originally proposed to deal with probability estimation in n-gram language models and, more generally, in the presence of sparse data [14].

The underlying idea of smoothing is to decrease the probability mass of known events in order to reserve some probability to unseen events. To this end, we adjust our estimates as:

$$p_0(x) = \begin{cases} \frac{c(x)}{N}\left(1 - \frac{M}{N+M}\right) & \text{if } c(x) > 0\,, \\ \frac{1}{N+M} & \text{otherwise}\,, \end{cases} \tag{8}$$

where $c(x)$ is the number of times we have seen x in our history and $N = \sum_x c(x)$ is the total number of logins (events). Essentially, we discount the ML estimate $c(x)/N$ by adding M "unseen" events and giving each of them probability mass $1/(N+M)$. For instance, if we have seen $N = 9$ logins, each from a different IP address, and we assume $M = 3$ unseen IPs, then the *smoothed* probability p_0 of logging in from an unknown IP will be $1/12$, equal to the probability of logging in from any other known IP (instead of having 0 and $1/9$ for the unseen and known IPs, respectively).

However, we would like to use the fact that IP addresses are aggreated into larger groupings such as ISP and country to give higher probabilities to *unseen* IPs that come from *known* ISPs or countries, rather than considering all of them equally likely. Accordingly, we can estimate $p(x)$ as:

$$p_k(x) = \sum_{h'_k \in \mathcal{H}_k} p(x|h'_k)p(h'_k) = p(x|h_k)p(h_k)\,, \tag{9}$$

where h'_k represents a conditioning event, taking values in $\mathcal{H}_k$. The index k corresponds to the *level* of the observation, with higher values denoting more granular levels; e.g., level $k = 1$ may correspond to countries, and level $k = 2$ to ISPs. The level $k = 0$, discussed above, involves all login attempts, and we thus name it the *world* level. In addition, we create a top level ℓ ($\ell = 3$ in our example) representing the IP itself, in which case $h_\ell = x$. Notice that, if IP x belongs to ISP (or country) h_k, the probability of seeing x coming from a different ISP (or country) h'_k is zero, and this is why the marginalization in (9) yields $p_k(x) = p(x|h_k)p(h_k)$.

We can now smooth $p(x|h_k)$ by defining $p_k(x|h_k)$ analogously to Eq. (8), replacing N by the number of logins N_{h_k} seen from the ISP (or country) h_k, and M by the number M_{h_k} of unseen IPs from ISP (or country) h_k. The ML estimate of $p(h_k)$ is kept unsmoothed, and it is thus zero for unseen ISPs (or countries).[2] In this way, higher probabilities are assigned to unseen IPs coming from known ISPs or countries; Fig. 1 provides a concrete example.

In order to have a consistent number of unseen IPs at each level k, for $0 \le k < \ell$ we recursively define

$$M_{h_k} = \sum_{h_{k+1} \in h_k} M_{h_{k+1}} + \mu_{h_k}\,,$$

where μ_{h_k} is the number of unseen IPs belonging to no known sub-entity of h_k. (To set the base case for the recursion we define $M_{h_\ell} = 0$ for all IPs h_ℓ, as there are no unseen IPs belonging to an IP.) If we assume one unseen IP for each ISP, one unseen ISP per country, and one unseen country (i.e., $\mu_{h_k} = 1$ for any h_k) then the world-level M will be equal to the number of known ISPs plus the number of known countries, plus one. A similar strategy can be considered for browsers' useragents, considering higher-level groupings like operating system, browser family, and the corresponding versions.

According to the aforementioned procedure, we can compute estimates of $p(x)$ at different levels k, depending on the conditioning event h_k at level k. We then consider two distinct procedures to aggregate these different estimates, namely, *backoff* and *interpolation*, inspired from the homonymous techniques for n-gram probability estimation [14].

Backoff. The basic idea for backoff smoothing is that if there are no observations of an IP, we "back off" by using data from the ISP to which the IP belongs; if there are no observations of that ISP we back off to the country, etc. Technically, we combine the $\ell+1$ probability estimates $p_k(x)$, for $k = 0, \ldots, \ell$ as follows:

$$p_{\mathrm{bo}}(x) = \begin{cases} \alpha_\ell p_\ell(x) & \text{if } c(h_\ell) > 0\,, \\ \alpha_{\ell-1} p_{\ell-1}(x) & \text{if } c(h_{\ell-1}) > 0\,, \\ \cdots & \\ \alpha_0 p_0(x) & \text{otherwise.} \end{cases} \tag{10}$$

Since granularity increases with k (e.g., for IP addresses $k = 0$ is the world level $k = \ell$ is the IP level), Eq. (10) tells us to use the estimate from the most granular level for

[2] Note that, if smoothing is not applied, all the estimates $p_k(x)$ will be equal to the ML estimate $p(x)$.

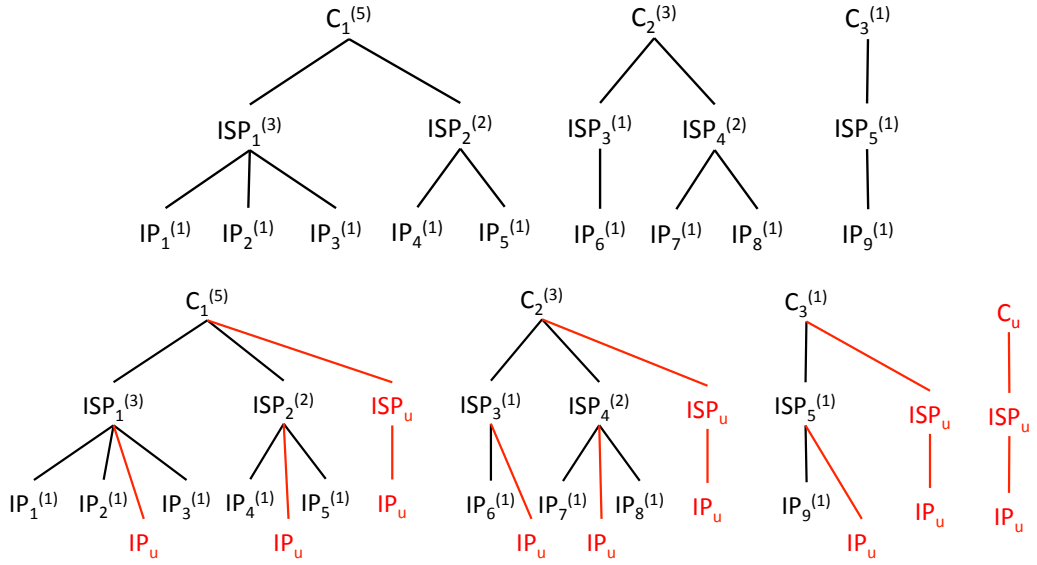


Fig. 1: An example of smoothing probability estimates for unseen IPs, involving 3 countries (C), 5 ISPs (ISP), and 9 known IPs (IP). Unseen events are shown in red. The counts $c(\cdot)$ for each element are reported in parentheses; e.g., $\text{ISP}_4^{(2)}$ means that ISP_4 has been seen twice. Assuming one unseen IP per known ISP, unknown ISP and known country, and both unknown ISP and country, we have a total number $M = 5+3+1 = 9$ of unseen IPs (denoted with IP_u). According to Eq. (8), given $N = M = 9$, for an unseen IP, $p_0(x) = \frac{1}{18}$. For country-based estimates ($k = 1$), an unseen IP coming from C_1 has $p_1(x) = \frac{1}{8} \times \frac{5}{9}$ (the first term is the smoothed estimate $p(x|h_k)$ including unseen events), while if it comes from C_3, $p_1(x) = \frac{1}{3} \times \frac{1}{9}$. For ISP-based estimates ($k = 2$), if the unseen IP comes from ISP_1, then $p_2(x) = \frac{1}{4} \times \frac{3}{9}$. This value is higher than the probability of an unseen IP from ISP_5, i.e., $p_1(x) = \frac{1}{2} \times \frac{1}{9}$. For an unseen IP from ISP_1, we have: $p_2(x) = \frac{1}{4} \times \frac{3}{9} > p_1(x) = \frac{1}{8} \times \frac{5}{9} > p_0(x) = \frac{1}{18}$. However, for an unseen IP from both unseen ISP and country, we have that $p_0(x) = 1/18$, and $p_1 = p_2 = 0$.

which sample data exists. Here the coefficients $\alpha_0, \ldots, \alpha_\ell$ are normalization factors that can be computed offline to ensure that $\sum_x p_{\text{bo}}(x) = 1$ (see, e.g., [14] for details).

