Leveraging the Crowds to Disrupt Phishing

Jason Gustafson and Jun Li*
{jtg, lijun}@cs.uoregon.edu
Network & Security Research Laboratory
University of Oregon

Abstract—**Preventative anti-phishing approaches are not always effective. As there seem to be always users who ignore warnings, use old anti-phishing software, or rely on obsolete blacklists of phishing sites, phishers continue to find new victims who surrender their credentials to phishing sites. We thus take an aggressive anti-phishing approach, a research direction rarely explored. We study how we may disrupt phishing operations by injecting to phishing sites many fake credentials—also called honey tokens—that are indistinguishable from real credentials, allowing institutions under attack to detect and track phishing activities when honey tokens are used. We address the limitations from our early work, Humboldt 1.0, which automatically submits fake credentials but can fail if phishers take smart countermeasures. Based on a new architecture, Humboldt 2.0, we study how we may leverage human users to submit honey tokens successfully, while being resilient to various malicious countermeasures to our system. We further analyze its effectiveness and evaluate its cost using the Amazon Mechanical Turk service, showing that Humboldt 2.0 can successfully complement Humboldt 1.0.**

Keywords: anti-phishing, phishing disruption, honey token, Humboldt

I. Introduction

In spite of significant research into anti-phishing methodologies, phishers continue to find new victims [1], [2], resulting in 32,581 attacks per month on average, a 19 percent increase over the previous year. [3]. Although current methods mitigate these losses from climbing even further, they are not always effective. For example, the takedown of phishing sites is seldom immediate, blacklists of phishing sites are often obsolete, new software tools are only useful if users install them, and browser warnings are only effective if users heed them. Phishers also become more sophisticated, such as by launching spear-phishing or spreading phishing URLs through social networking sites. But if phishers will collect victims *anyway*, can we still do anything to protect web users?

In this work, we investigate an anti-phishing option that has not received enough attention—aggressive phishing disruption. We present a phishing disruption system called **Humboldt** (named after the Pacific squid known to attack night-time fishers). Different from attempting to take down new phishing sites or prevent users from accessing them, Humboldt is designed to inject large amounts of fake credentials into the phishing sites. These fake credentials are indistinguishable from real ones, and each of them is a **honey token** that Humboldt shares with the institution under attack. The institution can monitor site transactions. If a phisher attempts to use a fake credential in a transaction with the institution, the institution (or law enforcement agencies) can then identify the transaction and even further track the transaction. The more fake, indistinguishable honey tokens Humboldt can inject, the more likely the phisher will use one of them. (Humboldt can be particularly deadly when attacked institutions cooperate to create "honeypot" accounts for submitted honey tokens. This would allow an institution to monitor the account closely to see how the phisher uses it, and could even lead to prosecution.)

A critical issue here is that phishers will make adaptations if they suspect their data is being poisoned. For example, if submissions are done from a small number of IP addresses, it would be trivial for the phisher to monitor submission rates and filter any IPs with high submission rates. In our previous work, we proposed a prototype of Humboldt, i.e., **Humboldt 1.0**, which leveraged a distributed network of clients to submit honey tokens [4]. Participating users simply installed "thin client" software running in the background, and periodically coordinated its actions with a central server.

But while Humboldt 1.0 makes fake submissions indistinguishable from those of genuine submissions, it depended on an *automated* submission procedure. Once a phisher realizes this, she could try to make it impossible for the fake credentials to be submitted in the first place. While leaving the appearance of her phishing site unaffected, the phisher can increase the complexity of the phishing site's underlying structure to foil the automatic profiling of the site. Since modern web page design provides considerable flexibility for doing this (e.g. Javascript, CSS, Flash, HTML5 Canvas, etc.), it is of little trouble for the phisher to do this. In fact, the phisher could just use a CAPTCHA [5] to thwart any automatic submission mechanism.

However, the phisher has a fundamental weakness. No matter what she can do to obfuscate the structure of the phishing web page or change the submission dynamics, the page must always remain usable by *people* and must always accept their submissions. If not, then there is no point in phishing! We can therefore systematically defeat all of the above countermeasures by leveraging actual people to do the submissions.

In this paper, we evaluate the practicality of this idea. We enhance Humboldt 1.0 to build a honey token submission network that enables actual people to inject fake data to phishers. We call this new version **Humboldt 2.0** (in the following we use Humboldt 2.0 and Humboldt interchangeably). We particularly consider the implications of relying on human submitters and

* Corresponding author.
This material is based upon work partially supported by the National Science Foundation under Grant No. CNS-0644434. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

how Humboldt 2.0 can be resilient to various countermeasures of phishers. In order to determine whether it is feasible, we also study its effectiveness and its associated costs based on experiments with real human submitters.

The rest of this paper is organized as follows. We first present the architecture of Humboldt 2.0 in Section II, followed by Section III that discusses its resilience to adversarial countermeasures. We then analyze the effectiveness of Humboldt 2.0 in Section IV, and evaluate its cost of relying on human submitters with experiments using the Amazon Mechanical Turk service [6] in Section V. Finally, we discuss the related work in Section VI, and conclude the paper in Section VII.

II. Humboldt 2.0 Architecture

A. Overview

Figure 1 shows the logical design of Humboldt 2.0. It has four primary components:

- a *central server* which coordinates the assignment and submission of honey tokens;
- a *marketplace* which distributes honey token **submission tasks** to people;
- a set of servers called *exit nodes* which serve as the last hop in each submission; and finally
- a set of external sources called *phishing feeds* which discover new phishing sites.

In this design, Humboldt offers itself as a service to institutions under a phishing attack. Briefly, such an institution provides Humboldt with detailed submission requirements (e.g. how many honey tokens should be submitted each day), and some tangible incentive for users to submit honey tokens. The Humboldt central server then uses the incentive to create submission tasks on the marketplace, targeting any sites posing as the institution which are discovered through the phishing feeds. It then facilitates submission of the honey tokens; once a user submits a honey token, the token will first reach the central server, which then routes the token to an appropriate exit node, which then forwards it to the phishing site.

