Humboldt: A Distributed Phishing Disruption System

Paul Knickerbocker, Dongting Yu, and Jun Li
{pknicker, dongting, lijun}@cs.uoregon.edu
Department of Computer and Information Science
University of Oregon

***Abstract*—Conventional techniques for combating phishing have focused primarily on detecting phishing web sites and preventing users from revealing their passwords to such sites. This passive form of defense is by its nature incomplete and does nothing to protect users that do reveal their passwords. Combating the phishing threat requires more than simple avoidance—it requires a more active approach to disrupting even successful phishing operations.**

Humboldt is a distributed system that submits poisonous fake data to phishing web sites that is indistinguishable from the input of actual phishing victims. The poisonous data collected by a phisher produces detectable behaviors when the phisher attempts to use it and provides a mechanism for tracking activities associated with identity theft. We evaluate Humboldt to show that it is effective in disrupting phishing operations with a reasonably low overhead.

Keywords: phishing, active phishing defense, Humboldt

I. Introduction

Phishing is a severe security threat on today's Internet. By deceiving users to provide personal information to a fake web page that is visually almost indistinguishable from its legitimate counterpart, phishers have caused widespread identity theft and immense finance loss.

Anti-phishing work has been focusing mostly on detecting phishing sites and preventing users from accessing them. However, while detection and prevention of phishing is an essential part of combating phishing, users may still unknowingly submit their personal data to phishing sites and thus fall prey to phishing attacks. To address this problem, researchers have proposed an aggressive solution, BogusBiter [1] to try to pollute the data that phishers steal. Whenever a web client visits a phishing site and is asked to provide a credential, BogusBiter will simultaneously send a constant number of bogus credentials similar to the original credential, causing the phisher to be confused with which credentials it has collected are real. Unfortunately, although every individual bogus credential is indiscernible from a real one, the phisher can simply discard all those credentials received in a batch from the same machine. Moreover, since BogusBiter relies on those users who have installed BogusBiter to visit phishing sites manually, only a limited number of bogus credentials can be submitted to pollute the phisher's database.

In this paper, we introduce a new, aggressive approach to phishing called Humboldt. Like BogusBiter, it also poisons the data that phishers obtain *en masse* in order to actively disrupt phishing activity. But Humboldt takes a different approach to injecting fraudulent submissions into the phishing site's collected data. It relies on Humboldt clients distributed over the Internet to submit poisonous data to every phishing site it targets. In particular, as an aggressive approach, Humboldt has the following properties:

- Poisonous data from Humboldt is indistinguishable from the data submitted by real phishing victims, not only in terms of the data itself, but also in the way the data is submitted;
- The submission of poisonous data is coordinated among Humboldt clients in order to prevent detectable behavior which would make post-processing by phishers easier and also to avoid the risk of launching DDoS attacks against the innocent machine that hosts the phishing site; and
- Data submission from Humboldt is also automated, without requiring manual intervention from users.

With enough clients, Humboldt can inject a significant amount of fake data into the phisher's database, either disrupting the phishing campaign or exposing the phishers when they try to use these fake credentials—which are generated and recorded by Humboldt—on the real web sites they were pretending to be. Humboldt can also cause data entries stolen from real victims to be

978-1-4244-4626-1/09/$25.00 ©2009 IEEE

interspersed among fake entries, protecting those phished users now that their entries are harder to pick.

We describe how Humboldt achieves these properties in this paper in detail. After presenting the basic design requirements and how to meet them in Section II, in Section III we describe in detail Humboldt's architecture that is composed of a central server with three major components and multiple clients that are responsible for injecting poisonous data to phishing sites. In Section IV we discuss how Humboldt can be robust against smart phishers that try to circumvent Humboldt. We then evaluate Humboldt to show that it is effective in disrupting phishing operations with a reasonably low overhead in Section V. Finally, we present related work in Section VI and conclude the paper in Section VII.

II. Humboldt Design

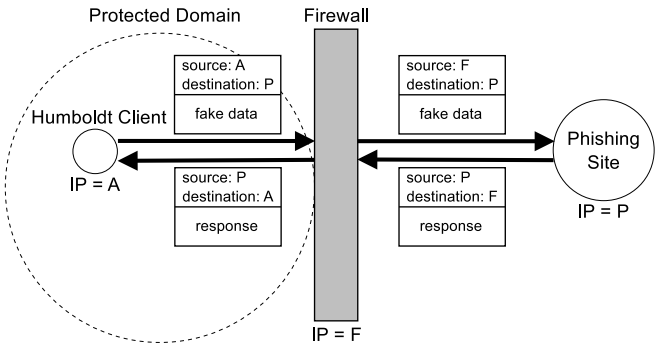
Humboldt must meet a few key requirements in order to be effective. While taking an offensive approach towards the phishers, it must ensure that the data it submits is indistinguishable from real phished data. It also must ensure that clients have incentives to join Humboldt. Last, it also must not bring disruptions or harm to innocent parties, especially the web hosting company that is unknowingly hosting the phishing site and Humboldt's own clients. For the rest of this section we discuss how Humboldt meet these requirements.