Recall from our discussion of smoothing above, we have $p_k(x) = p(x|h_k)p(h_k)$, with $p(x|h_k)$ the *smoothed* estimate of seeing the IP x in the entity h_k, and $p(h_k)$ the *unsmoothed* estimate of seeing the entity h_k. Our definition of $p_{\text{bo}}(x)$ in (10) implies that in the two extreme cases we use two different IP-level estimates: if IP x has been seen, then we use the unsmoothed estimate $p_\ell(x) = p(h_\ell) = c(x)/N$, while if x comes from an unseen country, then we use the smoothed estimate $p_0(x) = p(x|\text{world}) = c(x)/(N + M)$.

Linear Interpolation. Another method for aggregating the different estimates $p_k(x)$ is to linearly combine them as:

$$p_{\text{linear}}(x) = \sum_{k=0}^{\ell} \lambda_k p_k(x)\,, \tag{11}$$

where the coefficients $\{\lambda_k\}_{k=0}^{\ell}$ are learned using a held-out set to maximize the likelihood over unseen data, under the constraint that $\sum_{k=0}^{\ell} \lambda_k = 1$ (to ensure that the total probability mass sums up to one). The detailed procedure is given as Algorithm 1.

Similar procedures can be exploited for events conditioned to a given user and to the class of legitimate logins, i.e., to estimate $p(x|u, \text{L})$, restricting the available counts to the conditioning events.

Algorithm 1 Computing interpolation coefficients

Input: A training set T described as a set of vectors $\{\vec{t}_j\}$. Each $\vec{t}_j$ is an $\ell+1$-dimensional vector corresponding to a legitimate login event. The entries of $\vec{t}_j$ are the probabilities $p_k(x)$ for that event, as computed in Equation (9).

Output: Coefficients $\{\lambda_k\}_{k=0}^{\ell}$, s.t. $\sum_k \lambda_k = 1$.

1: Define a function σ that is a bijection between $\mathbb{R}^\ell$ and the open ℓ-dimensional simplex $\Sigma^\ell \subset \mathbb{R}^{\ell+1}$.

2: Define a function $L_T(\vec{v})\colon \mathbb{R}^\ell \to \mathbb{R}$ by

$$L_T(\vec{v}) = -\sum_j \log(\vec{t}_j \cdot \sigma(\vec{v})) \tag{12}$$

3: Use a numerical algorithm to compute the vector $\vec{v}_0 \in \mathbb{R}^\ell$ that maximizes $L_T(\vec{v})$.

4: Compute $\vec{\lambda} \leftarrow \sigma(\vec{v}_0) \in \mathbb{R}^{\ell+1}$.

5: Return $\vec{\lambda}$.

D. Feature Weighting

The independence assumptions used to derive our scoring function $g_u(\boldsymbol{x})$ (Eq. 7) lead to a formula in which all features have equal weight. However, given labeled account compromise data we may find that certain features are more important in detecting account compromises. To incorporate this property we can use a modified scoring function

$$\hat{g}_u(\boldsymbol{x}) = \left(\prod_{k=1}^{d} p(\text{A}|x^k)^{\alpha_k} \frac{p(x^k)^{\beta_k}}{p(x^k|u, \text{L})^{\gamma_k}} \right) \frac{p(u|\text{A})^\delta}{p(u|\text{L})^\epsilon}\,, \tag{13}$$

where the $\alpha_k, \beta_k, \gamma_k, \delta, \epsilon$ are real-valued weights. We can learn the values of these weights from the labeled training data by running a logistic regression classifier; specifically, we regress the sample labels against $\log(\hat{g}_u(\boldsymbol{x}))$.

III. Attack Model and Scenarios

We exploit an attack model defined in [6], [5], which builds on a popular taxonomy of potential attacks against machine learning proposed in [3], [2], [31]. This model helps identify potential attack scenarios that may be incurred by the learning algorithm during operation, and may suggest some simple countermeasures to mitigate their impact. The attack taxonomy categorizes attacks along three main axes: the *security violation*, the *attack specificity* and the *attack influence*. Based on these characteristics, the aforementioned attack model allows one to make explicit assumptions on the attacker's goal, knowledge of the attacked system, capability of manipulating the input data, and to devise a corresponding attack strategy.

Attacker's Goal. The goal is defined based on the following two characteristics:

- $(g.i)$ The desired *security violation*. The attacker can affect system **integrity** (if account takeovers are undetected), **availability** (if legitimate users can no longer access the system), or **privacy** (if confidential information about the system users is leaked) [3], [31], [6].
- $(g.ii)$ The *attack target*. The attack can be **targeted** (if the attack targets a specific user or set of users, i.e., if the attacker aims to have some specific samples misclassified), or **indiscriminate** (if any user account is potentially subject to the attack, i.e., any sample can be misclassified) [3], [31], [6].

Attacker's Knowledge. The attacker may have different levels of knowledge of the learning system [6], [5], [31] and, in particular, about:

- $(k.i)$ the training data;
- $(k.ii)$ the feature set, i.e., what features are used (e.g., IP and useragent);
- $(k.iii)$ the learning algorithm, i.e., the decision function $g_u(\boldsymbol{x})$ (Eq. 7);
- $(k.iv)$ its (trained) parameters, i.e., the probability estimates involved in the computation of $g_u(\boldsymbol{x})$;
- $(k.v)$ feedback on decisions (e.g., the attacker may observe whether a login attempt is classified as legitimate, suspicious, or malicious).

In authentication problems, it is also worth remarking that the attacker may know the user credentials, or exploit techniques to get them (e.g., information leakage from the targeted website or database).

Attacker's Capability. It consists of defining:

- $(c.i)$ the *attack influence*, i.e., whether the attacker can manipulate only testing data (**exploratory**), or also training data (**causative**) [3], [31], [6]; and
- $(c.ii)$ how samples (and the corresponding features) can be modified. This aspect should be defined based on application-specific constraints; e.g., in the authentication setting, an attacker may modify geolocation features if she can use a different IP address or gain access to a botnet.

Attack Strategy. This amounts to defining how the attacker implements the attack, based on the hypothesized goal, knowledge, and capabilities. In its most general sense, the attack strategy can be formulated as an optimization problem that in the end tells the attacker how to manipulate data to reach the given goal.

Attack Scenarios. Two main attack scenarios are often considered in the field of adversarial machine learning, i.e., **evasion** and **poisoning** [4], [7], [57], [3], [2], [31], [6], [5]. In an evasion attack, the attacker manipulates malicious samples at test time to have them misclassified as legitimate by a trained classifier, without having influence over the training data. This corresponds to an indiscriminate integrity violation. In a poisoning attack, the goal is to maximize classification error at test time by injecting *poisoning* samples into the training data. Influence in the poisoning setting is mainly on training data, and the goal is to cause an indiscriminate, availability violation.