B. Description

We now provide further details on the purpose and operation of each component of Humboldt 2.0.

1) Central server: The main component in this design is the central Humboldt server, which is responsible for creating submission tasks, making them available to would-be submitters, and coordinating the actual submission through the exit nodes (described below). The core of this server is a relational database known as *HumboldtDB*. It is responsible for storing information on attacked institutions, generated honey tokens for submission, submission statistics to various phishing sites (including which exit nodes are used), and information about worker reliability. Logically, there are two distinct roles for the Humboldt central server. The first is submission task creation, i.e., listening to the phishing feeds, generating honey tokens, and posting them to the market. The second is submission handling; the server is responsible for routing honey token submissions to appropriate exit nodes (discussed below), and monitoring the submission traffic for accounting.

2) Honey token marketplace: The honey token marketplace is responsible for mapping submission tasks to human users. Humboldt tracks whether and by whom every submission task is completed. As mentioned above, there must be incentive for users to submit Humboldt's honey tokens. This could be coupons from the attacked institution, something tangible such as music or books, actual money, or online benefits such as virtual money for use in online games or "reputation points."

3) Exit nodes: In order to evenly distribute the submission tasks, we attempt to leverage a pool of exit nodes to serve as the final hop before reaching the phishing site. The exit node's only job is to proxy the submission to the phishing site, so the machine requirements are minimal and the proxy service does not require human interaction. Exit nodes allow us to spread submissions from individual users across multiple IP addresses. This means we can take advantage of nonuniform distributions of participation among submitters.

4) Phishing Feeds: Finally, we require a source to point Humboldt to new phishing sites. In fact, this could be either an internal (i.e. part of the central server) or external component. In Figure 1, we show it as external since there are already excellent services to identify new phishing sites (e.g. Phishtank [7]). Alternatively, specific phishing URLs could be provided by the targeted institution when they complete the service agreement.

C. Advantages

The first advantage of the Humboldt 2.0 architecture is its reasonable assurance of submission work accomplished. The fact that submissions are routed through the Humboldt central server and onto exit nodes enables Humboldt to monitor the submission of honey tokens. Users cannot simply claim that they have submitted a honey token unless they do have done so. However, there is a limit to what we can do here. Since we do not control phishing sites, we cannot verify the credentials toward a phishing site actually make it to the site. What we seek is "reasonable assurance" of work done by the users.

Humboldt 2.0 is also advantageous in its adoption of exit nodes. Exit nodes are cheap and require no user involvement. Hence it can have a huge number of exit nodes in proportion to the number of actual submitters. In effect, the exit nodes magnify the power of the submitters, thereby solving the problem of distributing submissions. Furthermore, although we have implied each exit node corresponds to a single machine, the only thing that matters is that there are a large source of IP addresses to work with. A single machine which can intercept traffic for unused IPs on a subnet could serve as a "super" exit node. Or even better, we could use a cloud-based infrastructure with virtual machines across many geographic locations as the exit nodes; based on techniques in [8], phishers can hardly detect and block the IP addresses of such virtual machines.

Furthermore, since all honey token submissions are forwarded through the Humboldt server, the server can attempt to learn the submission characteristics. The central server can observe several "correct" submissions, match their corresponding

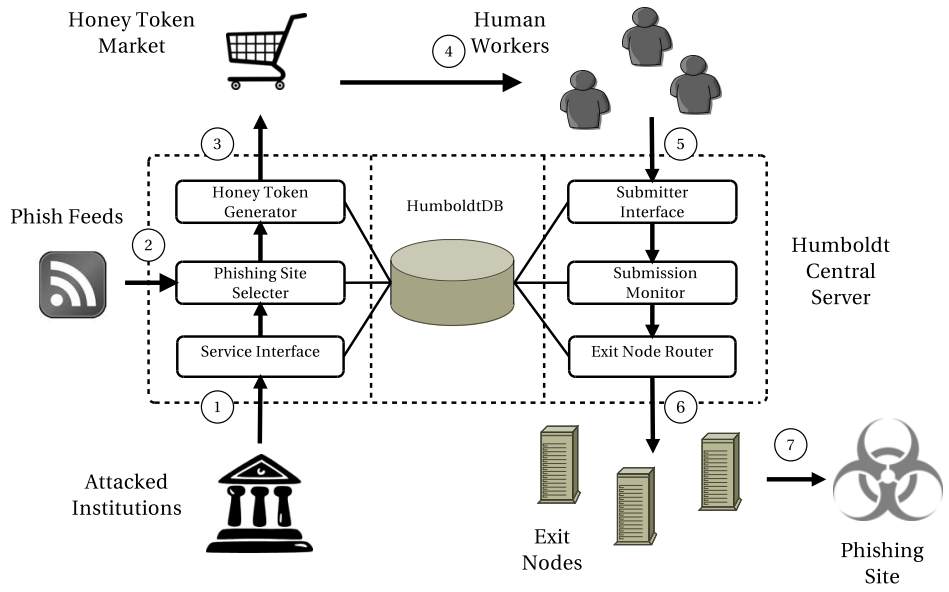


Fig. 1: Humboldt 2.0 Architecture and Work flow: In step 1), an institution provides submission requirements and rewards for people who submit honey tokens; in step 2), Humboldt listens to various phishing feeds to identify phishing sites attacking that institution; after such a site is found, in step 3), honey tokens are generated and the submission task is posted to the market; in steps 4) and 5), the user accepts the task and begins submission of the honey token through the central server; finally, in steps 6) and 7), Humboldt routes the submission to an appropriate exit node, which then forwards it to the phishing site.

HTTP fields, and then directly validate subsequent submissions. For example, after several submissions reveal the username and password form fields, the central server can verify further submissions whether they have included the username and password of the assigned honey token in the right fields.

