Distinguishing Attacks from Legitimate Authentication Traffic at Scale

Cormac Herley
Microsoft Research
cormac@microsoft.com

Stuart Schechter[†]
stuart@post.harvard.edu

Abstract—**Online guessing attacks against password servers can be hard to address. Approaches that throttle or block repeated guesses on an account (e.g., three strikes type lockout rules) can be effective against depth-first attacks, but are of little help against breadth-first attacks that spread guesses very widely. At large providers with tens, or hundreds, of millions of accounts breadth-first attacks offer a way to send millions or even billions of guesses without ever triggering the depth-first defenses. The absence of labels and non-stationarity of attack traffic make it challenging to apply machine learning techniques.**

We show how to accurately estimate the odds that an observation x indicates that a request is malicious. Our main assumptions are that successful malicious logins are a small fraction of the total, and that the distribution of x in the legitimate traffic is stationary, or very-slowly varying. From these we show how we can estimate the ratio of bad-to-good traffic among any set of requests; how we can then identify subsets of the request data that contain least (or even no) attack traffic; how these least-attacked subsets allow us to estimate the distribution of values of x over the legitimate data, and hence calculate the odds ratio. A sensitivity analysis shows that even when we fail to identify a subset with little attack traffic our odds ratio estimates are very robust.

I. Introduction

Web service providers who offer password authentication to their users must protect against both online and offline guessing attacks. If an attacker gains access to the password file she can mount an offline attack by repeatedly guessing at a rate limited only by her hardware. The power of modern password cracking tools make this a severe threat. To protect against offline attacks it is generally recommended that passwords be salted and hashed [1], that an iterated [2] or memory-hard [3] [4] hash be used, that honey tokens be employed to trigger an alert in the event that the file leaks [5], and/or that stealing the hashed file be made infeasible [6]. Efforts to get user to choose passwords that will resist offline guessing have largely failed [7]: studies of actual passwords show persistent weakness [8] [9] [10] [11]. Possible explanations are that users' perception of their vulnerability is unrealistic [12] or that they perceive too great a burden for too little return [13].

Online guessing attacks might seem less serious in the sense that the attacker cannot guess at the same rate as offline. Even so they represent a widespread threat. Since the basic purpose of an authentication server is to admit those who present a valid username/password pair, every server must anticipate online guessing traffic.

Online guessing attacks are generally divided into targeted (or depth-first) and untargeted (or breadth -first) approaches. In a targeted attack a large number of guesses are directed against a small number of accounts that presumably are of particular interest to an attacker. The email or social networking accounts of celebrities, for example, are obvious targets. Wang et al. suggest that targeted online guessing is underestimated as a threat [14]. In an untargeted attack guesses are distributed among a large number of accounts. Here the attacker is not interested in particular individuals so much as access to the resource that an account represents. For example, email accounts of users with good reputation can be used to send spam with much higher success rates than recently-opened accounts with no history.

Not only has it proved difficult to get users to choose passwords that will withstand offline guessing, the much lower threshold of resisting online guessing has also proved hard [10]. Users show considerable preference for easy-to-remember passwords, and great ingenuity in circumventing rules and policies put in place to forbid common choices [15] [16]. Studies of passwords from breached data sets consistently show that a dictionary of 100 or so passwords captures on the order of 5% of accounts [10] [11]. This means that attackers can reliably compromise significant numbers of accounts by spreading guesses (most of which will fail) among many accounts.

Untargeted attacks can be harder to detect since the guessing is distributed across many accounts. In spite of how common the problem is many of the recommended defenses appear vague or ambiguous. The National Institute of Standards and Technology (NIST) recommends that [17] "login traffic be monitored for suspicious activity." The UK's CESG recommends [18] "protective monitoring to detect and alert to malicious or abnormal behaviour, such as automated attempts to guess or brute-force account passwords." The Open Web Application Security Project (OWASP) recommends that [19]

[†] Work performed while author was at MSR.

Network and Distributed Systems Security (NDSS) Symposium 2019
24-27 February 2019, San Diego, CA, USA
ISBN 1-891562-55-X
https://dx.doi.org/10.14722/ndss.2019.23124
www.ndss-symposium.org

"all failures are logged and reviewed." It is not entirely clear how these recommendations might be turned to action. It is also not feasible to review the potentially millions of daily failures that a large site might receive. Large web service providers presumably have some mechanisms for dealing with online guessing, but none appears to publicly divulge their approach.

The problem of online guessing appears significant. In 2009 Twitter was the subject of a successful large-scale online dictionary guessing attack which (among others) compromised the accounts of Barack Obama and Britney Spears. Speaking of online guessing against Microsoft Accounts, Alex Weinert wrote in 2016 [20] about "more than 4 billion credentials we detected being attacked last year." He continued:

> *We detect more than 10 million credential attacks every day across our identity systems. This includes millions of attacks every day where the username and password are correct, but we detect that the person attempting to log in is a cyber-criminal.*

Akamai reports that 3.6 billion of 8.3 billion login attempts across the Akamai platform in November 2017 were online guessing attempts [21]:

> *In other words, 43% of all logins seen by Akamai were attempts to log in to an account using password guessing.*

Wordfence, a firm that offers protection to users of the popular WordPress blogging software, documents an average of over 20 million guesses per day in December 2016 [22]; they detail over 30k unique IP addresses being involved in the attacks. Freeman et al. describe a statistical approach to detecting logins that may be attacks (even though username and password are correct) at LinkedIn [23]. Twitter and Microsoft both proactively check user chosen passwords against blacklists of a few hundred of the most common passwords [7]; since this is a defense that primarily affects breadth-first guessing it is likely that this is in response to observed attack traffic. Florêncio et al. suggest a blacklist of 10^6 or so [7]. NIST updated their authentication guidelines in 2016 to suggest blacklisting very common password choices and throttling as a defense against this attack [24]. The solution that many might prefer, that passwords at long last be replaced by something better, appears optimistic [25]. Thus it appears this problem will be with us for some time.

An idealized model of authentication would have the server grant access when username and password are correct and deny it when they are not. The fact that some users choose passwords that are so simple that they will not withstand even 100 guesses and attackers can send high volumes of guessing traffic complicates this view. Providers are thus naturally driven to incorporate side information to improve decisions. Login requests accompanied by a previously-issued cookie, or from a familiar location or IP address might naturally be weighted higher. Logins from locations that appear anomalous might be weighted as more suspicious (this is the approach taken by Freeman et al. [23]).

Contributions: Our contributions are as follows. First, we show how we can robustly estimate the ratio of bad traffic to good among login requests. We show that we can use this estimate to identify the subsets of the request data that contain least (or even no) attack traffic. We then use these least-attacked subsets to estimate the distribution of values of any feature of interest over the unattacked data. We put the pieces together to calculate the odds that any particular request is malicious given our observations. To the best of our knowledge no previous work details how to estimate the amount of attack traffic at a site, or explains how to calculate the likelihood that an observation indicates attack traffic. Finally we perform a sensitivity analysis that shows that even if we fail to identify a subset with little attack traffic our odds ratio estimates are very robust. Since we explicitly estimate the bad-to-good traffic ratio, our approach has the advantage over fixed-threshold schemes (e.g., three strikes lockout and variants) that we avoid base rate neglect [26]. It has the advantages over machine learning schemes that it requires no labels, and does not assume stationarity of attack traffic.

II. Related work

In spite of the fact that all password-protected servers face this problem there appears very little work on protecting against online guessing attacks. As mentioned earlier, recommendations by organizations such as NIST [17], CESG [18] and OWASP [19] tend to be vague and non-actionable.

The most commonly referred to defense against online guessing is a "three strikes" type policy which locks an account after three consecutive unsuccessful attempts. There's been little formal analysis of the efficacy of this scheme. Brostoff and Sasse [27] examine actual login attempts at a large university and find that increasing from three to ten would have negligible effect on false negatives, while reducing unnecessary lockouts (false positives) considerably.

Other approaches include throttling the rate at which the server will respond to guesses against an account. A 1985 Department of Defense report recommends limiting the rate at which guesses will be served to between one per second and one per minute [28]. Florêncio et al. [29] propose a variant with exponentially increasing time between successive requests against a particular account. Pinkas and Sander [30] propose interjecting a Captcha challenge periodically to disrupt brute-force guessing attacks. Van Oorschot and Stubblebine [31] offer a simplification which greatly reduces the number of challenges presented to legitimate users. Alsaleh et al. [32] further enhance this approach. Despite the hold it has on popular imagination in 2010 Bonneau and Preibusch found that "three strikes" type rules to block accounts are uncommon [33]. Lu et al. [34] find the situation little changed in 2018: of 182 websites studied 131 either implemented no lockout and throttling mechanisms, or only ones that were easily evaded.