While understanding the impact of poisoning on our system may be of interest, it may be very difficult for an attacker to get access to the training data (i.e., the login history) and actively manipulate it. On the contrary, evasion attacks are more likely to occur in practical settings, as the attacker can more easily manipulate data at test time (e.g., by changing the IP address or the browser's useragent) to increase chances of evading detection by our reinforced authentication module. For this reason, in this work we focus on the simulation of evasion attacks against our reinforced authentication system, as detailed in the next section, and leave the investigation of poisoning scenarios to future work.

A. Evasion Attacks

According to the framework discussed above, we define here several evasion settings considering attackers that have different knowledge of the attacked system and capabilities of modifying the way account takeovers are performed.

1) Attacker's Goal: In the context of user authentication, the goal of an evasion attack is to manipulate some features (e.g., the IP or browser's useragent) to have account takeover attempts misclassified as legitimate. Accordingly, the security violation amounts to an *integrity* violation (i.e., log in with the credentials of another user without being caught), while the attack specificity can be *targeted* (if the goal is to log in to a specific account, i.e., attack a specific user or set of users), or *indiscriminate* (if the goal is to log in to any account).

2) Attacker's Knowledge: We can define different kinds of attackers based on different assumptions about their level of knowledge of the attacked system. We consider here three distinct cases, corresponding to increasing levels of knowledge: the *no- knowledge attacker*, the *known-password attacker*, and the *phishing attacker*.

No-knowledge attacker. This is the least skilled attacker. She does not know any of the system details or implementation,

and does not even know any user credentials. An example is an attacker trying the password "password" against a list of possible usernames. Should one of the username/password pairs happen to be valid, this instance then falls into the known-credentials case.

Known-credentials attacker (KCA). In this case, it is assumed that the attacker has access to the full credentials for a user, and is a powerful threat to any password-based system. Without further security measures, there is no security left and the attacker can just access the account. This attacker is characterized by the following points:

- He knows full credentials for one or many users of a site. Usually, he knows username and password for a user on a site not protected by two-factor authentication.
- He may have a single (or a small number) of credentials (targeted KCA), or he may have access to a large number of credentials (such as a leaked list unknown to the site owner), and be interested in breaking into *any* account (indiscriminate KCA).

Phishing attackers. The most skilled attackers considered here are those we refer to as phishing attackers. They may have information beyond the credentials about the user in question; in particular, additional information could be obtained in one of the following ways:

- The attacker may personally know the victim and thus may know where the user is located and which device/OS/browser the user typically uses.
- In a sophisticated phishing attack, the attacker may also obtain more detailed information about the user, such as useragent string and IP address.

In terms of the points $(k.i)$-$(k.v)$ discussed in Sect. III, in these three cases scenarios the attacker has different levels of knowledge of the training data and the features used, potentially gathered by querying the targeted classifier and looking at the provided feedback on its decisions, while she does not exploit any knowledge on how the features are combined by the classifier.

3) Attacker's Capability: In the evasion setting, the attacker can only manipulate data during system operation, while she can not modify the training data. Each feature value of a login attempt can be manipulated depending on its nature, and on specific assumptions. In the following, we thus consider potential manipulation of the IP address and the browser's useragent. Similar reasoning should be extended to other features, if considered.

IP address. The attacker can change her IP address by using a remote proxy server or a botnet. If she attempts to log in as the targeted user and does not succeed, she may try to randomly use another IP, potentially not blacklisted. If knowledge about the user's ISP or country is available, the attacker may even try to retrieve an IP from the same source (i.e., ISP or country).

Browser's useragent. This feature can be also manipulated by the attacker, by using different browsers to login. Thus, if an attack is not successful, the attacker may attempt to login from another browser. Clearly, if it is known that the targeted user logs in usually using a specific browser, the same one can be adopted by the attacker.

4) Attack Strategy: Under the aforementioned assumptions, the attack strategy that achieves the attacker's goal of impersonating a targeted user (maximizing the probability of success of each attack) amounts to *mimicking* the behavior of the targeted user, under the constraints imposed by the attacker's knowledge and capability of manipulating the feature values. For instance, if the attacker comes to know that the targeted user logs in from a given country and using a given browser, she will do her best to mimic this behavior, i.e., log in using an IP from the same country and the same browser. In the adversarial evaluation of Sect. V we will consider two specific implementations of *KCA* and *phishing* attackers, each with a different level of knowledge.

It is worth remarking that if an attacker can almost exactly mimic the behavior of the targeted legitimate users in terms of their features (in our experiments, by correctly adjusting the IP address and useragent), then knowing how the classification function works in more detail will not increase the probability a successful attack. Exploiting knowledge of the classification function becomes useful when the classification system relies on a large number of features (with some potentially more difficult to mimic than others), so the attacker can understand which subset is worth attacking first to improve chances of misleading detection (see, e.g., [4], [36], [16]). Since in our evaluation we consider attackers that can successfully phish both the IP address and the useragent (i.e., can potentially mimic exactly the behavior of legitimate users), it is not worth considering attackers that also exploit knowledge of the classifier. We thus leave a more detailed investigation of such attacks to future work.

IV. System Implementation

We discuss here the prototype system implementation used in our experiments. Fig. 2 depicts the architecture of the proposed system in production. We maintain online a table of all successful user authentication attempts along with preprocessed feature attributes. The feature attributes store includes the precomputed "risk score" and global probabilities for the mentioned features. Both the user authentication history store and the feature attributes store feed into the scoring model, which computes $g_u(\boldsymbol{x})$ or $\hat{g}_u(\boldsymbol{x})$. Depending on the provided confidence score, the user is either granted or denied access. Further information may be requested using a challenge-response scheme if the scorer is not very confident. If the login attempt is successful, the online user authentication store is appropriately updated.

In terms of user experience, if the model score is in the "gray area" where the user cannot be determined with confidence to be either legitimate or malicious, we ask for further information as a proof of the user being legitimate. This could be a CAPTCHA challenge, a verification PIN sent to the user's registered phone via SMS, or a link sent to the user's email. Since the purpose of a CAPTCHA is to distinguish bots from humans, this defense can only be effective against a bot attack. For human-generated attacks the phone or email verification serves the same purpose as two-factor authentication: a legitimate user should easily be able

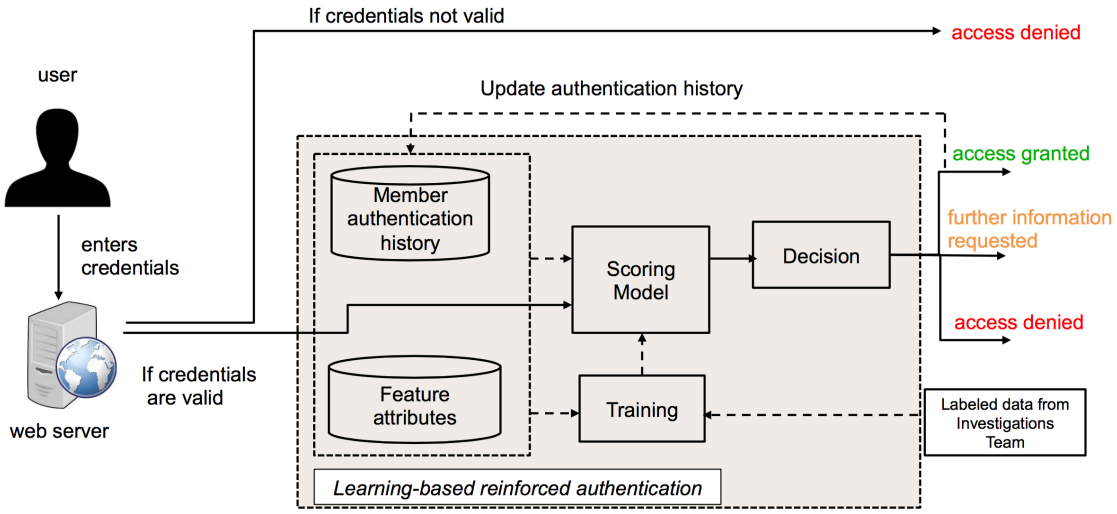


Fig. 2: Architecture of the proposed system with learning-based reinforced authentication. If the user credentials are wrong, the user is denied access. If the provided credentials are correct, additional information extracted from the login attempt is processed by the reinforced authentication system. The system outputs a low confidence score $g_u(\boldsymbol{x})$ if the retrieved information matches the normal behavior of the given user, as learned from the training data (e.g., logging in from a typical IP address and browser for that user). If the confidence score is lower than a predefined user-specific threshold, i.e., $g_u(\boldsymbol{x}) < \theta$, the user is correctly authenticated. Otherwise, access is denied even if the provided credentials are correct.

to pass the challenge, while the attacker must obtain access to the user's phone or email account in order to complete the login.