III. Adversarial Resilience

Humboldt 2.0 is further designed to be resilient to adversarial threats. In this section, we discuss the specific threats and show how Humboldt handles them. We first introduce a threat model which describes the capabilities we assume that phishers have individually and collectively. We then systematically discuss how to address countermeasures of phishers.

A. Threat Model

Our threat model defines what actions phishers can take and what information they can use to defend their sites or mitigate the threat of repercussions from attempting to use honey tokens. First and foremost, we assume that phishers know about the existence of Humboldt, including how it works and information about its control structure (e.g. the IP address of the central server), but we do not assume phishers can subvert and control Humboldt itself. We assume that phishers are capable of making changes to the sites they control at any time. The only real limitation we place on the phisher is that they must maintain the site's usability. This is reasonable since a phishing site has no point if people cannot use it. We also assume that certain human workers or exit nodes can be malicious, and can collude with phishers to counter Humboldt's operation.

Moreover, we assume that phishers can collect statistics about their visitors. They can easily determine, for example, the IP address that each submission came from, and what the characteristics of the submission are (e.g. the time between page load and submission). They can also store cookies or use other client-side method to try to track users who revisit the site.

Finally, we assume phishers can deploy potentially many phishing sites and they can also collaborate with other phishers in order to gain additional information about submitters. Or, the phisher may simply collude with certain submitters.

B. Adversarial Tactics

Although we do not make any assumptions about the phisher's operating costs, it is clear that they will prefer methods which are cheaper. Additionally, they will certainly prefer methods which can provide guarantees of effectiveness. While a comprehensive listing of tactics is impossible, we can broadly characterize phisher tactics as either *active* or *passive*.

1) Active Tactics: Active tactics interfere with the ability to submit honey tokens. The most obvious such attack is a denial of service (DoS). Humboldt depends on a central server which assigns submission tasks to a set of nodes which were then responsible for the actual submissions. If this server is taken offline, no submissions can take place and the phisher is protected. Denial of service is covered extensively in the literature, and can be dealt with, for example, by distributing the task of assigning submissions to multiple servers.

The phisher could also target the honey token marketplace. In a simpler form, the phisher may use bots to accept and conduct submission tasks. We can prevent this attack by requiring every submitter to first solve a CAPTCHA to ensure it is a human user. The more complex form is when the phisher attempts to enlist malicious users for submission tasks. Doing so will not only cut into the phisher's operating costs, but will also be hardly effective as explained below:

After accepting a submission task posted by Humboldt, a malicious user could choose to (1) not act on it; (2) submit the assigned honey token incorrectly; or (3) submit the token correctly but also collude with the phisher, including using an out-of-band channel to notify the phisher of the token. However, given a reasonably sophisticated marketplace, it is difficult for such attacks to be successful. First, we can set an expiration time for every submission task, and assign expired tasks to new users. Second, we can refuse submissions from users who have failed previously (recall Humboldt's central server monitors every submission). Third, all correctly submitted credentials—including those submitted by malicious users or an accomplice of the phisher, once used in transactions with the institution under attack, can reveal the ongoing phishing activities. The phisher can discard those honey tokens that she learned via collusion, but not all tokens.

In fact, the marketplace itself could even make getting new accounts difficult (e.g. by requiring multiple forms of identification). For any market of reasonable size (such as Amazon Mechanical Turk), the population of malicious users will be small, and tokens correctly submitted by malicious users will only be a small portion of those submitted by benign users. The only advantage of malicious users are rewards that they can receive for successfully submitted tokens. But given that the number of malicious users is small, Humboldt can treat such rewards as an acceptable overhead, and further reduce it by being frugal in rewarding successful submissions.

Finally, the phisher may also target the exit nodes. For example, the phisher can have its own machines become exit nodes or compromise current exit nodes. These malicious exit nodes can then notify the phisher of the honey tokens they have seen. Or, they can fake responses from the phishing site, perhaps even collaborating with malicious users, while actually not forwarding honey tokens to the site. Here, Humboldt cannot know for certain if an exit node actually forwards the submissions to the phishing site since Humboldt does not control that final hop.

For this threat, recall Humboldt does not need to guarantee all honey tokens to be successfully submitted to the phishing site. Humboldt is successful as long as enough honey tokens successfully make to the phishing site. Humboldt also controls which exit node to use for every submission, and can simply randomly pick one every time. Therefore, unless the phisher controls most exit nodes or Humboldt is stuck with using malicious exit nodes, this tactic cannot stop Humboldt, either.

The above malicious tactics will be even more difficult if Humboldt can identify trusted human workers and exit nodes, and only rely on them. Trust management, however, is an active research topic by itself, and thus out of the scope of this paper.

2) Passive Tactics: Since honey token are submitted by actual people, attempts to hamper robots are useless. There may, however, still be some ways for a phisher to detect honey tokens without interrupting the Humboldt service. For example, the phisher might attempt to detect them purely from the submitted data. If the fake submission includes a password which does not meet the length or entropy requirements of the target site, then the submission can be rejected without further thought. The phisher might also try statistical methods, such as comparing the information content of each submission against a set of known genuine credentials. However, these methods are unlikely to be reliable given any reasonably sophisticated method for generating usernames and passwords. Furthermore, there will be some chance of generating false positives, which will cut into the phisher's potential for profit.

For geographically local targets, such as a small community bank, phishers can attempt to filter any submissions coming from outside the region where the bank is operating. One such method is IP geo-location. We can also imagine a CAPTCHA which is designed to be solved only by people from a particular region (e.g. a classic CAPTCHA, but with Chinese characters). This problem could be solved by adding a location constraint to the submission task. We note again that this tactic will likely also result in false positives for the phisher.