Approaches that involve account locking, or rate-limiting or serving Captcha challenges appear more suited to targeted guessing attacks than untargeted ones. The prevalence of this line of thinking may be due to the fact that untargeted attacks seem a relatively recent phenomenon; breadth-first attacks make most sense against services with very large user populations. Web services with hundreds of millions of users were probably not imagined when, e.g., the 1985 DoD guidance was written. The number of sites with user populations in the millions (which are thus attractive targets for breadth-first guessing) has grown rapidly in recent years.

Establishing reputation for an IP address has the potential to help with breadth-first attacks, if we assume that these are a scarce resource for attackers. Simple heuristics might involve blocking requests from an address if the number or fraction of failed attempts exceeds some threshold. Obviously it is important to count distinct fails rather than repeated attempts with the same wrong password (as might come from a poorly configured client with a cached password). While not much studied in the academic literature it appears likely that large providers use some variant. Several companies offer IP reputation lookup services (e.g., Webroot, Trend Micro, Cyren). This technique has the potential to catch attacks that evade account throttling. Even if dispersed among many accounts a high volume of guesses will generate many fails. Assuming that the attacker operates from an IP address pool that is a) relatively small an b) distinct from that of the legitimate user population we should be able to block significant amounts of traffic. The increasing use of botnets has undercut these assumptions somewhat. Many attackers have access to millions of IP addresses many of which co-exist with legitimate user traffic.

A. Binary classification and Machine Learning

The decision to allow or deny access (or block an IP address or other defensive action) is in principle a binary classification. If an observation were associated only with attack activity then our task would be easy. However, almost everything we can observe that may be indicative of attacker traffic is also present in legitimate traffic. For example, password-guessing attackers generate a lot of failed requests, but users also mis-type and mis-remember their passwords. So, failed login attempts certainly occur in legitimate traffic. Thus, we often have to deal with observations that may be more common in attack than legitimate traffic (or vice versa) but we have to deal with questions of degree rather than binary demarcation criteria between malicious and benign. Fails are almost certainly much more common in attack than legitimate traffic, but by how much? How should we use this fact to protect user accounts?

The common way of using an observation to decide between two possibilities is with a likelihood ratio test [35]. Let's call our observation x and suppose that we are trying to decide between malicious, mal, and non-malicious, $\overline{\text{mal}}$ based on x. The likelihood ratio test says that if

$$\frac{P(\text{mal}|x)}{P(\overline{\text{mal}}|x)} = \frac{P(x|\text{mal})}{P(x|\overline{\text{mal}})} \cdot \frac{P(\text{mal})}{P(\overline{\text{mal}})} > T$$

then we decide that the observation is malicious, and otherwise not. The threshold, T, controls the trade-off between false positives and false negatives.

This test clarifies the difficulty of simply associating some observation with attacker behavior. The likelihood ratio test shows that observing that $P(x|\text{mal})$ is high is not enough to allow us to conclude that the odds of being malicious are high. For example, suppose our observation is a failed login attempt where the attempted password is on a dictionary of the 100 most common passwords. This might account for 95% of malicious requests (i.e., $P(\text{Top-100 fail}|\text{mal}) = 0.95$) and only 0.5% or so of benign ones (i.e., $P(\text{Top-100 fail}|\overline{\text{mal}}) = 0.005$). However, we're still not in a position to decide whether the request is malicious or not since the ratio of bad traffic to good $P(\text{mal})/P(\overline{\text{mal}})$ might vary by orders of magnitude. Ignoring this factor is known as base-rate neglect [26].

The difficulty then in applying binary classification approaches is estimating the quantities in the likelihood ratio test. This will be our main task in Section IV.

An obvious approach is to use machine learning. Bonneau et al. suggest that traditional password authentication be augmented in this way [36]; that is, providers should incorporate other pieces of information (e.g., presence of cookie, familiarity of location, etc) when making a decision. Supervised machine learning approaches generally train using a fixed number of labelled samples. Freeman et al. describe a learning approach to identifying whether a successful request is actually from the user or an attacker [23].

The limitations of machine learning for security problems are explored by Sommer and Paxson [37]; while their emphasis is on intrusion detection many of the lessons are also applicable to our problem. In a recent book Chio and Freeman explore applications to spam and account hijacking [38]. While they do not cover password guessing explicitly some of the challenges they highlight are applicable. Two factors limit the applicability of machine learning to this problem: the absence of labels, and non-stationarity of the attack traffic.

First, we do not have an easy way of getting a training set of requests that are labelled benign and malicious. Second, a fundamental assumption of most supervised learning approaches is that the distributions from which the training data are drawn are representative of what will be seen in-the-wild, and stationary [39]. This may be true of the benign traffic: as the result of independent actions of a large population of users we can expect patterns to vary very slowly. However, we must expect extreme non-stationarity in the attack traffic. This is so because a) attackers deliberately change traffic patterns to confuse detection mechanisms, b) attackers may change strategy or have fluctuating resource availability, and c) malicious traffic is probably due to a small number of attackers (and thus will have much higher variance). Kelly et al. find that changing populations can have disastrous effects on classifier performance [39].

Thus, we cannot expect $P(x|\text{mal})$ to be even approximately stationary: the user-agents, IP addresses and attack strategy can change enormously. Equally, $P(\text{mal})/P(\overline{\text{mal}})$, the ratio of bad traffic to good can change enormously: 50% of traffic might be malicious at some times, and it might be only 1% at others. Some IP ranges might have no attack traffic, while at others it might be 100%.

We suggest that even extreme fluctuations in attack traffic are common and to be expected as attackers acquire, lose or re-direct resources. Further, any assumption of non-stationarity greatly eases the attackers' task. A supervised learning approach learns a classification boundary between benign and malicious traffic. To evade the defences the attacker need simply change along the input features. For example, the usefulness in making classification decisions of a feature like browser version can be greatly reduced by simply changing browser version with a high enough frequency that it ceases to be a good predictor.

The problem of classification where the distribution changes has been studied under the name "concept drift" in the statistical learning literature. Gama et al. [40] give a survey of the approaches. These techniques have been used with some success for the detection of slowly-evolving credit-card fraud [41]; however they do less well when faced with extreme non-stationarity.

III. Overview of the problem

An authentication request should contain at least a timestamp, the IP address and useragent of the requesting client, the username and the submitted password. On receiving the request the server retrieves the salt and salted-hash associated with the username. If the salted-hash of the submitted password matches the stored salted-hash then authentication proceeds, but is otherwise rejected. A log of the request will be written and might contain:

[`Time, IP, UserAgent, Username, Outcome`].

Note that the submitted password is not logged; it is important that it not leave the volatile memory of the server. The reason, obviously, is that storing the submitted password would compromise all accounts if an attacker gained access to the logs.

A. Absence of labels

The stream of arriving authentication requests is generally an unknown mixture of benign and malicious traffic. There will be failures and successes among both the legitimate and malicious traffic. If we had a labelled training set an obvious approach would be to use supervised learning. Unfortunately getting reliable labels, even for a small subset of the traffic, appears impossible, as Chio and Freeman explain [38]:

> *Unlike spam, which can be given to a human to evaluate, there is no reasonable way to present an individual request to a reviewer and have that person label the request as bot or not.*

A blog posting from the Uber accounts team [42] finds similarly: "when we look at something like unauthorized login attempts [...] we never know the ground truth." This is meaningfully different from many other classification problems. Human graders can reliably classify spam, or cat versus dog images, or inappropriate content. While expensive, for these problems we can have humans label a very small set and leverage this for training. However, the barrier to labelling authentication requests is more fundamental than one of simply cost or scale. Even if we had unlimited supply of human reviewers we lack any basis on which to instruct them to decide if a given time, IP, UserAgent etc, tuple is malicious.

The inability to get labels obviously rules out supervised learning approaches. It also means, however, that we will be unable to compute metrics such as accuracy, true and false positives, precision and recall, etc. This greatly complicates the question of evaluation. Since they lack labels, unsupervised approaches to classification must make assumptions about the classes and/or how they differ [43] [44] [45]. The better the assumptions match reality, of course the better the performance will be, but without labels we can't directly measure how well an assumption holds.

A simple example may illustrate the difficulty and the importance of choosing assumptions conservatively. A basic approach to defending against breadth-first guessing with IP blocking might be of the form:

`if (<condition>) then` {`block IP address`},

where `<condition>` is a rule that attempts to cover as much malicious and as little benign traffic as possible. Example conditions might be:

A: Sends 10 consecutive failed requests.