A. Indistinguishability Between Humboldt and Victim Submissions

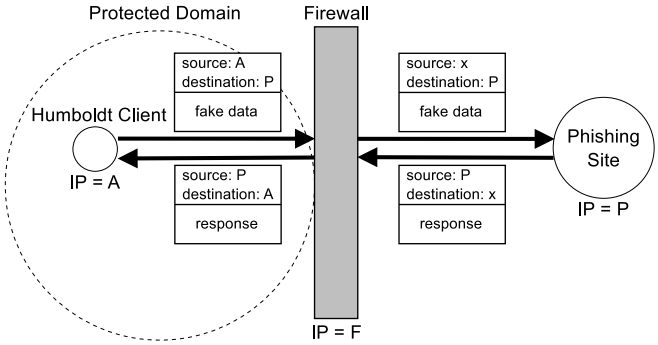
Humboldt must assume that every phisher is aware of Humboldt. For every piece of stolen data it collects, a phisher can check its source, its pattern of submission, or its content, to determine whether the data is from Humboldt or from a real phished user. If Humboldt submissions can be easily identified based on log file information on the phishing server, then all the fraudulent submissions from Humboldt can be readily removed. Therefore, the design of Humboldt must consider how to make sure its submission of fake data is undetectable in all three aspects of the fake data, i.e., the source, the pattern, and the content.

In filtering out submissions from Humboldt, the most obvious filtering attribute that a phisher may adopt would be IP address. Submissions from Humboldt, therefore, would have to be made from a pool of machines, not a central server. This requires Humboldt to operate on a distributed nature, which it does using individual client machines. The phisher may then try to learn the individual IP addresses of Humboldt clients; more specifically, since Humboldt clients may be more aggressive in submitting data than real phishing victims, a phisher could watch where the submissions are from and assume those who submit more frequently are Humboldt clients (the phisher could even do so across multiple phishing sites to be more statistical). Humboldt addresses this issue from several aspects. First, in today's Internet many machines are behind NAT boxes, and all machines behind a NAT box will appear to be from the same IP address. If the phisher does not accept multiple visits from an IP address, when multiple victims from behind the same NAT box submit to the phishing site, all of their submissions will be ignored—a loss to the phisher but a benefit to users. If Humboldt selects a client that is behind a NAT box, and this client is aggressive and submits frequently to a phishing site, all machines behind the NAT box will be immunized from the phishing. Second, a Humboldt client can also use the DHCP protocol (which is also very common on today's Internet) so that every time it submits to a phishing site it uses a different IP address. The Humboldt client can log its IP addresses in the previous submissions. Third, if a Humboldt client is behind a firewall that is not performing NAT, the firewall can be configured to collaborate with Humboldt. As shown in Figure 1(a), the firewall can act as a NAT box for all communications to phishing sites identified by Humboldt, achieving the same effect as the first case above. Alternatively, whenever the Humboldt clients communicate with phishing sites, the firewall could replace the IP of Humboldt clients with a random IP from the local domain (Figure 1(a)). With this method a single Humboldt client could effectively act as multiple clients, up to the number of IP addresses in the local domain.

Humboldt must also adopt a submission pattern that is undetectable to the phisher, or simply display no pattern in its submission. For example, it must avoid simultaneous submission of a constant number of forged credentials in a batch as done in BogusBiter [1]. To achieve this, Humboldt's central server coordinates the submission process of individual volunteer clients. For each instruction that Humboldt issues, Humboldt attaches a time parameter specifying when this associated particular entry should be submitted. Moreover, even though fake data is automatically submitted, every Humboldt client complies with the HTTP protocol so that it is impossible for a phisher to check server logs or HTTP headers to differentiate the origin of a submission For example, before submitting any form data, every Humboldt client will issue an HTTP GET request; a form submission without a previous GET request would be a give-away that an automated system is in place.



(a) Firewall acting as a NAT box.



(b) Firewall replacing the IP address of a Humboldt client with a random local IP address.

Fig. 1. Firewall configured to collaborate with Humboldt.

Humboldt clients can further mimic the natural delay in submission as the user reads, processes, and fills in the page; this delay can be modeled as a randomized function with respect to the amount of visible text on the screen. In addition, the fact that Humboldt clients have a diversity of connection speeds, HTTP header information, and browser brand and version numbers also makes data submissions look real.

Finally, Humboldt must ensure its fake data is similar to real phished data. To do so, Humboldt generates fake user names and passwords carefully from a dictionary of English words, along with occasional numbers in either of the fields. The result of such a combination closely mimics real life user names and passwords. It is possible to even accommodate a victim site's password requirements, such as the existence or the amount of capital letters, numbers, and special symbols, into this generation algorithm. Should a credit card or bank card number be needed, Humboldt can also generate sensible such card numbers according to the standardization of the numbering scheme, such as one that is generated by the Luhn algorithm. The intention is to make the phishers either unsure of the received data, or make it necessary to have the data checked for validity with the corresponding legitimate corporation or organization, exposing the phishers themselves by doing so.