Yet another passive approach is to use an indirect method to confirm which credentials are real. For example, many web services provide an e-mail account with the same username that is used to log in. The phisher could simply send e-mails to these addresses and see which ones are bounced back. In some cases, user information may even be publicly available (e.g. a game service which posts user scores). The solution to this threat is simple: Humboldt can ensure the username in every honey token is real (i.e., matches the username of a real user of the targeted institution), but the password and other credential fields are not. Since those real users might not like being used for this purpose, a better option might be associating *honey accounts* with each honey token. This is, in fact, one of the great opportunities which Humboldt offers, since it could allow law enforcement agencies to observe the tactics used by phishers to turn the account to money.

Perhaps the most troublesome passive method (from Humboldt's perspective) is for the phisher to track the IP addresses of each submission and filter all submissions from IP addresses which have high submission rates. We refer to this as the **source heuristic**. The simplest idea is to ensure that every phishing site receives honey tokens from different IP addresses. While reasonable, this idea overlooks an important point: our threat model allows the phisher to use several sites and collaborate with other phishers, while we cannot know when the same phisher may see submissions to separate sites. Since this is a fundamental limitation of Humboldt, we next analyze its effect in detail.

C. Source Heuristic

Suppose that a phisher can access the submission statistics to s sites. We are interested in how effective the source heuristic can be in filtering submissions with the extra knowledge obtained from these sites. Below, we first analyze the worst case for Humboldt when the phisher greedily filters all instances of duplicate IP submissions, then discuss if the phisher can be always greedy, and how Humboldt can mitigate the threat from the source heuristic.

Suppose that fake submissions can be done from a set of N distinct IP addresses, and that $v \leq N$ honey token submissions are done to each of the s sites, so that no two submissions to the same site come from the same IP address.

Suppose first that $s = 2$. Let $\mathcal{I}_0$ and $\mathcal{I}_1$ represent the two sets of IP addresses collected from the two sites respectively. The source heuristic allows the phisher to discard submissions from IP addresses in the intersection of these sets. Let the random variable D_2 represent the size of this intersection (i.e. $D_2 = |\mathcal{I}_0 \cap \mathcal{I}_1|$). Assuming that the submission IP addresses are selected randomly for each site (without replacement), the expected value of D_2 is given by

$$E(D_2) = \sum_{i=1}^{v} i \frac{\binom{N}{i}\binom{N-i}{v-i}\binom{N-v}{v-i}}{\binom{N}{v}^2}. \quad (1)$$

With 100 distinct nodes at our disposal and 50 credentials submitted to the two sites, the source heuristic allows the phisher to discard 50% of the submissions (on average). Looked at from another direction, this means that we would have to submit 50 credentials in order to have 25 of them succeed. Clearly this presents a problem.

For $s > 2$, the situation is more dire. Since an exact formula becomes fairly complicated, we consider an approximation which is based on the $s = 2$ case. Let the sets of IP addresses be given by $\{\mathcal{I}_i\}_{i=1}^{s}$, and let $\mathcal{I}_{ij} = \mathcal{I}_i \cap \mathcal{I}_j$ for each of the $\binom{s}{2}$ distinct pairs of sets. The number of IP addresses which are eliminated by the source heuristic is given exactly by $|\bigcup_{i \neq j} \mathcal{I}_{ij}|$. Now, there will possibly be some overlap between each pair of the $\mathcal{I}_{ij}$'s, but we can get an upper bound by assuming they are disjoint. In other words,

$$|\bigcup_{i \neq j} \mathcal{I}_{ij}| \leq \sum_{i \neq j} |\mathcal{I}_{ij}| \quad (2)$$

Let the random variable D_s be given by $D_s = |\bigcup_{i \neq j} \mathcal{I}_{ij}|$. Using the previous formula for $E(D_2)$, the expected intersection size between two sets, we can write the approximation as follows of $E(D_s)$ as

$$E(D_s) \leq \binom{s}{2} \sum_{i=1}^{v} i \frac{\binom{N}{i}\binom{N-i}{v-i}\binom{N-v}{v-i}}{\binom{N}{v}^2}. \quad (3)$$

Figure 2(a) shows the effect of the source heuristic when 50 credentials are submitted to each of the sites controlled by the phisher. In this plot, the x-axis is the number of distinct nodes; the y-axis is the average number of credentials to each phishing site which are not filtered by the phisher. If the phisher controls 25 sites, we require roughly 3000 distinct nodes in order to have 40 credentials pass through (on average). We could improve on the approximation by subtracting the expected intersection between sets of size 3, 4, and so on. But in fact, when the value of v is small in proportion to N, this approximation is already quite close to the true value.

The worst case for Humboldt is when the phisher greedily filters all instances of duplicate IP submissions. However, the phisher may be reluctant to take this action. In particular, given the prevalence of Network Address Translation (NAT) in modern networks, as well as the wide usage of the Dynamic Host Configuration Protocol (DHCP), two users submitting from the same IP address is not uncommon. If the phisher greedily throws out all of these submissions, then there could be a large number of false positives. On the other hand, if the phisher throws out only submissions from IP addresses which have been the source of some number of distinct submissions, then they might reduce the false positive rate, but at the cost of accepting more honey tokens. Figure 2(b) shows a more conservative approach where the phisher filters IP addresses only if they are the source of at least 3 submissions.

Although the source heuristic presents a significant problem, we can take several actions to mitigate the threat. First, if we have a large and *dynamic* pool of exit nodes to select from, this tactic becomes useless to the phisher. This implies that the best exit nodes might be mobile devices which change IP addresses frequently. Second, when we know that an exit node conceals a large number of NAT users, we can afford to have its IP address exposed to the phisher through this heuristic. If the phisher's only response is to filter submissions from this IP address, then we have effectively protected all of the NAT users.