B: Has success rate $< 50\%$ over the last 100 requests.

Clearly A and B are not equivalent: some requests blocked by A will be allowed by B and vice versa. Thus, the true and false positive rates they achieve for any particular traffic set will be different. However, even if we had access to the logs of an attacked server it is not possible to say which will do better. To do so we would need to know how likely benign and malicious traffic were to trigger conditions A and B. This in turn requires knowing if attacks come from a small pool of IP addresses, and to what degree attack and benign traffic originate from the same addresses. If an attacker has only a small number of IP addresses which are completely disjoint from those of benign users then both A and B might be very effective. However, if attackers have access to large IP pools (as sometimes appears to be the case [22] [21]) and both benign and malicious traffic can come from some addresses (e.g., botnets on consumer machines, proxies, VPNs etc) they will work far less well. For example, a university in Florida might do very well with heuristics such as A and B protecting against an attacker with no access to IP addresses inside US; it would do far less well against a student with bots installed on various campus and student-housing machines. For a global service like Facebook it is probably unrealistic to expect attack and benign traffic to come from entirely disjoint IP pools. Thus, all we can really say is that for some assumptions of attack traffic A will do better than B and for others it will do worse. There is a parallel here with the no-free-lunch (NFL) theorem for machine learning which states that every learner must incorporate assumptions beyond the labelled data [44] [45]. That is, all learners have equivalent performance when the off-training-set loss is integrated over all possible datasets.

Thus we have a quandary. Without labels we have no choice but to rely on assumptions for our block decisions. However, without labels we also can't compute the metrics needed to test how well those assumptions are doing. Further, many of the suggestions to addressing credential-guessing involve assumptions about attacker traffic and capabilities, which are notoriously volatile and subject to change. Some attackers might have small pools of IP addresses while others might have many; some attackers might rotate through a predictable set of browser UserAgents while others are more sophisticated. Assuming that a pattern seen in attack traffic in one place will hold more generally seems very brittle, unless that pattern represents something that is inherent to attack traffic. Hence, anything we might learn from a particular set of logs can't necessarily be relied upon elsewhere.

Given these difficulties, in this paper we make only one assumption about attack traffic:

A1: *Malicious logins are a small fraction of total logins.* This should be inherently true of guessing attacks. (The same is of course not true for an attacker exploiting password re-use or other non-guessing approach.) First, any service for which it is not true (i.e., legitimate logins do not heavily outnumber malicious ones) would appear to offer little utility to its users. Second, even in the best case the attacker will have a per-guess success rate of about 0.1% [11]; if legitimate requests succeed 95% of the time, and traffic is a 50/50 mix of malicious and benign then benign logins would outnumber malicious ones by more than 950:1.

Note that this is the only substantive assumption we make about attack traffic. We do not assume that attack traffic is stationary or independent across features. We don't assume that attacks come from a limited IP address pool. We don't assume that patterns observed in malicious traffic at one provider at a certain time will be seen at other providers, or at the same provider at other times.

B. Our attack on the problem

Our attack on the problem is as follows. We'll assume that x is a categorical feature and that we wish to estimate:

$$\frac{P(\text{mal}|x)}{P(\overline{\text{mal}}|x)} = \frac{P(x|\text{mal})}{P(x|\overline{\text{mal}})} \cdot \frac{P(\text{mal})}{P(\overline{\text{mal}})} \triangleq \Theta(x) \cdot \Psi. \tag{1}$$

This is the odds that observing a particular value of x implies that the request is malicious [35]. This might allow us to decide to lock an account, block requests from an IP address or other defensive action. We note that the observed distribution of x is the sum of the distributions in the benign and malicious traffic:

$$P(x) = \alpha \cdot P(x|\overline{\text{mal}}) + (1-\alpha) \cdot P(x|\text{mal}), \tag{2}$$

where $0 \leq \alpha \leq 1$. The problem is that we don't know $P(x|\overline{\text{mal}})$, $P(x|\text{mal})$ or α. While the malicious traffic is non-stationary, we expect the benign traffic, $P(x|\overline{\text{mal}})$, as the aggregate of the independent actions of a very large number of users, to vary slowly, if at all. We first show how we can estimate the second term in (1), the ratio of bad traffic to good: $\Psi \triangleq P(\text{mal})/P(\overline{\text{mal}}) = (1-\alpha)/\alpha$. Using this we can then identify subsets where there is very little attack traffic (i.e., $\alpha \approx 1$) and thus the observed distribution closely approximates that of the benign traffic, i.e., $P(x) \approx P(x|\overline{\text{mal}})$. This then allows us to estimate the first term in (1) by subtraction:

$$\Theta(x) \triangleq \frac{P(x|\text{mal})}{P(x|\overline{\text{mal}})} = \frac{P(x) - \hat{\alpha} \cdot \hat{P}(x|\overline{\text{mal}})}{(1-\hat{\alpha}) \cdot \hat{P}(x|\overline{\text{mal}})}.$$

Here we are using the ˆ symbol to represent an estimate.

To avoid encumbering the notation we present our approach for a single categorical feature. Extension to multiple independent features is trivial. We represent the two component ratios of the likelihood ratio test as Ψ and $\Theta(x)$. Ψ is the ratio of bad-to-good traffic (often called the pre-observation odds); $\Theta(x)$ is the ratio of how common x is in the malicious traffic over how common it is in legitimate traffic (often called the likelihood ratio). The product, $\Theta(x) \cdot \Psi$, is often called the post-observation odds. We perform a sensitivity analysis in Section V to determine how our estimates will change assuming that we mis-estimate some of the parameters along the way.

The only assumption we make about the difference between malicious and benign traffic is **A1**. In addition we make the following assumptions about benign traffic:

A2: *Rate of login failures from benign users is independent of feature categories.*

A3: *Distribution of features in the benign traffic is independent:* $P(x, y|\overline{mal}) = P(x|\overline{mal}) \cdot P(y|\overline{mal})$ *and slowly-varying.*

Assumption **A2** says that the benign failure rate should be approximately constant when users are grouped by features such as browser version, IP address pool, time, or username. For example, we expect that the benign users of Chrome 64.0.3282.119 to fail at the same rate as the users of Firefox 27.3; we expect the benign traffic from one particular range of IP addresses to fail at the same rate as others. We are not assuming the same of the malicious traffic, and we are not assuming the same of the observed traffic (i.e., the mix of benign and malicious in (2)).

A2 and **A3** can be viewed as restrictions rather than assumptions; i.e., we can restrict ourselves to features where they hold. For example, if we believe that browser version does have a dependence on time of login in the benign population we can throw out one or other of those features. Alternatively, we can restrict attention to blocks of requests where we have higher confidence independence will hold. For example, the benign failure rate on mobile devices is different from that on regular keyboards; the simplest approach is to treat the two categories of devices separately; see Section VI-B. Equally, the distribution of browser version almost certainly varies somewhat by country. Treating each country separately reduces these effects on **A3**.

Two additional limitations are worth noting. First, this approach requires large scale. While the method to estimate the benign to malicious traffic ratio should work for smaller sites, accurate estimation of the odds requires on the order of millions of benign users. If we divide our data by categorical features we require about many requests per category to get reasonably tight confidence intervals in (4). This implies the approach works best at a scale of millions or tens of millions of users. For sites that have only thousands or users the confidence intervals are likely to be too wide. Second, this approach describes how we may accurately estimate the odds that an observation x is malicious, but if the distribution of x in benign and malicious traffic are similar the odds may not be useful. That is, if $P(x|\text{mal}) \approx P(x|\overline{\text{mal}})$ the odds may not be useful in making a decision. For example, if the same fractions of benign and malicious traffic have useragent Chrome 64.0.3282.119 then this will not be useful in separating the two traffic types. This is, of course, a limitation of all statistical approaches: features of the attack traffic that are drawn from the same distribution as the benign traffic won't be useful to discriminate between the two.

IV. Estimating the test statistic

A. Estimating $P(mal)/P(\overline{mal})$

Consider an authentication server which handles login requests. Over any collection of requests it sees L successful logins and F failed attempts. We assume that there's an unknown mix of legitimate and malicious traffic: the combined number of benign and malicious logins is

$$L = L_b + L_m$$

and the combined number of benign and malicious fails is

$$F = F_b + F_m.$$

We first wish to estimate the last term in the likelihood ratio test, the ratio of bad traffic to good, $\Phi = P(\text{mal})/P(\overline{\text{mal}}) = (1-\alpha)/\alpha = (L_m + F_m)/(L_b + F_b)$, but can observe only L and F.