B. Incentives for Clients

Humboldt clients who are responsible for submitting poisonous data to phishing sites must have a strong incentive to participate in Humboldt. Humboldt provides such incentive from several aspects. First of all, the legitimate web site that is a victim of a phishing attack will have a strong incentive to urge everyone of its users to become a Humboldt client. Victim web sites used to be able to do virtually nothing except warn their users, which was ineffective, and phishers continued stealing credentials. With Humboldt, they now can inject poison into the phisher's database and further identify the phisher when they use Humboldt-injected, fake data to log in to legitimate web sites. Second, every Humboldt client will be in a win-win situation whether or not it is known by a phisher to be a Humboldt client: If unknown by phishers, the client can submit fake data to phishing sites aggressively without limit, maximizing the damage it can do to the phishers; otherwise, the phisher will not take the input from this client, immunizing this machine—and even those who share the same IP address behind a NAT box or firewall (see Section II-A)—from phishing attacks.

C. Minimal Harm to Third Parties and Humboldt Clients

If not careful, Humboldt's distributed submission system could be misused as a DDoS system. If there is a large pool of Humboldt clients and they all try to submit at the same time, it could overwhelm the server hosting the phishing web site. Since a majority of phishing sites are hosted on commercial hosting services (either by paying for the service or compromising a server or a web site), Humboldt should not direct its clients to perform an equivalent of a DDoS attack. This requirement coincides with the submission pattern requirement discussed in Section II-A, as a flood of submissions would also produce a noticeable anomaly in the log files of phishing sites and allow for a more effective elimination of Humboldt entries. Humboldt meets this requirement in the same way as the submission pattern requirement: through the coordination of the central server. As Humboldt's central server coordinates the submission process of individual clients and avoids flooding fake data from all clients to the same phishing site, Humboldt will have minimal disruptions to the hosting services.

It is also important not to bring Humboldt's clients to any danger because of their participation in Humboldt. Humboldt clients expose some information about their

machine to the phisher, such as their IP address, web browser version, and the operating system. However, this information cannot directly lead to attacks toward the clients.

III. Architecture

A. System Overview

The Humboldt system is composed of a central server and multiple clients. The client machines are responsible for submitting fake data to phishing sites, and the server is further broken up into three major components, handling the input, processing, and output of the system, respectively:

- *Phishing site profiler* that examines known phishing sites, develops a data submission profile for them, and populates them into the Humboldt database;
- *Humboldt database (HumboldtDB)* that stores the profile for each phishing site, generates fake data for submitting to each phishing site, as well as logs fake data submitted to each phishing site for forensics analysis; and
- *Feed creator* that coordinates all the clients' activity and distributes data to the clients.

As shown in Figure 2, Humboldt's working procedure based on these components is as follows. First, the phishing site profiler receives a list of active phishing sites from a feed such as the one provided by PhishTank [2] or contracted mail servers that gather phishing URLs from phishing emails. For each phishing site URL, it fetches the HTML page, makes sure it exists, and analyzes the page to create its profile, including determining the fields in the page and what they expect as input. For every new phishing site profile, HumboldtDB then determines the number of Humboldt submissions needed, generates that amount of fake data, and passes them along with other instructions through the feed creator to Humboldt clients. The clients then submit the Humboldt-generated fake data to the phishing site at the predetermined time using mechanisms that make their submissions appear indistinguishable from legitimate users.

B. Phishing Site Profiler

The main purpose of the phishing site profiler is to process and profile the web page of any phishing site in order to properly submit forged data to it. In particular, it needs to determine the method for data submission (e.g., GET or POST) and the field type of every field on the web page (e.g., user name, password, credit card number, or PIN number). While the purpose of every field may be clear to a human user visiting the page, it can be hard for Humboldt to discern using a simple contextual analysis of the underlying HTML. To compound this problem, phishers can also try to make their sites harder to classify through polymorphic field names (though this may improve phishing site detection). On the other hand, despite the challenges facing an automatic classification system, there are trends and popular techniques which point to an effective solution. For example, 52.6% of phishing sites reported on PhishTank are Rock-Phish sites [3], following a predictable format that can be detected and classified.

Humboldt uses automatic classification to handle the easily recognizable phishing sites and relies on manual classification for the rest. In particular, we start out with automatic detection for the phishing sites that attempt to capture user names and passwords, as they have easily recognizable HTML patterns. With the analysis of the Rock-Phish sites, the percentage of sites which can be classified automatically should rise significantly. Meanwhile, the manual evaluation of phishing sites which are not automatically classified can also be greatly sped up using an interactive plug-in to the web-browser, which automatically creates profiles based on the user's identification of the fields.

C. Humboldt Database (HumboldtDB)

HumboldtDB is the data management center for the entire system. It generates and records all the fake data distributed, as well as the destinations of such data, and interacts with the feed creator to coordinate data submission among clients. Because of the amount of information recorded, HumboldtDB can provide invaluable information for tracking phishing behavior and performing forensic analysis, especially when a phisher uses the fake data injected to its phishing site.