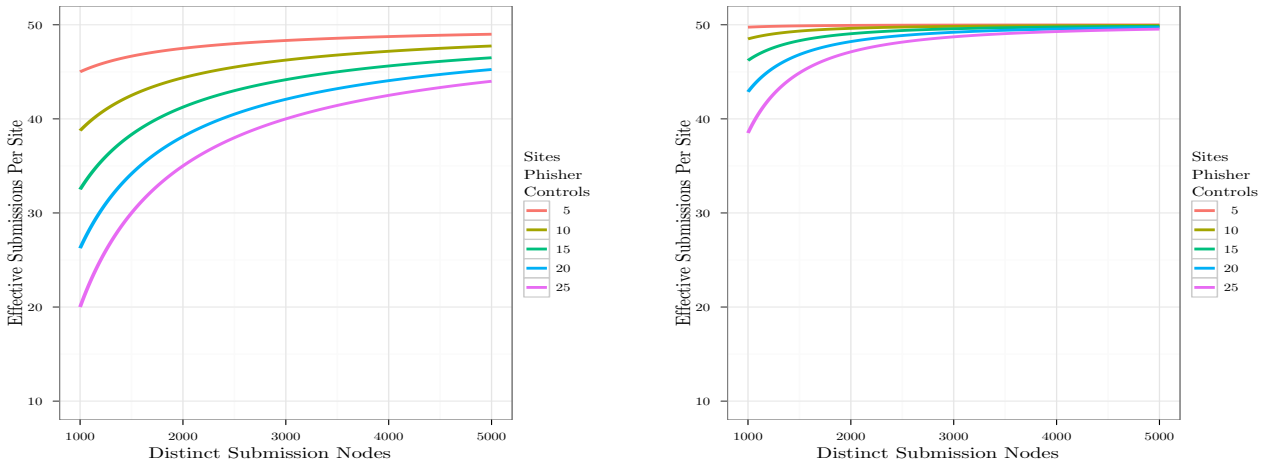
IV. Effectiveness of Humboldt 2.0

From a high level, the goal of Humboldt is to submit "enough" honey tokens in an undetectable way to potentially many phishing sites. In this section, we describe the requirements to be effective in this sense, and the metrics which we will use to evaluate it. We also attempt to determine what "enough" is. Note that our focus in this section is to analyze how effective Humboldt can be; we will show how Humboldt can enlist human users to disrupt a "phishing" site we controlled in the next section.

In fact, given that in a phishing campaign every honey token should be submitted through a different exit node, knowing the number of honey tokens needed will also help gauge how many exit nodes Humboldt needs to employ. Note, however, once there are enough exit nodes to spread submissions from users, Humboldt can leverage either one willing and capable user for all honey token submissions, or more users to share the submission load.

Since we do not control the phisher's website, the only direct way we can measure effectiveness is by cooperating with the attacked institution to monitor the use of honey tokens. However, we can estimate the number of submissions required in order to be effective with a probabilistic model. Suppose a phishing site receives totally n submissions, h of which are from Humboldt (and $n - h$ from genuine victims). Let $P(X)$ represent the probability that the phisher will attempt to use some number $X \leq h$ of the credentials submitted by Humboldt.

Our first goal is simple detection: we only require that the phisher use one of the submitted honey tokens. Suppose that the phisher verifies a total of k submissions out of the total n, then the probability that at least one of these submissions



(a) With Greedy Filtering

(b) With Conservative Filtering

Fig. 2: The effect of the source heuristic on 50 submissions

comes from Humboldt is given by

$$P(X \geq 1) = \sum_{i=1}^{h} \frac{\binom{h}{i}\binom{n-h}{k-i}}{\binom{n}{k}} \qquad = 1 - \frac{\binom{n-h}{k}}{\binom{n}{k}}. \tag{4}$$

As an example, with $n = 100$ and a modest $k = 10$, we must submit just 20 honey tokens in order to have $P(X \geq 1) \geq 0.9$. Assuming the phisher has no method for detection, this suggests that we will be effective even with relatively few submissions.

The second goal we might reasonably strive for is to detect the use of honey tokens before actual victims have been attacked. Such *early detection* may give us an opportunity to protect the victims. Intuitively, for early detection to be effective, compared to using credentials from actual victims, the likelihood that the phisher will use fake credentials submitted by Humboldt must be high. Assume for simplicity that we are able to stop the phisher's actions after we detect l transactions with honey tokens from Humboldt (e.g. if l logins all come from the same IP, the target institution can block that IP). The question is what percentage of victims can be helped?

Let V be a random variable representing the number of victims targeted by the phisher before the required l submissions from Humboldt have been detected. Then $P(V)$ is the probability that there will be exactly V such victims. Our goal is to determine what h should be in order to have the expected number of unprotected victims $E(V)$ as few as possible. If we assume that the submissions from Humboldt are distributed randomly among the total submissions, we can consider this from the perspective of partitioning the set of h Humboldt submissions around the pivotal lth submission. From this perspective, it is clear that the probability that the ith overall credential used by the phisher will also be the lth credential submitted by Humboldt is given by $\binom{i-1}{l-1}\binom{n-i}{h-l}/\binom{n}{h}$. The number of victims which occur before the lth Humboldt submission then is simply given by $i - l$. By then letting $j = i - l$, we can therefore derive an equation for $E(V)$ by summing over each possible value for j.

$$E(V) = \sum_{j=1}^{n-h} j \cdot \frac{\binom{j+l-1}{l-1}\binom{n-j-l}{h-l}}{\binom{n}{h}} \tag{5}$$

If we let $a_m = m\binom{m+l-1}{l-1}$ and $b_m = \binom{m+h-l}{m}$, then this summation can readily be seen to be the $(n-h)$th term in the convolution of the respective generating functions. From this, we can derive the much simpler equation

$$E(V) = \frac{l(n-h)}{h+1}. \tag{6}$$

Our goal is to minimize $E(V)$. Define r as $(n-h)$, i.e., the number of real submissions, Equation (6) can be written as

$$E(V) = \frac{l * r}{h+1}. \tag{7}$$

Though we will not generally know how many real submissions occur (i.e., the value of r), we can estimate it by estimating the size of the phisher's advertising campaign and the response rate. We then can further estimate how many submissions will be needed in order to protect a given percentage of the victims.