Consider the ratio of fails to logins. Under our assumption, **A1**, that malicious logins are a small portion of the benign ones (i.e., $L_m/L_b \approx 0$) we get:

$$\begin{aligned} \frac{F}{L} &= \frac{F_b + F_m}{L_b + L_m} = \frac{F_b/L_b + F_m/L_b}{1 + L_m/L_b} \\ &\approx \frac{F_b}{L_b} + \frac{F_m}{L_b}. \end{aligned} \tag{3}$$

We detailed our justification for this assumption in Section III: if legitimate requests succeed 95% of the time, and traffic is a 50/50 mix of malicious and benign then $1 + L_m/L_b = 1.0011 \approx 1$.

Next observe that benign traffic is the aggregate of a large number of users (at a large provider millions or even hundreds of millions) acting independently. For a large enough number of requests, $F_b/L_b \approx$ const. This is so because we expect login failures from legitimate users to happen independently at some rate p. These should happen because of typos, misremembered passwords or usernames etc, which we assume happen independently across the population. So, a total of $L_b + F_b$ benign requests should result in $p \cdot (L_b + F_b) = F_b$ failures. Thus, using the well-known normal approximation to the binomial distribution [46], we can have 95% confidence that an approximation of the benign failure rate $\hat{p} = F_b/(L_b + F_b)$ will lie in the interval

$$p \pm 1.96 \cdot \sqrt{\frac{p \cdot (1-p)}{L_b + F_b}}. \tag{4}$$

For example, if $p = 0.07$ then 95% of the time we would expect an approximation $\hat{p} = F_b/(L_b + F_b)$ to fall within ± 0.00158 of the true value of p for $L_b + F_b = 100,000$ and within ± 0.000158 for $L_b + F_b = 10^7$; so we're within 0.226% of the true value of p with 10 million benign requests. Since p is small the confidence interval for $F_b/L_b = \hat{p}/(1-\hat{p}) \approx p/(1-p) =$ const. will be fractionally higher. (A value of $p = 0.07$ is broadly in line with the value that Chatterjee et al. [47] measured for typos, but we will show how to estimate it below).

Thus, the ratio of benign fails to logins should be fairly constant over large enough collections of authentication requests. Hence, if we divide our data into large-enough subsets of requests, we get for any given subset:

$$\frac{F(k)}{L(k)} \approx c + \frac{F_m(k)}{L_b(k)} \tag{5}$$

where $c = p/(1-p)$ and k is just an index over subsets.

The subsets can be any grouping of the login request data so long as they have enough entries to ensure that the confidence interval in (4) is small relative to p. For example, the subsets can have request data from particular time slots, requests that come from particular IP-address ranges, requests with particular user-agent strings, or requests against particular sets of accounts. Combinations of those things are also possible, e.g., we could form a subset of request data during a particular time slot, from a particular range of IP addresses against a certain collection of accounts. We'll take subsets that are grouped by time intervals (e.g., hour or day) as well as by features such as browser and IP address; to avoid unnecessarily encumbering the notation we don't explicitly add an index for time.

Note, we are assuming that the benign failure rate p is independent across features; that is, $F_b(k)/L_b(k) \approx$ const. for each of the subsets. For example, we expect that the benign users of Chrome 64.0.3282.119 to fail at the same rate as the users of Firefox 27.3; we expect the benign traffic from one particular range of IP addresses to fail at the same rate as others. We are not assuming the same of the malicious traffic, and we are not assuming the same of the observed traffic (i.e., the mix of benign and malicious in (2)). We postpone the question of mobile devices and proxies (which may have different traffic and failure patterns) until Section VI-B.

Now (5) is interesting because it tells us that variations (across subsets or over time) in the ratio of observed failures to observed logins, $F(k)/L(k)$, are due to malicious traffic. That is, if a subset has no malicious traffic, $F_m(k) = 0$ and then $F(k)/L(k) = c = p/(1-p)$. Further, since $F_m(k)$ must be positive, malicious traffic can cause increases in $F(k)/L(k)$ above the baseline value c, but cannot cause decreases.

Suppose then that we knew the value of the constant in (5). We then have a straightforward way to estimate the malicious traffic in a particular subset:

$$F_m(k) = L_b(k) \cdot \left(\frac{F(k)}{L(k)} - c \right) \approx F(k) - c \cdot L(k). \quad (6)$$

So we could estimate the amount of malicious traffic in a timeslot, or from a range of IP addresses, or from a particular user-agent if we can determine this single number $c = p/(1-p)$. This value should also be very stable over time: we do not expect that rate at which users mis-type and mis-remember passwords etc, to change significantly over time. (We examine the question of traffic containing a mixture of types with different failure rates, e.g., mobile and desktop, in Section VI-B). Hence, we have a potentially accurate estimate of the amount of malicious attack traffic if we can determine this single constant c, which at least in principle, need only be measured once. For example, as a thought-experiment, if attackers were to "take a day off" and pause all malicious traffic then we could get a value for $c = F(k)/L(k)$ for all K of the subsets on that day. We could use this estimate for months or even years to come. In practice, of course we would want to re-estimate c regularly to confirm that nothing is changing, but we can do a great deal with (6) if only we can find c. Also, the larger the subset that we use the smaller is the confidence interval in (4). Hence, there's a significant advantage to having large subsets.

From (6) the ratio of bad traffic to good for any subset that we need in the likelihood ratio test (1) can now be easily estimated as

$$\begin{aligned} \frac{P(\text{mal})(k)}{P(\overline{\text{mal}})(k)} &\approx \frac{F_m(k)}{L_b(k) + F_b(k)} \\ &= \frac{F(k) - L(k) \cdot c}{L(k)(1+c)}. \quad (7) \end{aligned}$$

This involves only quantities that we observe directly (i.e., $F(k)$ and $L(k)$) and $p = c/(1+c)$ (which we have yet to estimate).

Note that in an abuse of notation we've indexed the probabilities $P(\text{mal})(k)$ and $P(\overline{\text{mal}})(k)$ by subset k. This is so since the malicious to good traffic ratio can (and in general will) vary enormously from subset to subset. For example, traffic from some IP addresses may be 100% malicious, while that from others may be 100% benign, and many will contain some mix.

B. Estimating $c = p/(1-p)$ and $P(x|\overline{mal})$

The potential to estimate the ratio of bad traffic to good is promising, but we still need an accurate estimate of the constant $p/(1-p)$. As noted earlier there are several components to the benign failure rate p. One possibility is that we might measure these individual components. That is, perform a measurement study to estimate the rate at which users enter either password or username incorrectly. Chatterjee et al. [47] offer a measurement of one of these components: they estimate failures due to password typos at 4.5% overall and 9% from touchscreen keyboards. It's likely that the rates at which benign failures occur are somewhat stable across sites, however we wish to leave open the possibility that they vary. For example, it is likely that a tax-related site where people login only once a year or so will have a much higher benign failure rate than a social networking or email site where a majority login every day. Sites that force mandatory password expiration every 90 days likely have higher fail rates than those that do not (and this might be time-dependent if those changes are synchronized across users). Also, we can expect benign failures that come from mobile clients to occur at a very different rate from clients that have a conventional keyboard. The mix of traffic from mobile to conventional clients is probably very site dependent (e.g., many people probably access facebook or twitter using apps on their phone, while it is likely that few people access a government tax site in this way). Finally, users may feel more reticent to allow a browser to auto-fill credentials for a bank site than for a video-streaming service or online newspaper; this might make password submission more likely to be automated and thus less subject to error. For all of these reasons it seems preferable (if possible) to estimate p on a per-site basis and to update this estimate periodically.

Clearly, from (5), malicious traffic can increase $F(k)/L(k)$ above c but not decrease it. Thus, a simple upper-bound estimate for our constant is:

$$\hat{c} \triangleq \min_k \frac{F(k)}{L(k)} \geq c. \quad (8)$$

Here the minimization is taken over all subsets large enough to satisfy our assumption that the confidence interval in (4) is small enough to ensure that F_b/L_b is approximately constant. We'll examine subsetting strategies in Section IV-C below. Obviously, once we have this we can estimate $\hat{p} = \hat{c}/(1+\hat{c})$.

Call the index of the subset of our data that achieves the minimum k_0. Note from (5) that k_0 is the subset, where $F_m(k_0)/L_b(k_0)$ is minimized. It's easy to show that $F_m(k)/L_b(k)$ is monotonically decreasing with $\alpha(k)$ and thus this subset contains the least amount of attack traffic among all the subsets. Hence, in finding our estimate $\hat{c}$ we've also found an estimate for the distribution of x over the benign traffic:

$$P(x|\overline{\text{mal}}) \approx P(x)(k_0). \quad (9)$$

Note that we've dropped the k-index since, by assumption **A3**, the distribution of x in the benign traffic is stationary and independent of features such as browser version, IP address and time.