In our prototype implementation we choose to use the two features of IP address and useragent. We chose these features for several reasons: they are "sticky" for real users; they admit natural hierarchies, so we can apply the techniques of Sect. II-C to compute the desired probabilities in the presence of sparse data; they have different levels of cost to mimic (see Sect. VI); they are not strongly correlated with bot traffic (our primary goal is to improve detection of non-bot account compromise); and we could obtain them easily from the LinkedIn data set. It is true that in practice IP and useragent are correlated, and thus the independence assumption that allows us to derive Eq. (7) from Eq. (6) does not hold in practice; for example, mobile useragents are more likely to show up on a phone carrier's network. However, as we will see in Sect. V, this correlation is not strong enough to prevent the two features from providing complementary signals to the model.

In production the model will be periodically trained to incorporate changing trends in attack behavior. The training phase in our case amounts to estimating the parameters of the probability distributions involved in the computation of $\hat{g}_u(\boldsymbol{x})$ (Eq. 13) from the available data and computing the feature weights via logistic regression. This last phase uses labeled data from a team that manually investigates reports of account takeover. The output of the training phase is a set of parameters used to update the scoring model.

Our code for model evaluation and experiments was written in R [47] and executed on a single 2.8 GHz MacBook Pro. Once all the data was loaded into memory, the (amoritized) cost for scoring was less than 10 milliseconds per user. In practice the time taken to score will be dominated by the time required to fetch user history and feature attributes from online data stores.

V. Experimental Analysis

A. Experimental Setup

We built a prototype implementation of the model described in Sect. II using one year of login data from LinkedIn, the popular professional network. We acquired labeled data in the form of three classes: legitimate logins, compromised accounts, and login attempts from a single botnet attack. We distinguish the two types of compromise events because they have very different characteristics: compromised accounts by definition got past LinkedIn's existing login defenses, while the vast majority of the botnet attempts were blocked by said defenses.[3] Our success criterion in measuring our new model is to maintain a high level of protection against the botnet while improving protection against the known instances of account compromise.

For simplicity, our experiments focused on two features, IP address and useragent; see Sect. IV for a discussion of this choice. In practice LinkedIn's present and future login scoring models incorporate more than just these two features. Our experiments were conducted offline and at no point were live members scored by the particular models described below.

B. Dataset and Ground Truth

For our historical dataset we computed statistics on all successful logins to LinkedIn for the six months July–December

[3]We unfortunately have no labeled data on attempts other than the botnet attack that were blocked by the existing defenses.

2014. For every login attempt made on the site, we maintain data available in the HTTP headers like IP address and useragent along with password validation, timestamp, etc. We extracted global and per-member tables for IP address and useragent. For each IP address x^1 and member u, we use the login frequencies from the table to compute $p(x^1)$ and $p(x^1|u, L)$. Similarly, for each useragent x^2 and member u, we compute $p(x^2)$ and $p(x^2|u, L)$. We also used LinkedIn's internal reputation scoring system to compute "risk scores" for IP addresses and useragents seen over the same six-month period.

Our training and validation data came from logins in the six months January–June 2015. There were two types of positive cases: (1) a single botnet attack from January 2015 in which passwords were compromised but nearly all attempts were blocked by LinkedIn's existing defense, and (2) a set of accounts positively identified by LinkedIn's Security team as being compromised during the time frame. For negative cases, we sampled roughly 300,000 legitimate logins from this time period. We constructed our sample so that all accounts had at least one login during the six-month historical period, as our model is not designed to protect accounts that have no login history. (One could argue that all logins after a long dormant period should be treated as suspicious.)

Since LinkedIn was running some form of login-scoring system during the data collection period, our labeled data may not be representative of the real-life distribution of account-takeover attempts. In particular, beyond the single botnet incident (which LinkedIn identified from signals in internal data) we have no way of identifying with high confidence account-takeover attempts that were already blocked by this system. We could simply mark all blocked login attempts as account-takeover attempts, but since we have reason to believe that such attempts are a relatively small proportion of the total blocked login traffic, marking all blocks as positive samples would pollute our labeled dataset.

C. Baseline and Performance Metrics

Our baseline is one of the simplest rules used to protect login: flag the login as suspicious if it comes from a country not in the member's prior login history. We evaluated our data set against this criterion and found the following:

Class	Country History Match
Legitimate	96.3%
Compromise	93.3%
Botnet	1.0%

The disparity in history match between the compromised accounts and the botnet victims reflects the fact that country history match is a component of LinkedIn's existing login defenses.

Since the simple country-mismatch rule is already strong enough to stop 99% of (this particular) botnet attack, our goal in developing and tuning our model will be to maintain a high level of botnet protection while improving the coverage of known compromises, without significantly increasing false positives on legitimate accounts. To turn this goal into measurable statistics, we will assess the performance of our model against different types of attacks (either observed or simulated) by computing the True Positive Rate (TPR, i.e., the fraction of correctly classified account-takeover attempts) at 10% False Positive Rate (FPR, i.e., the fraction of legitimate attempts misclassified as attacks). Since false positives translate into extra steps required for good users to log in (see Sect. IV), the choice of an acceptable FPR is entirely a business decision to be made on a per-implementation basis; we choose 10% here as an arbitrary yet reasonable baseline.

To compare different choices of model (e.g., different smoothing techniques) we plot the Receiver Operating Characteristic (ROC) curve, which shows how the TPR varies as a function of the FPR for different decision thresholds, and compute the Area Under the ROC Curve (AUC). Notice that using AUC as a performance metric allows us to avoid choosing a specific decision threshold for classification. We also remark that such metrics are insensitive to class imbalance in the training data, which is necessary as we significantly downsampled login data to create our training set.

D. System Performance and Model Variations

Using the LinkedIn data made available to us, we computed confidence scores $g_u(\boldsymbol{x})$ using Eqs. (7) and (13). We tried several different combinations of parameters and used area under the ROC curve as our metric for comparison. The dimensions along which we varied parameters were as follows:

Smoothing. We evaluated both backoff and linear interpolation smoothing, as described in Sect. II-C. We also evaluated two different choices for the parameter μ_{h_k} representing the number of unseen IP addresses for each entity h_k. Our first choice was $\mu_{h_k} = 1$ for all h_k, as represented in Fig. 1. However, we encountered two problems with this choice:

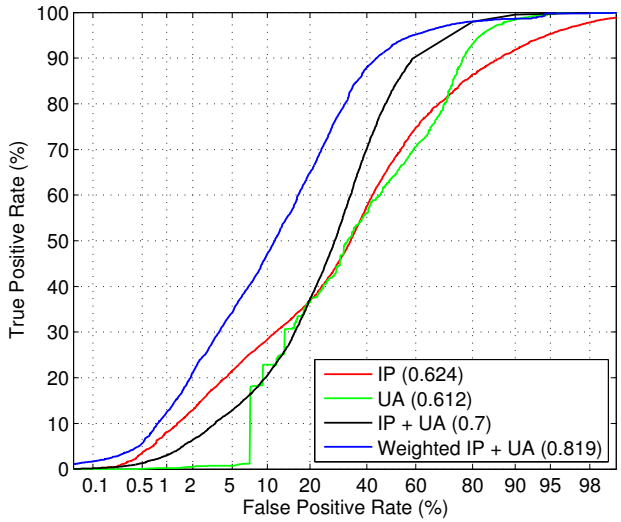
- First, $\mu_{h_k} = 1$ does not properly penalize a country mismatch. For example, in the scenario developed in Fig. 1, a login from an IP in the most common ISP in the most common country is assigned a probability $1/9$, while a login from a completely new country is assigned a probability $1/18$. From our experience we feel that the relative risk ratio of these two events should be much greater than 2, so we want to assign a much lower probability to the new country.
- Second, we found that Algorithm 1 did not converge when run on our data set with smoothing $\mu_{h_k} = 1$, while it did converge when we set μ_{h_k} to be much larger.