HumboldtDB is the source of all the fake data submitted by the Humboldt system. The fraudulent data used can be classified into several different categories such as user names, credit card numbers and their associated CVV code and expiration dates, passwords and addresses. Each category of data has its own requirements for appearing legitimate (e.g., credit card numbers should have valid Luhn digits, passwords are often dictionary based), and are generated using specialized algorithms designed to mimic legitimacy. As the generation of data is not CPU-intensive, Humboldt is able to continuously generate fake data to provide them to the feed creator for distribution.

HumboldtDB must ensure proper coordination between clients. When a new phishing site is entered by the profiler, HumboldtDB calculates a randomized

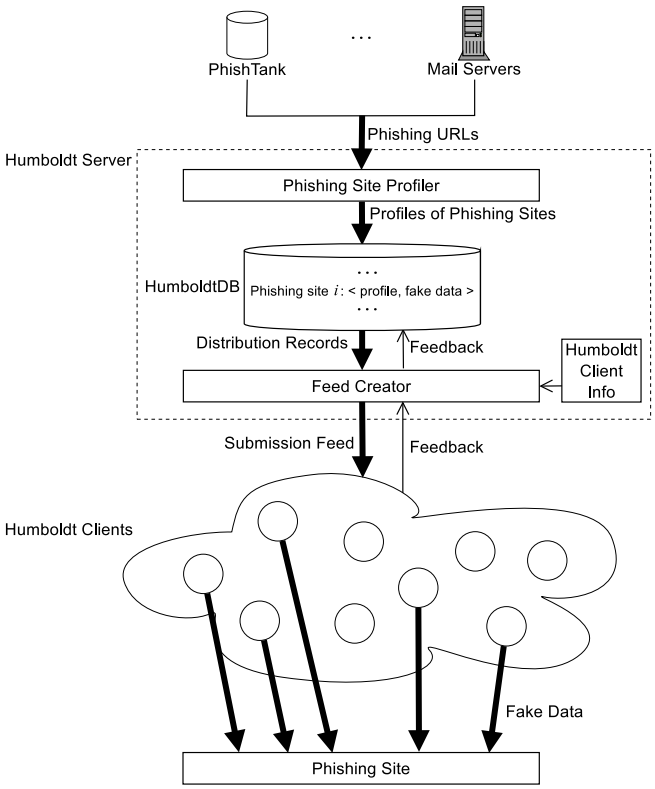


Fig. 2. Humboldt architecture.

submission schedule that follows the trend of expected phishing victims, estimated at 25 on average for the first day, and around 10 each day thereafter according to [3], in order to minimize the possibility that phishers can detect the introduction of Humboldt data. The data is distributed to the feed creator which then acts upon the schedule.

HumboldtDB is also the primary repository for all the information about submissions made. That is, it logs all fake data submitted, including the destination sites of those submissions. Since it is not hard to map a phishing site with the legitimate site that it tries to impersonate (e.g., the verification of a phishing site usually involves a positive identification by a human; a mapping can be made easily at that stage), Humboldt can inform each victim site of all the fake data submitted to the phishing sites that impersonate that victim site. The victim site can then identify phishers once it sees activities using these fake data. Security researchers and law enforcement personnel can also benefit from the logged submission information; the flow of the fake data can also be tracked as they are exchanged between different parties. Note since logged submission data is also of great interest to phishers, Humboldt provides such data only to trusted victim sites.

D. Feed Creator

The feed creator is the "output" component of the Humboldt server, responsible for passing to and coordinating actions among Humboldt clients with data passed from HumboldtDB. The feed creator also receives feedback information from Humboldt clients, such as whether a submission is successful, and passes them back to HumboldtDB.

More specifically, to make the distribution more coordinated, the feed creator keeps records of online Humboldt clients and some user data such as how frequent the client is able to make submissions. For example, clients behind a NAT box or a Humboldt-aware firewall may choose to submit as many as possible for the reasons we discussed in Section II-A, but other clients may make at most one submission per phishing site. With such knowledge the feed creator can make appropriate distribution decisions about which clients to involve and, after receiving fake data from HumboldtDB, passes them to selected clients to submit. The feed creator component also waits for feedback data, such as the success of the submission, or HTTP errors that the client may encounter.

E. Humboldt Clients

Humboldt clients are tasked with submitting fake data to phishing sites while obscuring the system's footprint. In order to be indistinguishable from the submission of phished users, the submission of data from Humboldt clients should mimic the behavior of a phished user close enough by following the design as discussed in Section II-A.

In fact, every client can be simply implemented as a web browser plug-in. Since the browser embeds browser data (such as the browser version) along with HTTP requests, using a browser plug-in will help exhibit a realistic distribution of visitor profiles to the phishing site. Moreover, implementing Humboldt clients as a web browser plug-in allows non-technical users help contribute to the Humboldt system through a simple install. The plug-in can operate without any user input, but simply interact with Humboldt and make submissions while the user performs other activities.

Joining Humboldt is easy. After a host installs the Humboldt plug-in, it can initiate a join request to the Humboldt server (more specifically, the feed creator), which then can add this host as a Humboldt client. (We discuss security concerns in adding clients in Section IV-B.)

The Humboldt server has a light-weight mechanism for maintaining its clients. Recall every client needs to send feedback to the feed creator after it is instructed to submit fake data. If a client does not send feedback in multiple instances (currently set to three times that happen during different time windows), the feed creator will simply remove that client.