Observe that (8) is generally an over-estimate (unless we find a subset with no attack traffic). If we over-estimate c then we wrongly consider some malicious traffic as benign. This means that we under-estimate the numerator and over-estimate the denominator in $P(\text{mal})/P(\overline{\text{mal}})$. Thus our estimate is conservative in the sense of erring toward considering traffic benign rather than malicious. We examine the sensitivity of our estimates in greater detail in Section V.

C. Subsetting strategies

The request traffic can be grouped into subsets many different ways depending on the features logged by the server. We'll consider subsets consisting of time slots, IP address ranges, and particular user-agents and particular collections of accounts. Other features may be recorded, but we assume most authentication servers log at least this set of features.

Some of the features are entirely under attacker control. Useragent or browser version can be chosen without constraint. IP addresses, on the other hand, are constrained by those the attacker controls; it is very likely that there will be large blocks of IP address ranges from which she sends no traffic. The account attacked lies somewhere in between: it's quite likely that the attacker does not have a complete and accurate directory of every single account at a service. Thus, it's likely that she has a list of accounts that she attacks; accounts not on any attackers list might see no attack traffic at all.

The goal in Section IV-B was to find the subset with the least amount of attack traffic while still being large enough to keep the confidence interval in (4) small. If we divide into very small subsets the likelihood that one or more see very little attack traffic is increased, but as the subsets become smaller the confidence intervals around our estimates becomes larger. On this point services with massive user-bases have a clear advantage. A service with 100 million users might have a million logins per day on a particular version of a particular browser (e.g., Chrome 64.0.3282.119); if this version happens not to be used by any attacker this would suffice to estimate c within a very narrow confidence interval. On the other hand, at a small university with only 10,000 students that same browser version might see only 100 logins per day, which is nowhere near enough to give a narrow confidence interval in (4). The university in this case might choose a much longer time interval.

Consider the particular case of dividing the data into subsets by account name. As explained above, we assume that while some accounts will be attacked a great deal some will almost certainly see very little or no attack traffic. One possible subsetting strategy is as follows. Choose a time interval and a number of subsets K_u such that the average number of successful logins per subset is large enough to make the confidence interval in (4) small. For example, if a service has 14 million logins per day we might choose our time interval to be one day and make $K_u = 20$; this gives approximately 700k benign requests per subset, so that we'll be able to estimate p with about 1% accuracy (i.e., if p were 0.05 we'd have $\hat{p} = 0.05 \pm 0.0005$).

Now assign accounts to demi-decile subsets (i.e., each has $1/20$ of the total) ordered by the increasing ratio of fails to logins F/L on one day. That is, the $k = 0$ subset has the 5% of the accounts that had the lowest F/L ratio, while the $k = K_u - 1$ subset has the 5% of accounts where this is highest. We can now use these subsets to estimate c as in (8). In sorting by F/L we increase the dissimilarity of amount of attack traffic between subsets (e.g., with respect to a random assignment). Obviously, we must do the estimation over a different time interval from the one used to assign the accounts into bins. For example, if we assigned accounts to subsets based on sorting over the F/L ratio on day-0, we would use days-$1, 2, 3, \cdots$ to estimate c. That is, we wish to be sure that $F(k)/L(k)$ is low because of low attack volume and not because of selection bias.

This subsetting exercise can be repeated for other features such as IP address, browser version, and any others available. All that is required to get an accurate estimate of c is that at least one of these subsets receives little or no attack traffic and is big enough to make the confidence interval in (4) small. Of course we cannot guarantee that we will ever find a subset entirely free of attack traffic. However, the sensitivity analysis of Section V will show that even imperfect estimates are very useful in getting the likelihood ratio.

Here, again, services with very large user populations appear to have a significant advantage. A service with hundreds of millions of users might be able to divide accounts (or IP addresses or browser versions) into 1% subsets while preserving narrow confidence intervals in a single day's worth of requests. It would then be possible to estimate c almost precisely if even a single 1% of accounts (or IP addresses or browser versions) see no attack traffic for a single day.

The assumption that some portions of account and IP address space may be free of attack traffic does not seem strong (since the attacker is constrained by addresses available and accounts that she knows of). While the useragent has fewer constraints, some popular password guessing tools, such as THC-Hydra, offer only limited ability to randomize user-agents in the traffic generated.

Unless the distribution of attack traffic across time, IP address ranges, user-agents and accounts is identical to those distributions in the legitimate traffic then some subsets will contain more attack traffic than others. The subset with the least attack traffic will be the one that minimizes (8) and thus gives the tightest bound on c.

Importantly, if we find an unattacked subset we can have high confidence that we have done so. In a subset that is free of attack traffic the value of $F(k)/L(k)$ should remain constant over time and give us the actual value of c. For example, with a time interval of one day, having $F(k)/L(k)$ vary within the margin of error for a period of weeks is a good indication that subset k is unattacked (or that the attack traffic does not vary over time in this subset). If several subsets yield the same value of $F(k)/L(k)$ then either they are all unattacked or the bad to good ratio $(1 - \alpha(k))/\alpha(k)$ is the same in all of them. Naturally, our confidence that we're close to the true value of c increases greatly if we get several subsets independently corroborating the same value. For example, if several user-agents are unused by an attacker, some accounts are unattacked, and she has access to only a portion of the IP address range then we will have independent measurements converging on the true value of c.

D. Estimating $P(x|mal)/P(x|\overline{mal})$

We've seen how we can estimate $P(\text{mal})/P(\overline{\text{mal}})$ over subsets. Since our goal is to use the likelihood ratio test (1) it remains to find $P(x|\text{mal})/P(x|\overline{\text{mal}})$. We show now how this can be calculated.

We already saw in (9) how to estimate $P(x|\overline{\text{mal}})$. Now recall from (2) that any subset that contains a mixture has proportions $(1-\alpha(k))$ and $\alpha(k)$ of attack and benign traffic respectively:

$$P(x)(k) = \alpha(k) \cdot P(x|\overline{\text{mal}}) + (1-\alpha(k)) \cdot P(x|\text{mal})(k).$$

(Note that we drop the k index for $P(x|\overline{\text{mal}})$ since it is constant across subsets). Equation (7) gives the ratio of bad to good traffic. This allows us to solve for $\alpha(k)$:

$$\frac{P(\text{mal})(k)}{P(\overline{\text{mal}})(k)} = \frac{1-\alpha(k)}{\alpha(k)}. \tag{10}$$

So, we can now calculate $\alpha(k)$ for any subset assuming that we have first estimated $P(\text{mal})(k)/P(\overline{\text{mal}})(k)$. That is:

$$\alpha(k) = \frac{1}{1 + P(\text{mal})(k)/P(\overline{\text{mal}})(k)}, \tag{11}$$

where $P(\text{mal})(k)/P(\overline{\text{mal}})(k)$ is calculated from (7).

Now, the malicious portion of the traffic is:

$$P(x|\text{mal})(k) = \frac{P(x)(k) - \alpha(k) \cdot P(x|\overline{\text{mal}})}{(1-\alpha(k))}.$$

In words: the malicious traffic (left-hand side) is what we have observed minus the benign traffic. The benign traffic in this particular subset is $P(x|\overline{\text{mal}})$ weighted by $\alpha(k)$. So now the overall odds become (using $\hat{}$ notation for quantities that are estimated rather than observed directly):

$$\Theta(x)(k) \triangleq \frac{P(x|\text{mal})(k)}{P(x|\overline{\text{mal}})} = \frac{P(x)(k) - \hat{\alpha}(k) \cdot \hat{P}(x|\overline{\text{mal}})}{(1-\hat{\alpha}(k)) \cdot \hat{P}(x|\overline{\text{mal}})}, \tag{12}$$

where $\hat{\alpha}(k)$ is an estimate of $\alpha(k)$, etc.

Extending to multiple features is trivial under the common assumption of independence; the likelihood ratio is simply expanded as the product of the individual probabilities. Thus, we can estimate the odds that a request is malicious given all of the independent observations available. We might get the odds that a request at a particular time of day, from a particular IP address range, from a particular browser version is malicious for example. Combining multiple features is, of course, particularly valuable when we have multiple weak indicators of attack, rather than a single strong one.

E. Overall algorithm

So our entire estimation procedure is as follows. We select a time-interval and break our request data into K subsets using, e.g., the strategy of Section IV-C. These should be such that the confidence interval in (4) can be expected to be small. This naturally depends on the number of users and logins per unit time and will vary enormously from service to service.