V. Evaluating Human Users

In order to prove the viability of Humboldt, we must evaluate the most unpredictable and potentially most costly piece: the human users we depend on for submissions. The facility cost for running Humboldt, such as storage, bandwidth and computation cost, is easily manageable and thus less interesting in this study. We are interested primarily in what it takes to enlist their participation and how reliably they are able to complete each submission task.

We designed and conducted several experiments using the Amazon Mechanical Turk (AMT) marketplace in order to validate the use of people in our architecture. Briefly, AMT centers around units of work known as Human Intelligence Tasks (HITs). HITs are posted by requesters for a given price and workers choose them at their discretion. After the worker

Total HITs	4643
Submitted HITs	3829
Expired HITs	814
Total Worker Cost	181.42
Total Amazon Commission	18.14
Avg. Cost per HIT	0.052
Unique Workers	213
Avg. HITs per Worker	17.82

Fig. 3: AMT Experiment Summary

accepts a HIT and completes the task, the result is sent to the requester who can either approve the work (in which case the worker is payed) or reject it. Requesters are allowed to specify the time allowed for the completion of a HIT.

In our experiments, a HIT corresponds to the submission of a single honey token. Rather than submitting to an actual phishing site, however, our HITs assigned workers to submit to a site we controlled. This allowed us to verify whether a submission was done correctly. Our submission site consisted of nothing more than a standard login form with fields for username and password, and a CAPTCHA so that submissions could only be done by people. Briefly, the HIT description gave a specific honey token for the user to submit. After a worker accepted the task, they input the credentials and solved the CAPTCHA. The site then presented the worker with a unique ID, which was then submitted back through the AMT interface. Since the honey token was visible to the public in this experiment, this last step ensured it was the assigned user who submitted it.

A. Experiment Results

The experiments were conducted over the course of one week and varied over prices per HIT and creation intervals. Figure 3 contains an aggregate summary of all the experiments.

In the first experiment, we were interested in understanding how the HIT price affected response delays from users. Based on a preliminary trial, we determined that we could get participation for as little as 5 cents per HIT. We wrote a program to create 50 HITs every hour for 10 hours. Initially, the price was set at 10 cents per submission. It was lowered one cent every subsequent hour. Figure 4(a) shows the effect of the price on the delay between the time the HIT was created and the time it was submitted. The X's mark the submissions which were done incorrectly (i.e. either the username or password did not match what was assigned). As expected, higher priced HITs are completed faster, though the difference is not as noticeable as might have been expected. Also, there is no clear pattern in the number of incorrect submissions. Indeed, there were just two incorrect submissions for prices lower than 4 cents. From this, there seems to be no compelling reason to pay any more than 1 cent per submission.

In our next experiment, we attempted to observe how worker participation (percentage of HITs completed) scaled with submission requirements (number of HITs to be completed). We set the time before expiration of each HIT at 15 minutes and created an increasing number of HITs over four such intervals. This experiment was fairly costly, so we only did this for prices of 2, 4, and 6 cents. Figure 4(b) shows the results. The y-axis shows the proportion of the HITs which were completed according to whether the submission was done correctly (black) or not (grey). As in the previous plot, there is no clear relation between the price and the number of incorrect submissions. However, offering higher prices clearly results in a larger proportion of the HITs being completed. At 2 cents per submission, only about 25% of the HITs are completed; at 6 cents per submission, almost all of them are. This tells us that we can expect to have to increase the price in order to meet submission requirements.

A reasonable question to ask is whether some workers should be preferred over others. Let the *reliability* of each worker be defined as the proportion of correctly submitted HITs to the total number of HITs they accept. Figure 5 shows how the reliability of each worker varied according to the number of total HITs they accepted, and the average price of the HITs they accepted. While there is a noticeable trend in reliability as the worker accepts more HITs, it is not great enough to give us a lot of confidence. The trend is even smaller when varied against the average price of HITs which users accepted. In fact, these results suggest that we will always have to account for some proportion of faked submissions. The good news is that the proportion is relatively small.

B. Cost Estimates

The cost per submission (i.e the exact price we payed the worker) is in some sense an inadequate metric. It does not tell us anything about whether the submission succeeded. The success of the submission depends on the worker submitting the correct credentials and the phisher not being able to filter it. We therefore define the *effective cost* as the net price payed per *successful* submission.

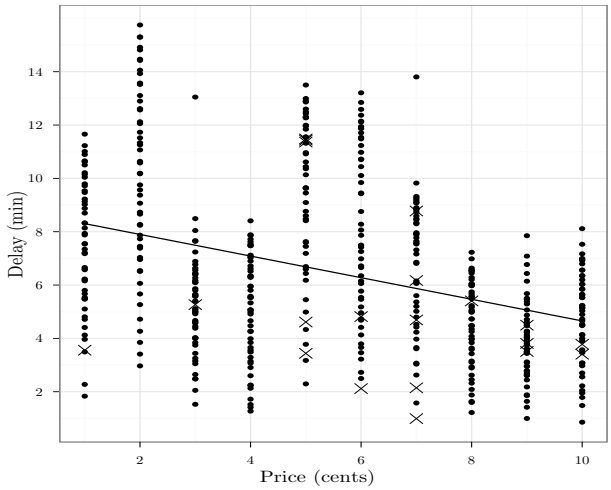
Suppose that the user fails in the submission with probability β. Based on the above experiments, we have found that we can expect $\beta \approx 0.07$. In other words, there is about a 7% chance that each submission will fail. If the phisher had no way to detect the submissions, our effective cost is clearly given by $C/(1-\beta)$, where C is the price payed for each submission.