We therefore recomputed the features with $\mu_{h_k} = |h_k|$, the total number of IP addresses seen in entity h_k, to obtain three different choices for smoothing: backoff with $\mu_{h_k} = 1$, backoff with $\mu_{h_k} = |h_k|$, and interpolation with $\mu_{h_k} = |h_k|$.

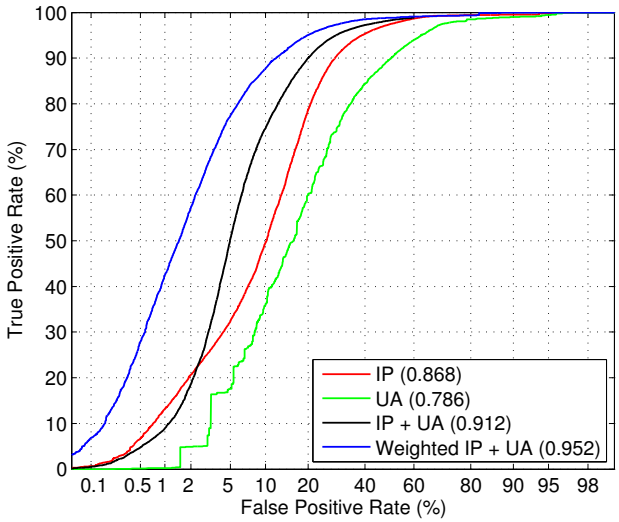
We reserved 40% of our training set to train the interpolation coefficients, and evaluated all of our models on the remaining 60% of the data.

Features and feature weights. For comparison purposes we considered four different ways to combine features:

- IP address only, using Eq. (7),
- Useragent (UA) only, using Eq. (7),
- IP address and useragent, using Eq. (7),

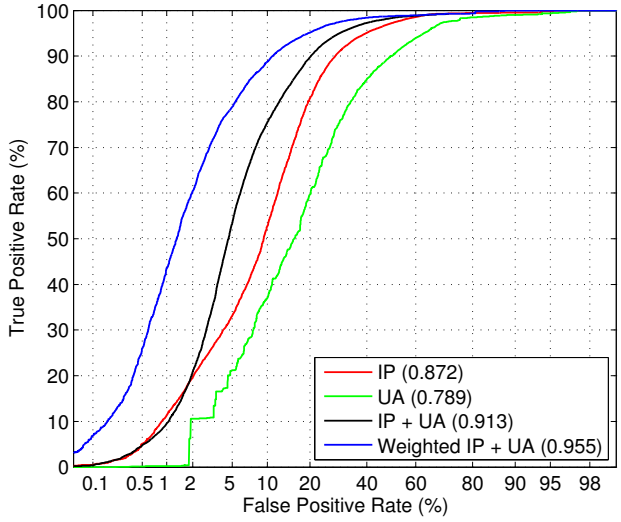


(a) Backoff smoothing, $\mu_{h_k} = 1$.

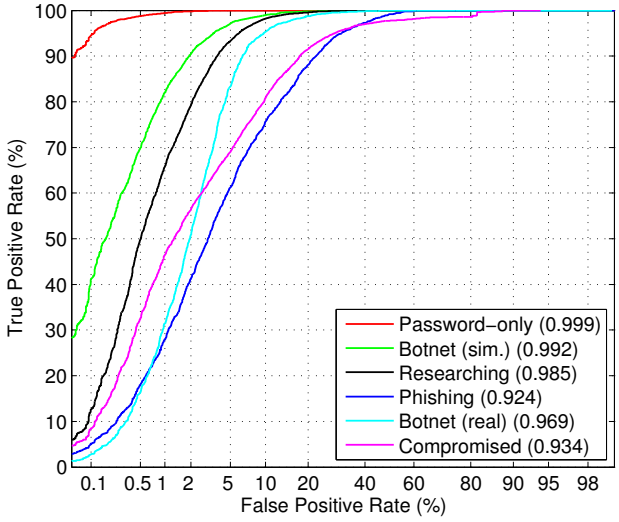


(b) Backoff Smoothing, $\mu_{h_k} = |h_k|$.

Fig. 3: ROC curves for the two backoff smoothing techniques with $\ell = 4$. AUC values are reported in parentheses.



(a) Interpolation smoothing.



(b) Evasion attacks.

Fig. 4: ROC curves for interpolation smoothing with $\ell = 4$ and evasion attacks. AUC values are reported in parentheses.

- IP and useragent features weighted by a logistic regression model, using Eq. (13) as described in Sect. II-D.

For the logistic regression model, we trained the classifier on 60% of the data and evaluated it on the remaining 40%.[4]

Levels of granularity. Both IP addresses and useragents can be aggregated into groupings of increasing size. We wanted to determine the effect of changing the number of levels ℓ of the hierarchy, so we considered two such hierarchies for each feature:

- For IP addresses, we considered hierarchies of (IP address, organization, autonomous system (AS), country, world) ($\ell = 4$); and (IP, AS, world) ($\ell = 2$).
- For useragents, we considered nested hierarchies of (useragent, browser family, operating system, app, world) ($\ell = 4$); and (useragent, OS, world) ($\ell = 2$). "App" refers to the user experience: desktop, mobile web, native mobile app, or unknown. By a "nested hierarchy" we mean that the level above useragent is app + OS + browser, then app + OS, etc.

Results. We computed 24 different sets of scores (3 smoothing methods $\times$ 4 feature combinations $\times$ 2 sets of feature hier-

[4]Since we already reserved 40% of the data to train interpolation coefficients, as a proportion of all entire dataset we trained on 36% and evaluated on 24%.

Features	Backoff, $\mu_{h_k} = 1$		Backoff, $\mu_{h_k} = \lvert h_k \rvert$		Interpolation, $\mu_{h_k} = \lvert h_k \rvert$	
	$\ell = 2$	$\ell = 4$	$\ell = 2$	$\ell = 4$	$\ell = 2$	$\ell = 4$
IP	0.56	0.624	0.82	0.868	0.82	0.872
UA	0.67	0.612	0.76	0.786	0.75	0.789
IP + UA	0.68	0.7	0.89	0.912	0.88	0.913
Weighted IP + UA	0.819	0.82	0.94	0.952	0.94	0.955

TABLE I: AUC values for various feature combinations, smoothing models, and number of levels ℓ in feature hierarchies.

Features	Backoff, $\mu_{h_k} = 1$		Backoff, $\mu_{h_k} = \lvert h_k \rvert$		Interpolation, $\mu_{h_k} = \lvert h_k \rvert$	
	$\ell = 2$	$\ell = 4$	$\ell = 2$	$\ell = 4$	$\ell = 2$	$\ell = 4$
IP	0.19	0.19	0.28	0.39	0.28	0.44
UA	0.29	0.32	0.41	0.46	0.40	0.47
IP + UA	0.14	0.15	0.58	0.74	0.56	0.75
Weighted IP + UA	0.43	0.43	0.83	0.88	0.83	0.89

TABLE II: TPR at 10% FPR for various feature combinations, smoothing models, and number of levels ℓ in feature hierarchies.

archies) for each login in our test set. Table I gives AUC for each combination. Table II shows the TPR at 10% FPR for each set of scores. ROC curves with ($\ell = 4$) and 4 feature combinations for the 3 smoothing techniques are plotted in Fig. 3 and Fig. 4.