For pre-processing, we first calculate the quantities that pertain to the benign distribution. These should be stable and do no need to be re-estimated often.

```
1: Estimate $\hat{c}$ from (8),
2: Calculate $\hat{p} = \hat{c}/(1+\hat{c})$
3: Estimate $\hat{P}(x|\overline{\text{mal}})$ from (9)
```

Now for a received request that belongs to subset k calculate (over the time interval that trails backward from the received request):

```
4: Estimate $\hat{\Psi}(k) = \hat{P}(\text{mal})(k)/\hat{P}(\overline{\text{mal}})(k)$ from (7)
5: Solve for $\hat{\alpha}(k)$ using (11)
6: For each value of categorical feature $x$: estimate
       $\hat{\Theta}(x)(k) = \hat{P}(x|\text{mal})(k)/\hat{P}(x|\overline{\text{mal}})$ from (12)
7:     Finally, calculate the post-observation odds as the
       product: $\hat{\Theta}(x)(k) \cdot \hat{\Psi}(k)$.
```

This allows calculation of the likelihood that any particular request is malicious. There are many different ways of acting on this information. The most obvious use might be, on a per-request basis, to deny those requests for which the likelihood exceeds a threshold and allow all others. Additionally a service might keep a running average of the odds of requests being malicious for a particular account or from a particular IP address. Accounts with a high average might be flagged for audit or additional protection. Traffic from IP addresses with a high average might be treated differently, e.g., subjected to Captcha challenges.

F. Toy example

We present a toy example to demonstrate the simplicity of the approach. We stress that these are "made-up numbers" and for illustration only.

Suppose that 25% of the request traffic at a service is malicious. Assume, however, that only 80% of accounts are known to the attacker (i.e., 20% are not on her list). We use the strategy in Section IV-C to group into ten subsets by account; we sort by the fail-to-login ratio and each subset contains 10% of logins. Clearly then two of the subsets will be free of attack traffic. The actual value will be $\Psi(k) = 0$ for two subsets and $\Psi(k) = (0.25/8)/(0.75/10) = 0.4167$ for the remaining eight. This leads to an estimate $\hat{c}$ that has 95% chance of being within one margin of error of c, and it allows us to accurately estimate the bad-to-good traffic ratio $\Psi(k)$ for $k = 2, 3, \cdots, 9$ in the 8 subsets of accounts that do receive attack traffic (using (7)). We also get an accurate estimate $P(x|\overline{\text{mal}})$ for any feature x of interest (from (9)).

Consider the simple binary feature mentioned in Section II-A: whether or not a failed request is on the list of 100 most common passwords. Armed with an accurate estimate of $P(\text{Top-100 fail}|\overline{\text{mal}})$ from one of our subsets of unattacked accounts we can accurately calculate $\Theta(\text{Top-100 fail})(k)$. For example, if 97% of the attacker's guesses are among the Top-100, but only 0.5% of benign failures are then $\Theta(\text{Top-100 fail})(k) = 0.97/0.005 = 194.0$.

Thus, the odds of a request that fails with a Top-100 password being malicious will be

$$\Theta(\text{Top-100 fail})(k) \cdot \Psi(k) = 194 \times 0.4167 \approx 80.8$$

if the account is in one of the subsets that receives attack traffic, while it will be

$$\Theta(\text{Top-100 fail})(k) \cdot \Psi(k) = 194 \times 0 = 0$$

if it is not.

V. Sensitivity Analysis

A main source of error is the possibility that we estimate c from a subset that is not entirely benign, and contains some attack traffic,. We can see how this affects the value that we get for $\hat{c}$ and then examine how that inaccuracy ripples through to the other quantities. Let's consider the case where we estimate c over a bin with $\alpha(k_0) \neq 1$ so that we end up with an over-estimate.

When we mistakenly believe that we've found an un-attacked bin we use (8) and instead of c we estimate

$$\hat{c} = c + \frac{F_m(k_0)}{L_b(k_0)} = c + \frac{1 - \alpha(k_0)}{\alpha(k_0)} \cdot (1 + c). \tag{13}$$

For each k, we next get, from (7), that:

$$\begin{aligned} \hat{\Psi}(k) &= \frac{\hat{P}(\text{mal})(k)}{\hat{P}(\overline{\text{mal}})(k)} = \frac{F(k)}{L(k) \cdot (1 + \hat{c})} - \hat{p} \\ &= (\Psi(k) + p) \cdot \frac{1 + c}{1 + \hat{c}} - \hat{p} \\ &= (\Psi(k) + p) \cdot \alpha(k_0) - \hat{p}. \end{aligned} \tag{14}$$

The equality $\alpha(k_0) = (1 + c)/(1 + \hat{c})$ used in the last line follows from (13). From this we estimate $\hat{\alpha}(k) = 1/(1 + \hat{\Psi}(k))$; i.e., use $\hat{\Psi}$ instead of Ψ in (11). Observe, for a given value of p, and estimate $\hat{p}$, that $\hat{\Psi}(k)$ depends on only the single independent variable Ψ (since $\alpha(k_0)$ can be calculated from $\hat{p}$ and vice versa). Thus, we can graph $\hat{\Psi}(k)$ as a function of $\Psi(k)$, which we do in Figures 1(a) and 2(a) below.

Finally, we estimate, from (12):

$$\begin{aligned} \hat{\Theta}(x)(k) &= \frac{P(x)(k)}{\hat{\alpha}(k) \cdot \hat{P}(x|\overline{\text{mal}})} - 1 \\ &= (\Theta(x)(k) + 1) \cdot \frac{\alpha(k)}{\hat{\alpha}(k)} \cdot \frac{P(x|\overline{\text{mal}})}{\hat{P}(x|\overline{\text{mal}})} - 1 \\ &= \alpha(k_0) \cdot \frac{\Theta(x)(k) + 1}{\alpha(k_0) + (1 - \alpha(k_0)) \cdot \Theta(x)(k_0)} - 1. \end{aligned}$$

The last line follows from the (easy to show) fact that $\alpha(k_0) = \alpha(k)/\hat{\alpha}(k)$. Note, for a given value of p, and estimate $\hat{p}$, that $\hat{\Theta}(x)(k)$ involves two independent variables: $\Theta(x)$ and $\Theta(x)(k_0)$. Recall that $\Theta(x)(k_0) = P(x|\text{mal})(k_0)/P(x|\overline{\text{mal}})$ is the ratio of how common x is in the attack traffic of subset k_0 to how common it is in the benign traffic. If a value of x is more common in attack traffic of subset k_0 than benign traffic then $\Theta(x)(k_0) > 1$ and if it's less common then $\Theta(x)(k_0) < 1$. Obviously, since both $P(x|\text{mal})(k_0)$ and $P(x|\overline{\text{mal}})$ sum to 1 we can't have that $\Theta(x)(k_0)$ is always greater than, or always less than 1. In the figures below, we'll use a range of values for $\Theta(x)(k_0)$ centered about 1. Thus, for a particular over-estimate $\hat{p} > p$ we can graph $\hat{\Theta}(x)(k)$ as a function of $\Theta(x)(k)$ by including curves for different values of $\Theta(x)(k_0)$.

Example 1: Suppose that $p = 0.07$[1] but we fail to identify a subset free of attack traffic and substantially over-estimate $\hat{p} = 0.0826$ (i.e., our estimate of p is off by almost 20%). Figure 1 (a) shows $\hat{\Psi}(k)$ as a function of $\Psi(k)$ for this particular over-estimate. Observe that for small attack volumes (e.g., $\Psi(k) < 0.05$) the estimate is inaccurate, but it becomes very accurate as the amount of attack traffic increases. This means that in subsets that contain $\geq 10\%$ or so attack traffic, that our estimate of the bad-to-good traffic ratio Ψ is extremely accurate even though we made a grossly inaccurate estimate of p. Figure 1 (b) shows $\hat{\Theta}(k)$ for the same choices of p and $\hat{p}$. The different curves show a range of values for $\Theta(x)(k_0) \in \{0.25, 0.5, 1, 2, 4\}$. In the absence of assumptions about the attack traffic in subset k_0 the value $\Theta(x)(k_0) = 1$ is most representative. Again, observe that the estimate is good when $\Theta(x)(k) \geq 0.1$. That is, we get a very accurate estimate of the true odds when x is at least $1/10$-th as common in attack traffic as it is in benign even though we made a grossly inaccurate estimate of p.