We also consider the effect of the source heuristic. Its impact is similar to that of β from above: it increases the effective cost of each submission. In Figure 6, we show how the effective cost increases as the number of submissions to a site increases. In Figure 6(a), we show the greedy filtering approach in which the phisher filters all submissions from IP addresses which have submitted at least twice. In Figure 6(b), we show the more conservative approach in which the phisher filters submissions only after an IP address has been observed three times or more. Clearly, when the phisher owns a large number of sites, the effective cost can become quite large.

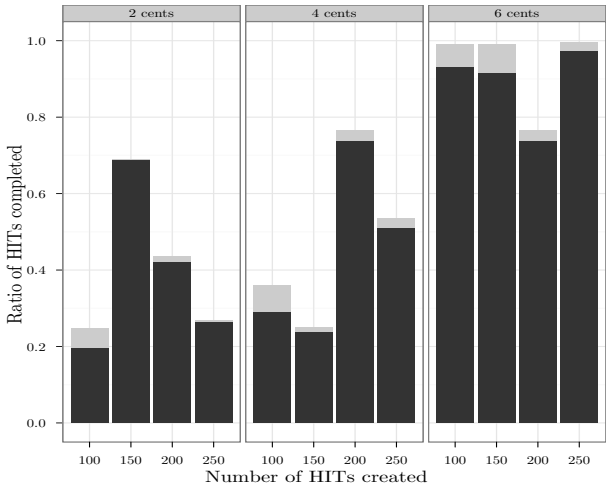
VI. Related Work

Anti-phishing defenses in the literature can be categorized as either preventative or aggressive. We discuss them below:

Preventative approaches aim to disallow users from accessing phishing sites. Solutions are typically integrated in the

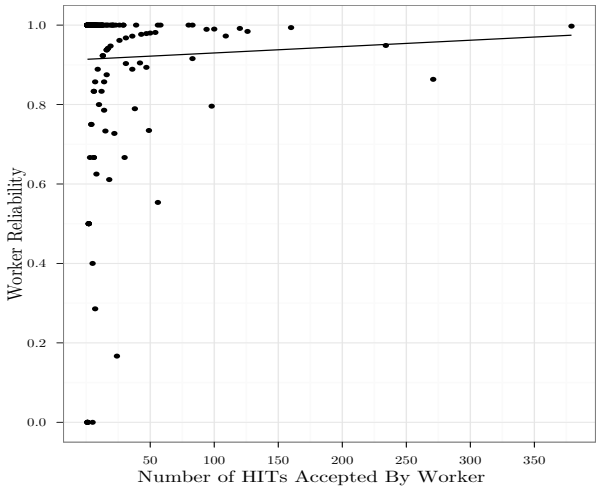


(a) On submission delay



(b) On HIT completion

Fig. 4: The Effect of Price



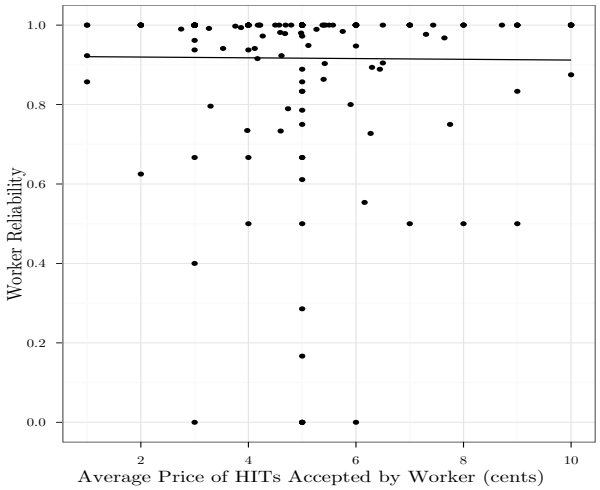
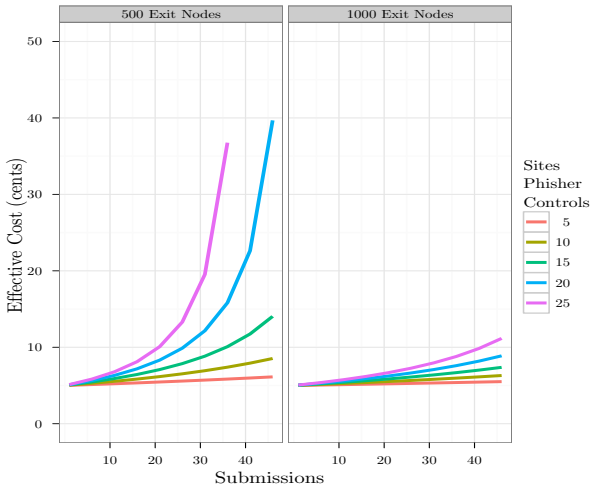
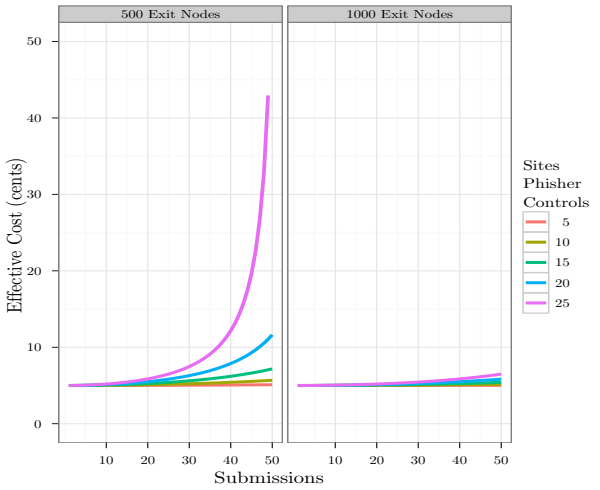


Fig. 5: Worker Reliability



(a) Greedy Filtering



(b) Conservative Filtering

Fig. 6: How Source Heuristic Affects Effective Cost

web browser or email client as plug-ins. The most popular method today uses blacklists or whitelists to distinguish between phishing and legitimate sites [9], [10]. There are also many heuristic-based or machine-learning-based approaches to detecting phishing web sites [11], [12], [13], [14], [15], [16], [17]. The critical weakness of these solutions is that they depend on new or updated software components, thus slow adopters are unprotected.