IV. Discussion

It is not enough for Humboldt to just have a short-term success as phishers can attempt to neutralize Humboldt's success in fighting against phishers. In this section, we identify countermeasures phishers can employ, and discuss how Humboldt can deal with them. Specifically, we discuss how phishers may embed a unique ID in their phishing URLs, pretend to be Humboldt clients, or compromise Humboldt's phishing site profiler or HumboldtDB, and what Humboldt's solutions are.

A. Phishing URLs with Unique Identifier

One potential problem for Humboldt is phishing URLs that carry a unique identifier. When a phisher sends out phishing emails in a phishing campaign, the phisher can make sure the phishing URL in every phishing email contains a unique ID, e.g. http://www.phisher.com/login.cgi?id=12345. (Note that such unique IDs are also used in certain legitimate cases, such as IDs for forgotten password emails, or account activation links.) The phisher can also log (or make sure it can derive) all the email addresses it targeted, together with the unique ID for each email address. As a result, the phisher can expect most of its phishing victims would simply click on a phishing URL with a unique ID—and doing so only once—to get directed to the phishing site, and only target such victims.

Such a design from phishers would force Humboldt to also submit its fake data to phishing URLs with unique IDs, and to use every unique ID only once. If a submission to the phisher's web site is based on a request address without an ID (e.g. http://www.phisher.com/), or a non-existent ID according to the phisher's knowledge, (e.g. http://www.phisher.com/login.cgi?id=nonexistent), or is a duplicate submission using the same phishing URL with an ID, the phisher can discard any information entered from the submission.

To deal with this strategy from phishers, Humboldt has to gather phishing URLs with unique IDs. Note that phishing links reported by services such as PhishTank may not contain many phishing URLs with unique IDs. Instead, Humboldt can team up with mail servers to collect phishing URLs, especially those with unique IDs, and then have Humboldt clients submit fake data using such phishing URLs. Phishers, upon receiving a submission, will not know if that particular submission is from Humboldt or a phished user since the ID is going to be valid in their local database.

If phishers learn which mail servers are collaborating with Humboldt and discard all submissions resulting from phishing emails toward those mail servers, then the entire mail domain of those servers will be immunized from phishing. This is a significant benefit for the domain, especially when the domain is as large as having millions of users.

B. Authenticity and Integrity of Humboldt Clients

Humboldt must take caution in adding a new client (but without being too restrictive to lose many potential clients). For example, Humboldt needs to make sure the IP address of every client is not spoofed. Humboldt also needs to make sure that it is not a bot from a botnet that is trying to sign up, probably by using a CAPTCHA technique. This way, if a phisher tries to join Humboldt as a client to compromise Humboldt, it can, but the phisher's machine will only sign up as one specific machine with an IP address on the record.

At the same time, certain clients—including any client belonging to a phisher—could misbehave, but Humboldt

is tolerant of that. A client could fail to submit fake data assigned to it, thus failing to pollute the database of a phisher. A client could also submit the fake data more than once, causing the phisher simply to ignore the submission. A client may also modify the fake data, still polluting the database of a phisher but making the fake data not trackable by Humboldt. However, all these cases will not affect other clients who do behave properly. No matter how many clients misbehave, the well-behaved clients will be able to continue disrupting phishing effectively. There is also no real data passed around anywhere in the system, so no misbehaviors can put any real data in danger. The possible compromise or malicious modification of client side software code is a likely indication of a compromised machine altogether, and such vulnerabilities fall under program and operating systems security, out of the scope of this paper.

C. Securing the Humboldt Server

The phishing site profiler, HumboldtDB, and the feed creator are the three critical components of the Humboldt server. Phishers can launch many security attacks toward them, but fortunately the possible types of attacks are common and already well-studied. For example, an obvious attack towards the Humboldt server is a DoS or DDoS attack. These types of attacks are very general and are widely studied as a separate research topic. We therefore do not discuss them here as they are out of our scope. The phishing site profiler needs to make sure its sources of phishing URLs are trustworthy, HumboldtDB must employ common database security solutions to make sure its data cannot be stolen or modified by phishers, and the feed creator must sign every dispatch to clients about what fake data to submit to a specific phishing site in order to guarantee the integrity of the dispatch.

D. Legal Issues

Humboldt does not cause harm to a web site or the identity of an individual. By hosting a web site online with its forms, the web site (who is probably a phisher) invites the public to input data into the form and make submissions. Since these web sites are public, anyone is allowed to submit data into these forms. Humboldt is merely a program that automates this data input and submission process as if a human is performing these actions. There is then no harm done to the web sites in this sense. A Humboldt client also is not inadvertently entering a contract for its client machine since, even if an EULA is present on the web site, it is unclear and debatable whether an agreement is present, especially if a phishing site is being dealt with. As a result, no contracts are entered. On the other hand, the information that is being submitted is not causing harm to a potential online retailer should a phisher uses it to make a purchase, because only fake generated data (i.e. not real data) is submitted to the phisher. The probability of having a coincidental match in credit card number, CVV, expiration date, and cardholder name to those of a real person's is, for practical purposes, zero. The same argument can be applied to show that Humboldt does not bring identity theft and thus bringing harm to an individual.