Example 2: Suppose, again, that $p = 0.07$ and we estimate somewhat accurately $\hat{p} = 0.0732$; the error is 2× the 95% confidence interval that we'd expect with 100k benign samples. Figure 2 shows how the estimates $\hat{\Psi}(k)$ and $\hat{\Theta}(x)(k)$ evolve as functions of the actual ratios. Observe that the estimates are extremely accurate (in the case of $\hat{\Theta}(x)(k)$ for all values of $\Theta(x)(k_0)$). Thus, if we identify an unattacked subset we get an extremely accurate estimate of the true odds. For large sites, where we might easily have millions of benign requests per time interval, we should be able to have great confidence in our estimates.

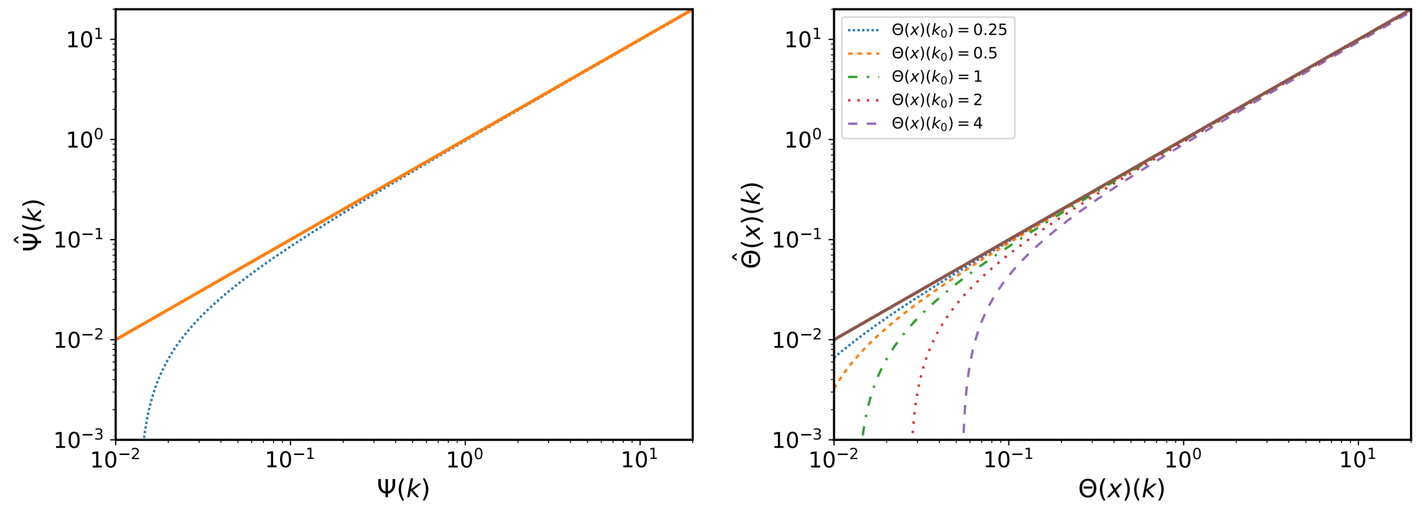
VI. Discussion

A. Limitations

This approach requires large scale. The more accurate the estimate of c (and hence p) the better our final estimate of the odds ratio will be. A large site has several advantages in getting an accurate estimate of c. First, it may be able to divide its user population (or other feature) into fractions that are 5%, or 1%, or 0.1% of the total while still being large enough to produce very small confidence intervals. It's very likely that at least 1% of the accounts are unattacked at most services, but only at very large services does this constitute enough data to allow us to estimate c and $P(x|\overline{\text{mal}})$ accurately.

The features must be dense. Even, if we identify a subset that is free of attack traffic in (9) to get an accurate estimate of $P(x|\overline{\text{mal}})$ we require that each category of the categorical feature x be densely populated. If our subsets each contain one million benign requests and the categorical feature x has 10 distinct categories then each value of $P(x|\overline{\text{mal}})$ in (9) will be estimated from 100,000 benign samples on average. This

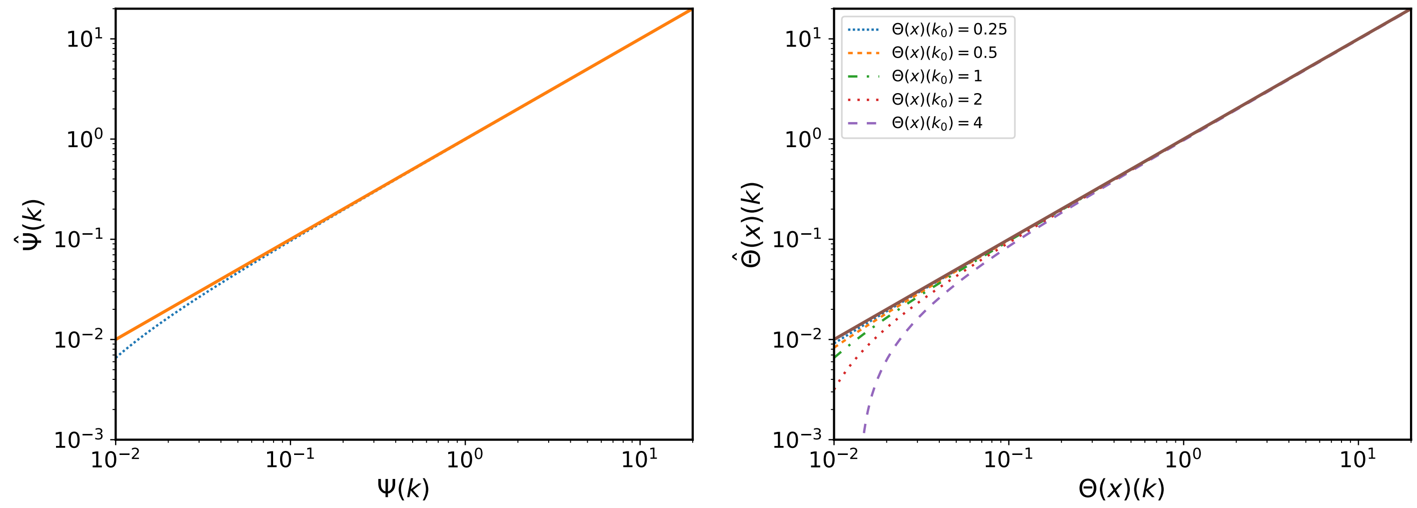
[1] Chatterjee et al. [47] estimated password typos at 4.5% of legitimate attempts, so this allows for additional effects such as username typos and mis-remembered passwords.



(a)

(b)

Fig. 1: Sensitivity analysis showing the effect of gross inaccuracies in $\hat{p}$ on our estimate of the test statistic $\Theta(x)(k) \cdot \Psi(k)$. The fail rate for benign requests is $p = 0.07$ but we estimate it as $\hat{p} = 0.0826$; i.e., we overestimate by almost 20%. Solid line is the actual value, dashed lines are estimates. (a) Effect on our estimate $\hat{\Psi}(k)$ versus the actual value $\Psi(k) = P(\text{mal})(k)/P(\overline{\text{mal}})(k)$. Note that in subsets where the ratio of bad to good traffic is high (e.g., $\Psi(k) > 0.10$) the estimate is very accurate. (b) Effect on our estimate $\hat{\Theta}(x)(k)$ versus the actual value $\Theta(x)(k) = P(x|\text{mal})(k)/P(x|\overline{\text{mal}})$. Different curves show different values of the ratio of x in attack to benign traffic $\Theta(x)(k_0)$ (in the subset used to estimate $\hat{c}$). Note that $\Theta(x)(k_0) = 1$ is the most representative value. Note that when x is relatively common in the attack traffic (e.g., $\Theta(x)(k) > 0.10$) the estimate is very accurate despite the gross error in estimating p.



(a)

(b)

Fig. 2: Sensitivity analysis when p is estimated fairly accurately. The fail rate for benign requests is $p = 0.07$ but we estimate it as $\hat{p} = 0.0732$. Solid line is the actual value, dashed lines are estimates. (a) Effect on our estimate $\hat{\Psi}(k)$ versus the actual value $\Psi(k) = P(\text{mal})(k)/P(\overline{\text{mal}})(k)$. (b) Effect on our estimate $\hat{\Theta}(x)(k)$ versus the actual value $\Theta(x)(k) = P(x|\text{mal})(k)/P(x|\overline{\text{mal}})$. Note that the estimates are extremely accurate, even though we've over-estimated p by about 5% (which is twice the confidence interval that (4) yields with 100k benign samples).

again favors services with large user bases: the more benign data in the subset we use to estimate $P(x|\overline{\text{mal}})$ the better our results will be.