The data clearly show that IP address and useragent are not very powerful on their own; they begin to detect a reasonable number of account takeovers when combined, and when the features are weighted via logistic regression they offer good detection capability. The data also show that adding levels to the hierarchy gives a small but noticeable performance boost. As for the optimal smoothing method, it is clear that setting $\mu_{h_k} = 1$ is the poorest choice. However, when $\mu_{h_k} = |h_k|$ the backoff and interpolation methods are essentially equivalent, with interpolation showing a very small edge.

Given these results, we choose as our best model a logistic regression model over IP and useragent features using the $\ell = 4$ hierarchy, smoothed via interpolation with $\mu_{h_k} = |h_k|$. Using this model we evaluated performance on the botnet attack and the compromised accounts separately. We found that at 10% FPR we could detect 95% of the botnet attack and 77% of the compromised accounts. Thus by relaxing our allowed FPR to 10% we can improve the detection rate of compromised accounts by a factor of 12, while only letting an additional 4% of botnet accounts through.

E. Experiments: Evasion Attacks

To test our model against attackers with various levels of sophistication, we simulated attack datasets for four types of attacks. We assume that at a minimum all of the victims' passwords are known to the attacker; in some cases the attacker has more information. In each case we simulated the attack by randomly sampling members from the historical period and assigning each attacked member an IP address and useragent as discussed in the following.

1) Password-only attacker: This is a known-credentials attack in which the attacker does not possess any information beyond the target members' credentials. The attacker writes a simple script using an off-the-shelf package and uses a hosting provider to launch the attack. To simulate this attack, we sampled IP addresses from a known hosting provider weighted by their "risk score" as calculated by LinkedIn, and we give all attempts the useragent "Python-httplib2/0.7.2 (gzip)". Our best model easily identified this attack, with AUC 0.999 and 99% TPR at just a 1% FPR.

2) Botnet attacker: This is also a known-credentials attack similar to the previous scenario except that the attacker here has more resources at his disposal. The attacker in this case knows that he should vary IP addresses and useragents to get around simple rate-limiting and bot detection. He thus employs a botnet to launch the attack and randomizes his useragent string. This would imply that both IPs and useragent of login attempts would appear to be distributed. To simulate this attack, we sampled IPs weighted by their risk score and sampled useragents uniformly at random from the historical data set. Our best model did well at detecting this attack, with AUC 0.992 and 99% TPR at 10% FPR. As expected, using useragent features only was useless in detecting this attack: using useragent features alone we could only detect less than 0.2% of true positives at 10% FPR.

3) Researching attacker: Moving on to a more sophisticated attack, we assume that the attacker in this case scrapes information about the target's country. The attacker then uses proxies only from the target's country to make the logins look less suspicious. To simulate this attack, we sample members with at least one successful login from a target country (United States in our experiments). We then sample IPs from this country weighted by their risk score and use one common useragent across all simulated login attempts. Our best model did well at detecting this attack, with AUC 0.985 and 99% TPR at 10% FPR.

4) Phishing attacker: Most motivated of all attacker types, we assume that this category of attacker has been successfully able to phish its targets thereby obtaining useragent of their browser and IP address. With this information, the attacker uses the exact same useragent and an IP from the same country as the victim to launch the attack. To simulate this attack type, we sample member sessions from training set. For these members we use the same useragent and sample an IP from the same country, again weighted by the IP risk score. As

Attacker	AUC	TPR at 10% FPR
Password-only	0.999	1.00
Botnet (simulated)	0.992	0.99
Researching	0.985	0.99
Phishing	0.924	0.74
Botnet (real)	0.969	0.95
Compromised accounts	0.934	0.77

TABLE III: Performance of our best model under attack.

expected, this was the hardest attack to detect. Our best model gave AUC 0.924 and 74% TPR at 10% FPR.

A summary of our experiment results is given in Table III, while the corresponding ROC curves are shown in Fig 4. We also include results on the two labeled sets of account takeovers from the LinkedIn data, i.e., the (real) botnet attack and the compromised accounts. Notably, the performance of our system improves against the less knowledgeable attackers simulated in this experiment, since the real attacks present in the LinkedIn data are actually more sophisticated.

F. Experiment: Feature Effects

Reputation systems. To test the effect of the reputation systems on our model, we evaluated our best model on the same training and validation set as above, but set the reputation scores $p(A|x)$ to be identical for all feature values x. We found that our ability to detect the botnet attack actually improved, from 95% at 10% FPR to 99%. On the other hand, our ability to detect compromised accounts degraded, going from 77% to 60% at 10% FPR. These results demonstrate that LinkedIn's reputation systems do reflect the probability of attack to some extent; they also show that the particular botnet incident in question came from IP addresses and/or useragents that were not scored as particularly abusive by LinkedIn.

As for the attack data, performance on Attacks 1 and 2 remained strong 99% and 97% TP at 10% FPR, respectively, while performance on Attack 3 decreased to 80% and detection of Attack 4 dropped dramatically, to 19%. These results show that for a motivated attacker having a robust and accurate set of reputation scores is essential to repel the attack.

Member history data. We wanted to test the hypothesis that members with more logins in their history are better protected by our model. To do this we split the member set into two parts: those with 8 or more logins in their history (27% of our data set) and those with fewer than 8 logins (73% of our data set) and evaluated our best model on each. We found that for members with less history, compromised accounts were slightly less likely to be detected (75% vs. 77% at 10% FPR) and botnet victims were slightly more likely to be detected (97% vs. 94%). We believe that this difference is mostly due to the fact that botnet victims were proportionally overrepresented amongst the group with few logins.

For Attacks 1–3 the model performance was essentially the same on both segments. For Attack 4 we found that members with more history were better protected; this is due to the fact that the attacker picked a session randomly from the member's history to emulate, so if the member has little history then the estimates for $p(\boldsymbol{x}|u, L)$ will be higher for the emulated session than if the member has many events in her history.

VI. Classifying features

Individual features chosen for our model have various properties that are relevant for reinforced authentication. We now propose a classification along three axes that allows us to study and compare their properties and their usefulness.

Phishability. This characteristic concerns the resistance of a feature to phishing attacks. In a traditional phishing attack, the attacker records the username/password pair, which allows him access to the service A. If the service A uses a scheme such as the one studied in this work, simply recording the username and password will not give the attacker access to A. However, an attacker can adapt to the new situation and, in addition to username and password, can record all the features that normally would be recorded by the service A, and can utilize the gained information when accessing A.

We say a feature is *phishable* or can be phished if observing a single instance of an accepted login allows an attacker to fake that specific feature with high probability. We say a feature is *non-phishable* if observing a single login allows the attacker to fake that feature with negligible probability only. Some features may be learnable not from a single observed login but from a few. Luring the same user to a phishing site more than once is substantially more complicated, in particular if there are constraints such as that the logins must be consecutive. Such features pose much less of a problem, but may need to be considered in some circumstances.

Cost to mimic. Even phishable features can be useful in identifying a phishing attack, as an attacker needs to mimic the feature when logging in, which comes at a cost.

We say a feature is *cheap to mimic* (resp., *expensive to mimic*) if mimicking the feature incurs minimal (resp., high) cost to the attacker. Here "cost" can refer to different quantities: money, time required for coding, time required for execution, and others. Also, when a cost is deemed "high" or "low" is most likely application-specific and outside the scope of this discussion.