Since Humboldt generates fake data for distribution, the legality of such actions is also worth discussing. For credit cards, the algorithm for generating and validating a Luhn number is in the public domain, as it is created by IBM scientist Hans Peter Luhn and described in U.S. Patent 2,950,048, filed on January 6, 1954, and granted on August 23, 1960. The purpose of Luhn numbers is simply to provide a 16-digit number that contains error-detection capabilities. The outcome that Humboldt tries to achieve is just to have the phisher accept the number, believing it is valid. It is no different to generate these Luhn numbers compared to generating a seemingly valid fake address or name. Humboldt or its clients do not gain anything, financially or otherwise, from the distribution of these fake data to the phishers. It is also not expected that these fake information can provide the phishers with any financial gain.

V. Evaluation

In this section, we evaluate both the effectiveness of Humboldt and the performance of the prototype that we have implemented. For effectiveness, we have devised two probability models to arrive at theoretical results. These theoretical results give us insight on possible requirements of Humboldt for a successful deployment, including the amount of fake data to be submitted to an individual phishing site and the probability of identifying a phisher with such fake data. For performance, we have implemented a prototype and collected necessary performance overhead data on a desktop machine with reasonable hardware.

A. Theoretical Results

In order to measure how effective and efficient Humboldt is, we must have a definition of effectiveness and efficiency, followed by methods to properly evaluate them.

We define an effective solution to be one that has the highest chance of identifying phishers. Therefore our

metric is the *probability of success*, where success is the ability to identify a phisher's login attempt to a victim web site. We also define a solution to be efficient if it requires a minimal number of submissions. Our metric to measure efficiency is then *the number of fake entries Humboldt deems necessary to submit*.

For both the effectiveness and efficiency metric, we derive a model and analyze how Humboldt does in terms of the metric.

1) Probability of Success: Suppose that a phisher, after running a phishing campaign, is left with a database of submissions into its phishing web site. Let N be the number of total submissions that the phisher receives, and H be the number of entries within N that are fake data submissions from Humboldt. Of course, the phisher does not know what H is. The number of real phished entries is $N - H$. Further, on the phisher's side, we let $n < N$ be the number of entries that the phisher tries with the legitimate web site in order to verify the legitimacy of the received entries. (The phisher might not test all received entries due to the fear of being caught or simply lack of time.) Since Humboldt can provide victim web sites with the fake entries that it has instructed its clients to submit to their phishing counterparts, the victim web site can recognize phishers when they attempt to log in with a Humboldt entry. We let $k \in \{1, 2, \ldots\}$ be the number of Humboldt entries that a victim web site sees within a time period before concluding that it is a phisher trying to log in. For a web site with a large user base, since it is possible that a Humboldt-created fake user name and password pair might coincide with that of a real existing user in the victim web site, k could be chosen such that $k = 2$, i.e., the victim site identifies a phisher after seeing at least two login attempts with Humboldt entries. However, choosing $k = 2$ is already very conservative as it is extremely unlikely that this coincidence happens. A k value of 1 is sufficient for most cases.

Let X be the random variable for the number of Humboldt entries that a phisher tries with the victim web site. We then have the probability

$$P\{X = k\} = \frac{\binom{H}{k}\binom{N-H}{n-k}}{\binom{N}{n}},$$

with k being the threshold to identify phishers. The probability of detection is actually

$$P\{PhisherDetected\} = P\{X \geq k\},$$

where the latter is a summation on k to an upper limit of H:

$$P\{X \geq k\} = \sum_{i=k}^{H} P\{X = i\} = \sum_{i=k}^{H} \frac{\binom{H}{i}\binom{N-H}{n-i}}{\binom{N}{n}}. \quad (1)$$

Considering a scenario where $N = 100$ and $n = 20$, we can use Equation (1) to derive the discrete probability values. As shown in Figure 3, we can see that Humboldt is effective in catching phishers: With $k = 2$, if the phisher randomly picks and tries 20 out of a total of 100 entries in its database, we will have 90% probability of detecting this phisher by only submitting 17 Humboldt entries to the phisher's database. With $k = 1$, then only 8 Humboldt entries are needed for the same 90% probability. The trends displayed in the same figure are mostly sigmoidal, meaning that in most cases adding just a few more Humboldt entries can greatly improve our chance of detecting the phisher.

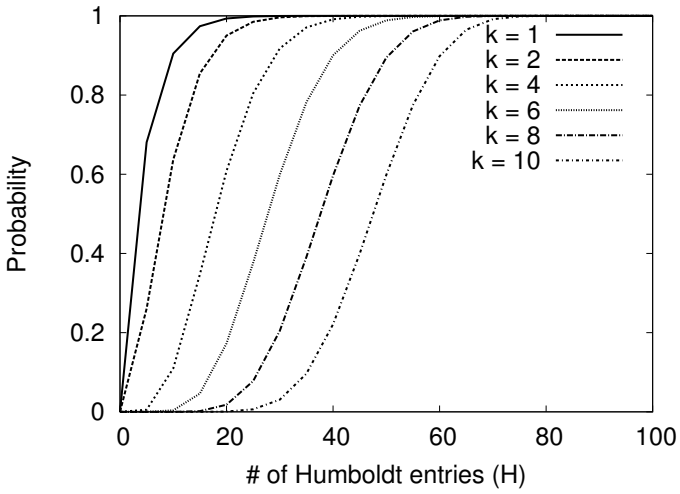


Fig. 3. Probability of identifying a phisher based on the number of Humboldt entries in the phisher's database (the phisher has a database with 100 entries and it tries 20 out of 100 randomly).