This technique provides accurate estimates of the odds ratio when the assumptions are met. However, as with any statistical technique, it is important that we use features that distinguish well between attack and benign traffic. An accurate estimate of the odds is helpful only if those odds are high enough or low enough to allow us to discriminate; if $\Theta(x) \approx 1 \forall x$ (i.e., the distributions of x in the attack and benign traffic are very similar) then it doesn't matter how accurate our estimates are. An example of a weak feature might be hour-of-day. An attacker might send guesses at a uniform rate throughout the day, while legitimate traffic typically exhibits a diurnal patter. Thus, requests at 3am might have a higher fraction of attack traffic than those at 3pm, but only by a modest factor. An example of a strong feature might be whether a failed attempt involved a Top-100 password. As shown in Section IV-F this gives a very strong indication of malice: the distribution of x is almost certainly very different between attack and benign traffic, and thus $\Theta(x)$ can be expected to be very high.

Note that from Figures 1 and 2 we always under-estimate rather than over-estimate the odds of a request being malicious. Thus our estimates of malice are always conservative and biased toward benign rather than malicious interpretation of observations. This fact is also easily shown directly by examining the formulae for $\hat{\Psi}$ and $\hat{\Theta}(x)(k)$.

B. Mobile clients and touchscreen keyboards

A key assumption is that the benign failure rate p is the same across whatever subsets we use to divide the data. The simple model of a user typing both username and password on a conventional keyboard might have accounted for the majority of traffic in 2008, but the enormous growth in mobile clients has altered this picture. In 2018 many sites probably see a majority of their traffic come from dedicated apps on phones or tablet devices. Chatterjee et al. measure a significant difference between fail rates on regular as opposed to touchscreen keyboards [47]. This confirms the importance of separating these two types of traffic.

The simplest solution is to treat mobile and non-mobile requests separately. It is generally easy to identify mobile devices, since web-sites do this already to serve content that is more appropriate for a smaller screen size. Thus, we can treat mobile and non-mobile as different problems and estimate c differently for each.

VII. Conclusion

We address the problem of protecting an authentication server from online guessing attacks. We give a simple robust procedure to estimate the ratio of bad-to-good traffic and show how this can be used to calculate the likelihood that any particular observation is indicative of malice. We perform a sensitivity analysis that shows our estimates are very robust to likely sources of bias. Our approach has the advantage over three strikes type lockout and variants that it does not assume a scarcity of, e.g, IP addresses for the attacker and we avoid base rate neglect. It has the advantages over machine learning schemes that it requires no labels, and does not assume stationarity of attack traffic.

Acknowledgements: The authors gratefully acknowledge helpful comments from the anonymous reviewers.

References

[1] R. Morris and K. Thompson, "Password Security: A Case History," *Comm. ACM*, 1979.

[2] N. Provos and D. Mazieres, "A future-adaptable password scheme." in *USENIX Annual Technical Conference, FREENIX Track*, 1999, pp. 81–91.

[3] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: new generation of memory-hard functions for password hashing and other applications," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 292–302.

[4] D. Boneh, H. Corrigan-Gibbs, and S. Schechter, "Balloon hashing: A memory-hard function providing provable protection against sequential attacks," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2016, pp. 220–248.

[5] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 145–160.

[6] G. D. Crescenzo, R. J. Lipton, and S. Walfish, "Perfectly secure password protocols in the bounded retrieval model," in *Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA,*, March 2006, pp. 225–244.

[7] D. Florêncio, C. Herley and P.C. van Oorschot, "An Administrator's Guide to Internet Password Research," *Usenix LISA 2014*.

[8] D. Florêncio and C. Herley, "A Large-Scale Study of Web Password Habits," *WWW 2007, Banff.*

[9] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *ACM CCS 2013*, pp. 173–186.

[10] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *Proc. ACM CCS 2010*, pp. 162–175.

[11] J. Bonneau, "The science of guessing: analyzing an anonymized corpus of 70 million passwords," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 538–552.

[12] B. Ur, J. Bees, S. M. Segreti, L. Bauer, N. Christin, and L. F. Cranor, "Do users' perceptions of password security match reality?" in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 3748–3760.

[13] C. Herley, "So Long, And No Thanks for the Externalities: The Rational Rejection of Security Advice by Users," *Proc. NSPW 2009, Oxford.*

[14] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 2016, pp. 1242–1254.

[15] A. Adams and M. A. Sasse, "Users are not the Enemy," *Comm. ACM*, 1999.

[16] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 523–537.

[17] W. E. Burr, D. F. Dodson W. T. Polk, "Electronic Authentication Guideline," in *NIST Special Publication 800-63*, 2006, http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.

[18] Centre for the Protection of National Infrastructure, "Passwords: Simplifying your approacht," https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/458857/Password_guidance_-_simplifying_your_approach.pdf.

[19] Open Web Application Security Project, "http://www.owasp.org."

[20] Alex Weinert, "How we protect #AzureAD and Microsoft Account from lists of leaked usernames and passwords," https://cloudblogs.microsoft.com/enterprisemobility/2016/05/10/how-we-protect-azuread-and-microsoft-account-from-leaked-usernames-and-passwords/.

[21] Akamai, “State of the internet / security Q4 2017 Report,” https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2017-state-of-the-internet-security-report.pdf.

[22] Mark Maunder, “6 Million Password Attacks in 16 Hours and How to Block Them,” 2016, https://www.wordfence.com/blog/2016/02/wordpress-password-security/.

[23] D. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto, “Who are you? a statistical approach to measuring user authenticity.” 2016.

[24] Nat’l Inst. Standards and Technology, “Digital Identity Guidelines,” in *NIST Special Publication 800-63B*, 2015, https://pages.nist.gov/800-63-3/sp800-63b.html.

[25] Bonneau, J. and Herley, C. and van Oorschot, P.C. and Stajano, F., “The quest to replace passwords: A framework for comparative evaluation of web authentication schemes,” in *Proc. IEEE Symp. on Security and Privacy*, 2012.

[26] S. Axelsson, “The base-rate fallacy and the difficulty of intrusion detection,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 3, pp. 186–205, 2000.

[27] S. Brostoff and M. A. Sasse, ““Ten Strikes and You’re Out”: Increasing the Number of Login Attempts Can Improve Password Usability,” *CHI Workshop*, 2003.

[28] “Department of Defense Password Management Guideline,” U.S. Dept. of Defense, Computer Security Center, Tech. Rep. CSC-STD-002-85, 1985.

[29] D. Florêncio, C. Herley, and B. Coskun, “Do Strong Web Passwords Accomplish Anything?” *Proc. Usenix Hot Topics in Security*, 2007.

[30] B. Pinkas and T. Sander, “Securing passwords against dictionary attacks,” in *CCS ’02: Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 161–170.

[31] P.C. van Oorschot, S. Stubblebine, “On Countering Online Dictionary Attacks with Login Histories and Humans-in-the-Loop,” *ACM TISSEC vol.9 issue 3*, 2006.

[32] M. Albrecht, K. Paterson, and G. Watson, “Plaintext recovery attacks against SSH,” in *2009 IEEE Symp. Security and Privacy*, pp. 16–26.

[33] J. Bonneau and S. Preibusch, “The Password Thicket: technical and Market Failures in Human Authentication on the Web,” *WEIS*, 2010.

[34] B. Lu, X. Zhang, Z. Ling, Y. Zhang, and Z. Lin, “A measurement study of authentication rate-limiting mechanisms of modern websites,” in *Proceedings of the 34th Annual Computer Security Applications Conference*. ACM, 2018, pp. 89–100.

[35] H. L. van Trees, *Detection, Estimation and Modulation Theory: Part I*. Wiley, 1968.

[36] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, “Passwords and the evolution of imperfect authentication,” *Communications of the ACM*, vol. 58, no. 7, pp. 78–87, 2015.

[37] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 305–316.

[38] C. Chio and D. Freeman, *Machine Learning and Security*. O’Reilly Media, 2018.

[39] M. G. Kelly, D. J. Hand, and N. M. Adams, “The impact of changing populations on classifier performance,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 367–371.

[40] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[41] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, “Learned lessons in credit card fraud detection from a practitioner perspective,” *Expert systems with applications*, vol. 41, no. 10, pp. 4915–4928, 2014.

[42] Lenny Evans and Karthik Ramasamy, “Exploring New Machine Learning Models for Account Security,” https://medium.com/uber-security-privacy/uber-machine-learning-account-security-3aaadef11e45.

[43] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, NY, USA:, 2001, vol. 1, no. 10.

[44] D. H. Wolpert, “The lack of a priori distinctions between learning algorithms,” *Neural computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[45] P. Domingos, “A few useful things to know about machine learning,” *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[46] W. H. Press, *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[47] R. Chatterjee, A. Athayle, D. Akhawe, A. Juels, and T. Ristenpart, “password typos and how to correct them securely,” in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 799–818.