Accuracy. Discriminating properties of the different features used in the model vary. Features with *high accuracy* (respectively, *low accuracy*) are, if matching, a strong (respectively, weak) indication that the login attempt is legitimate. The relative accuracy of different features can be assessed by evaluating the performance of a number of one-feature classifiers in a given model framework.

A. Reinforced Authentication Features

We now apply this classification to some features that are used for reinforced authentication and discuss their properties.

The *source IP*, as well as the directly related *source ASN* and *source country*, are *phishable* features; i.e., a phishing attack immediately gives away a valid source IP. However, they still constitute reasonable features since they are moderately expensive to mimic. Mimicking the exact source IP is usually quite hard; however, having a source IP from the same ASN

will usually be sufficient. Mimicking still requires effort, e.g., by having access to a botnet that performs the login attempts from a bot in the required ASN or country. Another quality is the *high accuracy* of the source IP, as established in Sect. V.

We discuss here two different variants of the useragent feature. The *useragent string*, i.e., the identifier transmitted as part of the HTTP header, is easily *phishable*. Unlike the IP address, the useragent string is very *cheap to mimic* by simply adapting the HTTP header field. This feature is less accurate than IP address, as demonstrated in Sect. V. However, one does not need to rely on the useragent string to determine the software running on the user's machine. *Browser fingerprinting techniques* can be used to gather precise information on the software running, including the browser, browser extensions, the OS, and if it is running inside a virtual machine. While the resulting feature is still *phishable*, precisely simulating all the gathered information can become costly, which makes the feature *moderately expensive to mimic*, in particular when targeting more than a very limited number of accounts.

Time between login events is an example of a feature that is *non-phishable*, and thus provides a different quality of security. Cost to mimic can be substantial, as the attacker has to learn from multiple (consecutive) successful login attempts by the victim and wait a specific time between login attempts. While we did not incorporate this feature into our models, a rough analysis of the LinkedIn data suggests that the feature is of low accuracy since the distributions of the feature for legitimate and attack events are very similar.

VII. Related work

The weakness of passwords has been understood for a long time. Password re-use is problematic as leakage from a single password puts multiple accounts of a single user at risk. Studies have consistently shown that users re-use passwords, and even re-use passwords from high-value accounts on low-value accounts [25], [1], [17]. Weak passwords is also a widely known problem. The strength of user-chosen passwords against password guessing attacks has been studied since the early times of password-based authentication [8], [56], [40] Current techniques for password guessing are Markov models [44], [21], [37] and probabilistic context-free grammars [55]; state-of-the-art tools include John the Ripper [51] and HashCat [52]. Historically, the strength of passwords against guessing attacks has been assessed by using password crackers to find weak passwords [42]. Recently much more precise techniques have been developed [8], [50], [13], [18], [22].

One common alternative to password-based authentication is using *authentication tokens*, usually in *two-factor authentication*. The authentication token can either be a hardware token, an app running on a smart-phone, or a second communication channel such as a mobile phone number. Security tokens offer a high security level, if implemented and used correctly, and are implemented at most major websites as an optional feature. However, adoption rates are low, as users are often unwilling to carry around the security token, which can be lost and stolen, and needs to be connected to the device for each authentication request. Furthermore, managing tokens for Internet-wide service with a diverse user-base is a challenge. Another common alternative is *biometric authentication*, based on fingerprints [39], face recognition [34], typing dynamics [28], [41], [32], [19], or many other factors. Biometric schemes are rarely implemented for large online services, as they often require special hardware on the client side, are difficult to implement securely for remote logins, and raise privacy concerns.

There is very little work available that considers classifying suspicious behavior at login time. While it is obvious from personal experience that a number of websites use some form of reinforced authentication, very few details about their systems are known and the effectiveness has never been publicly discussed. In a presentation [46], a Facebook engineer gives some idea about their system at the time, which uses a whitelist based on browser cookies, IP ranges, and geolocation, a blacklist based on several known attack patterns (such as many failed login attempts from an IP), and some undisclosed additional mechanisms. A high-level comparison of commercially available tools for protecting online accounts (named risk-based authentication) is available from Forrester [38], but no technical details are provided. RSA provides some information about it's risk-based authentication solution [24], but again the disclosed information is minimal. The lack of public evaluation is against the standards typically applied to cryptography and security, thus risking that weaknesses found by adversaries may get unnoticed.

Related systems are in place monitoring transactions in financial networks. For credit card transactions in particular, automatic classifiers are used to select possibly fraudulent transactions for manual review. An RSA white paper gives some insight into RSA's solution for credit card protection [23]. Florencio et al. [26] mention that, for financial online accounts, the security is indirectly given by the fraud-detection back-end, which uses machine learning as well.

In a recent survey paper, Bonneau et al. [10] give an overview over current challenges for password-based authentication, and mention a machine-learning approach as the most likely solution to the problems that user authentication faces today. Potential privacy implications of reinforced authentication have been also recently discussed in a position statement [9].

It is worth mentioning that, as advocated by the adversarial machine learning field, protection systems based on statistical knowledge extracted from data are prone to be attacked by skilled adversaries who can purposely modify data to mislead the outcome of the automatic analysis [6], [4], [31], [3], [12]. We have performed a preliminary investigation of the vulnerabilities of our system against *evasion* attacks, in which attackers aim to impersonate legitimate users during operation. Another interesting scenario may be that of *poisoning* attacks [3], [48], [31], [7], [57], in which the attacker may tamper with the training data to mislead learning; e.g., she may try to increase the reputation of IPs from which she is going to launch a future attack, by legitimately logging in several times from them, to accounts that are not necessarily associated to real users, but created on purpose. A crowdturfing campaign may be also staged to this end [53].

Besides considering more challenging attack settings, we may exploit some countermeasures proposed in the area of adversarial machine learning to improve system security in

more practical cases. Most of the work in that area has been devoted to countering attacks in a *proactive* manner, by explicitly modeling the interactions between classification algorithms and attackers, or considering attacks as outlying samples with respect to the expected, normal behavior [16], [12], [29], [45], [48], [15], [31], [3]. While investigating these countermeasures may be of interest for future work, it is well known that, in practice, system security can also be significantly improved in a *reactive* manner, by timely detecting novel attacks and retraining the system, and verifying the consistency of classifier decisions with the (labeled) training data [5], [31].

VIII. Conclusions and Future Work

In this work we have evaluated an approach to strengthening password-based authentication by classifying login attempts into normal and suspicious activity based on parameters available during login. This approach is particularly useful for Internet-wide services with a large and diverse user base, as it can be deployed without changing the user experience. Similar schemes are in use by large websites, and our work is the first public analysis and benchmark of such approaches. We make no claim that our system is more sophisticated or more accurate than any given non-public scheme, and we welcome contributions to the literature that offer contrasting approaches.

In Sects. II–VI we have described a statistical framework, provided a systematic study of possible attackers, developed a fully functional prototype implementation, and validated the system on a sample of real-life login data from LinkedIn, showing a recall of up to 89% for a false-positive rate of 10%, using the user's IP history as well as the useragent-string history.

Several directions seem promising for future work. The classifiers we considere do not take into account *temporal correlation* among login attempts. This may carry useful information, as attacks are often launched in campaigns and in a short time span. Hence, if a login attempt is marked as an attack, then other login attempts that are close in time to this particular one and have a substantial overlap of feature values are likely to be account- takeover attempts as well.

Manually labeling account-takeover events is a time-consuming task. Techniques like *active learning* can be used to reduce this effort, by smartly choosing login events that require manual labeling to maximally improve the classifier.

For our prototype evaluation of the model, the feature set computation is performed on historical data instead of in real-time. By updating *frequency features* in real time, we can potentially avoid cases which were wrongly marked as account-takeover attempts due to lack of updated user login history in our dataset.