Similarly, consider a scenario with $N = 100$ and $H = 20$, where we fix the number of Humboldt entries in the phisher's database, and observe how the probability of identifying the phisher may vary with the different n values, i.e., the size of the random sample that the phisher picks to try on the victim site. This scenario is shown in Figure 4. From this figure we can see that with $k = 2$, if Humboldt successfully submitted 20 entries into the phisher's database that has totally 100 entries, the probability of identifying the phisher will be 90% if the phisher randomly picks just 17 entries to attempt at the victim web site. And with $k = 1$, the phisher can be detected with the same 90% probability if it submits only 8 entries. Since the distributions are sigmoidal again and a slight increase in the number of attempts at the victim site would significantly increase the probability of identifying the phisher, one interesting observation here is that the phisher has to be very cautious in increasing its number of attempts at the victim site. On the other hand, to cause Humboldt to have a lower probability of

success, the phisher has to decrease its number of login attempts. Note that the probability value stays above 50% even if the phisher only attempts eight times for $k = 2$, and three times for $k = 1$.

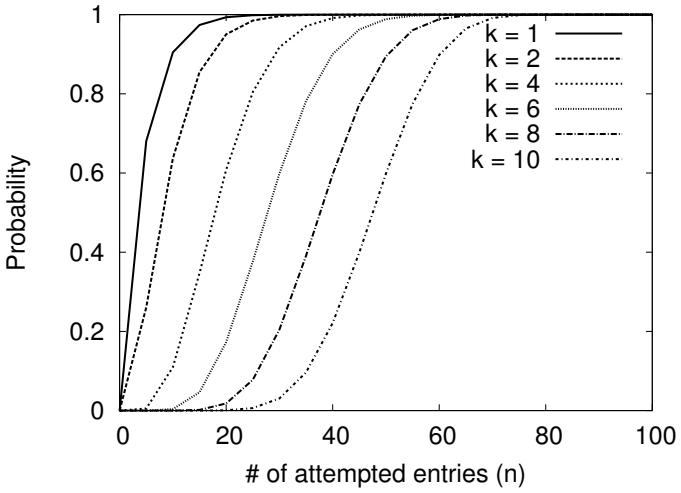


Fig. 4. Probability of identifying a phisher based on the size of the random sample chosen by the phisher (the phisher has a database with 100 entries and it tries 20 out of 100 randomly).

It is interesting to note that Figure 3 is identical to Figure 4 (with their respective fixed variable values). This is more of a property rather than coincidence: we can in fact show that although the two equations have different summations, the individual terms end up being the same if each term in the summation is enumerated. What this means is that the number of Humboldt submissions is just as important as the sample size that a phisher chooses to try. In the case that phishers decrease their sample sizes, Humboldt can easily deal with it by linearly increasing its size of submissions.

Overall, Humboldt is effective: With a reasonable number of Humboldt-submitted entries, the probability in detecting phishers is high as phishers use Humboldt entries in attempted logins at victim web sites. Even if a phisher concedes with fewer attempts, Humboldt can always injects more entries to the phisher's database to maintain the high probability of successfully identifying the phisher.

2) Number of Submissions: To analyze the number of submissions that Humboldt needs to inject into a phishing site, we define another model where the total number of submissions is not fixed as defined above, but rather a variable based on the scale of a phishing campaign and user response rate. Without loss of generality, assume the phisher sends out M phishing emails targeting a specific victim web site as part of a phishing campaign, and let β be the response rate of email users who have received one of M emails and followed through the phishing link provided. The number of entries from real victims is then $M \cdot \beta$, and we can represent the total number of entries that the phisher receives as $(H+M \cdot \beta)$. Plugging $(H + M \cdot \beta)$ into Equation (1) by replacing N, we now have:

$$P\{X \geq k\} = \sum_{i=k}^{H} P\{X = i\} = \sum_{i=k}^{H} \frac{\binom{H}{i}\binom{M \cdot \beta}{n-i}}{\binom{H+M \cdot \beta}{n}}. \quad (2)$$

Based on Equation 2, we can analyze the number of submissions needed from two angles. First, as shown in Figures 5 and 6, we can study how the probability of identifying the phisher varies based on the scale of a phishing campaign. Both figures assume $n = 20$ and $k = 1$, but in Figure 5 we have $H = 20$ and in Figure 6 we have $H = 100$. With a fixed H, it is obvious that the detection probability decreases as the size of the phishing campaign increases. Moreover, while the detection is not effective with $H = 20$, increasing H to 100 will greatly improve the efficacy of detecting phishers. Note for most phishing campaigns we evaluate, this value of H is about one magnitude smaller than the number of real phished entries that a phisher obtains.