Aggressive anti-phishing approaches such as Humboldt have been studied in previous literature. Phishing site take-down is one example. If a victim web site finds a phishing site for its own brand, the victim site can issue a take-down notice to the company hosting the phishing site [18]. However, there must necessarily be some delay before a takedown can take effect, during which users are unprotected. There are also ethical dilemmas in take-down [19]. Additionally, fast-fluxing techniques [20] make takedown virtually impossible since the hosting site is continually changing.

Methods for disrupting the ability of the phisher to steal credentials as Humboldt have also been studied. The earliest work describes this as "phish feeding" [21] and presents a preliminary framework for doing submissions. PHONEY [22] and BogusBiter [23] are more developed systems for fake credential submission. PHONEY intercepts the users email traffic and discovers phishing sites by studying the structure and content of the messages, while BogusBiter is activated when a user visits a phishing site. When a phishing site is found, they automatically generate and submit faked credentials to it. But like all systems depending on automatic classification, they are susceptible to false positives and false negatives. Furthermore, phishers can craft email messages specifically to avoid detection by PHONEY. Also, a phisher can easily filter out the faked submissions from BogusBiter by simply watching for batch submission from a particular address.

VII. Conclusion

In spite of significant research into anti-phishing methodologies, phishing remains a huge security threat to Internet users and phishers continue to find new victims. Although continued refinement of current methods helps to mitigate this threat, they are not always effective: the takedown of phishing sites is seldom immediate, blacklists of phishing sites are often obsolete, new tools are only useful if users install them, and browser warnings are only effective if users heed them.

As phishers are unstoppable in obtaining new victims, we proposed to disrupt their operations by injecting to phishing sites honey tokens that are indistinguishable from real credentials, allowing an institution under attack to monitor and track phishing activities. In order to address smart countermeasures that phishers may take to thwart automatic submissions, we focused on how human users may be leveraged for honey token submissions and designed an architecture called Humboldt 2.0.

Not only have we made Humboldt 2.0 resilient against both active and passive countermeasures from phishers, our analysis regarding its effectiveness is also promising. Our experiments using Amazon Mechanical Turk further shows the feasibility of using humans for honey token submission when automatic submission option is not available. Further work may include addressing certain legal issues and more interactions with potential victim institutions and law enforcement agencies when running Humboldt 2.0 in the real world.

References

[1] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74–81, 2012.

[2] Anti-Phishing Working Group. APWG phishing trends reports. [Online]. Available: http://anti-phishing.org/phishReportsArchive.html

[3] RSA FraudAction Research Labs, "Phishing in season: A look at online fraud in 2012," August 2012. [Online]. Available: http://blogs.rsa.com/phishing-in-season-a-look-at-online-fraud-in-2012

[4] P. Knickerbocker, D. Yu, and J. Li, "Humboldt: A distributed phishing disruption system," in *eCrime '09: Proceedings of the Anti-Phishing Working Groups 4th annual eCrime Researchers Summit*, 2009.

[5] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," *Advances in Cryptology–EUROCRYPT 2003*, pp. 646–646, 2003.

[6] Amazon.com Inc., "Amazon mechanical turk." [Online]. Available: https://www.mturk.com

[7] OpenDNS, LLC. PhishTank. [Online]. Available: http://www.phishtank.com/

[8] E. Ferguson, J. Weber, and R. Hasan, "Cloud based content fetching: Using cloud infrastructure to obfuscate phishing scam analysis," in *IEEE Eighth World Congress on Services (SERVICES)*, 2012, pp. 255–261.

[9] Google, Inc., "Google safe browsing for Firefox," http://www.google.com/tools/firefox/safebrowsing/.

[10] Microsoft, Inc., "Microsoft sender id framework," https://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx.

[11] G. Xiang, "Toward a phish free world: A feature-type-aware cascaded learning framework for phish detection," Ph.D. dissertation, Carnegie Mellon University, 2013.

[12] M.-E. Maurer and L. Höfer, "Sophisticated phishers make more spelling mistakes: using URL similarity against phishing," in *Cyberspace Safety and Security*. Springer, 2012, pp. 414–426.

[13] W. Liu, G. Liu, B. Qiu, and X. Quan, "Antiphishing through phishing target discovery," *Internet Computing*, vol. 16, no. 2, pp. 52–61, 2012.

[14] R. Basnet, A. Sung, and Q. Liu, "Feature selection for improved phishing detection," *Advanced Research in Applied Artificial Intelligence*, pp. 252–261, 2012.

[15] W. Loesch and D. Fluker, "Secure web site authentication using web site characteristics, secure user credentials and private browser," January 10 2012, US Patent 8,095,967.

[16] G. Xiang and J. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proceedings of the 18th WWW*, 2009, pp. 571–580.

[17] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell, "Client-side defense against web-based identity theft," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2004.

[18] T. Moore and R. Clayton, "Examining the impact of website take-down on phishing," in *eCrime '07*, 2007, pp. 1–13.

[19] ——, "Ethical dilemmas in take-down research," *Financial Cryptography and Data Security*, pp. 154–168, 2012.

[20] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and detecting fast-flux service networks," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2008.

[21] J. Brozycki, "Phish feeding: An active response to phishing campaigns," trueinsecurity.com, Tech. Rep., 2006. [Online]. Available: http://www.trueinsecurity.com/PhishFeeding.pdf

[22] M. Chandrasekaran, R. Chinchani, and S. Upadhyaya, "PHONEY: Mimicking user response to detect phishing attacks," in *Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks*, 2006, pp. 668–672.

[23] C. Yue and H. Wang, "Anti-phishing in offense and defense," in *Proc. of ACSAC*, 2008, pp. 345–354.