Acknowledgments

This work has been partly supported by the project "Advanced and secure sharing of multimedia data over social networks in the future Internet" funded by the Regional Administration of Sardinia, Italy (CUP F71J11000690002).

References

[1] D. V. Bailey, M. Dürmuth, and C. Paar, "Statistics on password reuse and adaptive strength for financial accounts," in *Security and Cryptography for Networks*, ser. Lecture Notes in Computer Science, vol. 8642. Springer, 2014, pp. 218–235.

[2] M. Barreno, B. Nelson, A. Joseph, and J. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, pp. 121–148, 2010.

[3] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. ACM Symp. Information, Computer and Comm. Sec.*, ser. ASIACCS '06. New York, NY, USA: ACM, 2006, pp. 16–25.

[4] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Part III*, ser. Lecture Notes in Computer Science, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds., vol. 8190. Springer Berlin Heidelberg, 2013, pp. 387–402.

[5] B. Biggio, G. Fumera, and F. Roli, "Pattern recognition systems under attack: Design issues and research challenges," *Int'l J. Patt. Recogn. Artif. Intell.*, vol. 28, no. 7, p. 1460002, 2014.

[6] ——, "Security evaluation of pattern classifiers under attack," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 4, pp. 984–996, April 2014.

[7] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *29th Int'l Conf. on Machine Learning*, J. Langford and J. Pineau, Eds. Omnipress, 2012, pp. 1807–1814.

[8] M. Bishop and D. V. Klein, "Improving system security via proactive password checking," *Computers & Security*, vol. 14, no. 3, pp. 233–249, 1995.

[9] J. Bonneau, E. Felten, P. Mittal, and A. Narayanan, "Privacy concerns of implicit secondary factors for web authentication," in *WAY 2014: Who are you?! Adventures in Authentication Workshop*, July 2014.

[10] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The past, present, and future of password-based authentication on the Web," *Communications of the ACM*, March 2015.

[11] J. Bonneau and S. E. Schechter, "Towards reliable storage of 56-bit secrets in human memory," in *Proc. 23rd USENIX Security Symposium*, 2014, pp. 607–623.

[12] M. Brückner, C. Kanzow, and T. Scheffer, "Static prediction games for adversarial learning problems," *J. Mach. Learn. Res.*, vol. 13, pp. 2617–2654, September 2012.

[13] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from Markov models," in *Proc. Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2012.

[14] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech & Language*, vol. 13, no. 4, pp. 359–393, 1999.

[15] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *IEEE Symposium on Security and Privacy*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 81–95.

[16] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Seattle, 2004, pp. 99–108.

[17] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2014.

[18] X. de Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2014.

[19] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and I know it's you!: Implicit authentication based on touch screen patterns," in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. ACM, 2012, pp. 987–996.

[20] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[21] M. Dürmuth, F. Angelstorf, C. Castelluccia, D. Perito, and A. Chaabane, "Omen: Faster password guessing using an ordered markov enumerator," in *Proc. International Symposium on Engineering Secure Software and Systems (ESSoS)*, 2015, to appear.

[22] S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov, and C. Herley, "Does my password go up to eleven?: The impact of password meters on password selection," in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13. ACM, 2013, pp. 2379–2388.

[23] EMC Corp., "RSA adaptive authentication," http://www.emc.com/collateral/data-sheet/11637-h9077-aaecom-ds.pdf.

[24] ——, "RSA risk-based authentication," http://www.emc.com/collateral/data-sheet/h11506-rsa-rba-ds.pdf.

[25] D. Florencio and C. Herley, "A large-scale study of web password habits," in *WWW '07: Proc. 16th International Conference on the World Wide Web*. ACM, 2007, pp. 657–666.

[26] ——, "Is everything we know about password stealing wrong?" *Security Privacy, IEEE*, vol. 10, no. 6, pp. 63–69, Nov 2012.

[27] D. Florencio, C. Herley, and P. C. van Oorschot, "Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts," in *Proc. 23rd USENIX Security Symposium*, 2014, pp. 575–590.

[28] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro, "Authentication by keystroke timing: Some preliminary results," DTIC Document, Tech. Rep., 1980.

[29] A. Globerson and S. T. Roweis, "Nightmare at test time: robust learning by feature deletion," in *Proc. 23rd International Conference on Machine Learning*, W. W. Cohen and A. Moore, Eds., vol. 148. ACM, 2006, pp. 353–360.

[30] "Google authenticator," Online at https://code.google.com/p/google-authenticator/.

[31] L. Huang, A. D. Joseph, B. Nelson, B. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *4th ACM Workshop on Artificial Intelligence and Security (AISec 2011)*, Chicago, IL, USA, 2011, pp. 43–57.

[32] M. Karnan, M. Akila, and N. Krishnaraj, "Biometric personal authentication using keystroke dynamics: A review," *Applied Soft Computing*, vol. 11, no. 2, pp. 1565–1573, 2011.

[33] S. Komanduri, R. Shay, L. F. Cranor, C. Herley, and S. E. Schechter, "Telepathwords: Preventing weak passwords by reading users' minds," in *Proc. 23rd USENIX Security Symposium*, 2014, pp. 591–606.

[34] S. Z. Li and A. Jain, Eds., *Handbook of Face Recognition*, 2nd ed. Springer, 2011.

[35] Z. Li, W. He, D. Akhawe, and D. Song, "The emperor's new password manager: Security analysis of web-based password managers," in *Proc. 23rd USENIX Security Symposium*, 2014, pp. 465–479.

[36] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. Chicago, IL, USA: ACM Press, 2005, pp. 641–647.

[37] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 689–704.

[38] E. Maler, A. Cser, S. Balaouras, and J. McKee", "The Forrester wave: Risk-based authentication," Online at http://www.arrowecs.be/?event=tools.ehgetfile.FileHandler&ting&f_string=/FMS/19644.the_forrester_wave_risk_based_authentication.pdf, 2012.

[39] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, 2nd ed. Springer, 2009.

[40] S. Marechal, "Advances in password cracking," *Journal in Computer Virology*, vol. 4, no. 1, pp. 73–81, 2008.

[41] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation computer systems*, vol. 16, no. 4, pp. 351–359, 2000.

[42] R. Morris and K. Thompson, "Password Security: A Case History," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, Nov. 1979.

[43] K. Nandakumar, Y. Chen, S. C. Dass, and A. Jain, "Likelihood ratio-based biometric score fusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 342–347, February 2008.

[44] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. 12th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2005, pp. 364–372.

[45] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *LEET'08: Proc. 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. Berkeley, CA, USA: USENIX Association, 2008, pp. 1–9.

[46] C. Palow, "After watching this talk, you'll never look at passwords the same again," Presentation at the Hacker News Meetup, London. Recording available online http://vimeo.com/80460475, November 2013.

[47] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014. [Online]. Available: http://www.R-project.org/

[48] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proc. 9th ACM SIGCOMM Internet Measurement Conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 1–14.

[49] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," *AAAI Technical Report WS-98-05, Madison, Wisconsin*, 1998.

[50] S. Schechter, C. Herley, and M. Mitzenmacher, "Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks," in *Proc. 5th USENIX Conference on Hot Topics in Security*. USENIX Association, 2010, pp. 1–8.

[51] Solar Designer, "John the Ripper," Online at www.openwall.com/john.

[52] J. Steube, "OclHashcat performance comparison," Online at http://hashcat.net/oclhashcat/.

[53] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao, "Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, 2014.

[54] WatchGuard Technologies, Inc., "Watchguard reputation authority," Available at http://www.borderware.com, February 2015.

[55] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2009, pp. 391–405.

[56] T. Wu, "A real-world analysis of kerberos password security," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 1999.

[57] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *JMLR W&CP - Proc. 32nd Int'l Conf. Mach. Learning (ICML)*, F. Bach and D. Blei, Eds., vol. 37, 2015, pp. 1689–1698.