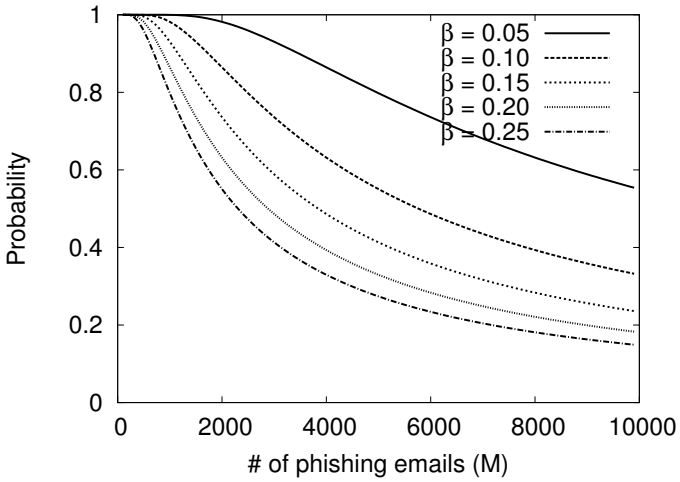


Fig. 5. Probability of identifying a phisher based on the number of phishing emails ($H = 20$, $n = 20$, and $k = 1$).

Secondly, we can study in order to achieve a given probability of identifying a phisher, how many Humboldt entries are needed for different sizes of phishing campaigns. Figures 7 and 8 plot the results for probabilities of 0.8 and 0.9, respectively. Both assume $k = 1$ and $n = 20$. As expected, the one magnitude difference between these two variables still hold, and we confirm that Humboldt can achieve high probabilities of success with a realistic number of clients. The trends between these two variables, the number of entries Humboldt needs to submit versus the size of a phishing campaign display basically a linear relationship, indicating that

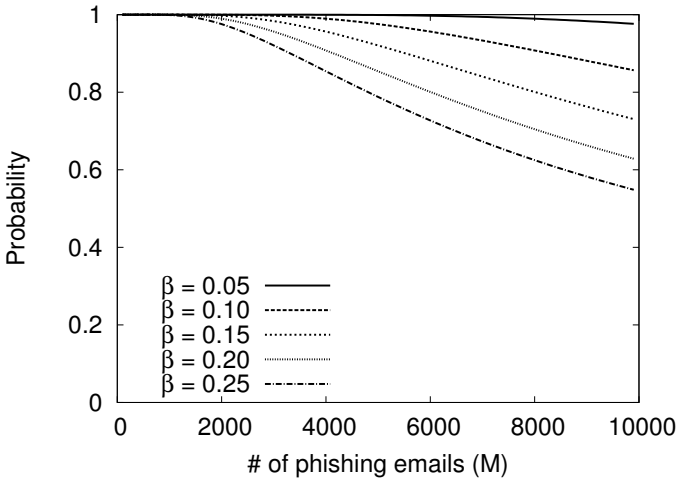


Fig. 6. Probability of identifying a phisher based on the number of phishing emails ($H = 100$, $n = 20$, and $k = 1$).

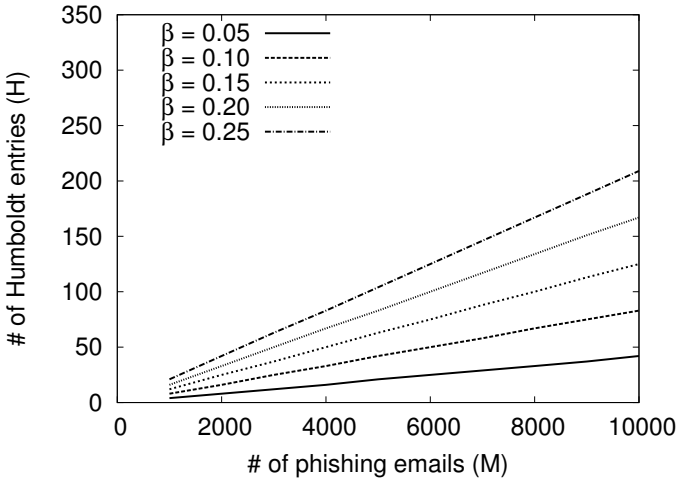


Fig. 7. Estimation of number of Humboldt entries needed for detection probability of 0.8 ($k = 1$ and $n = 20$).

In summary, Humboldt is efficient because, with a reasonable response rate of phishing emails, Humboldt can handle large-scale phishing campaigns in the range of thousands of phishing emails with one fewer magnitude of fake entries submitted, in the range of low hundreds. This capability of handling large phishing campaigns with minimal resources is crucial to the success of Humboldt.

B. Humboldt Prototype Evaluation

We have built and implemented a prototype of Humboldt. In this prototype we use PhishTank as the provider for phishing URLs, and the Humboldt server is a Pentium 4 desktop with a 3-GHz CPU and a 1-GB memory. From our tests and results, the prototype achieves

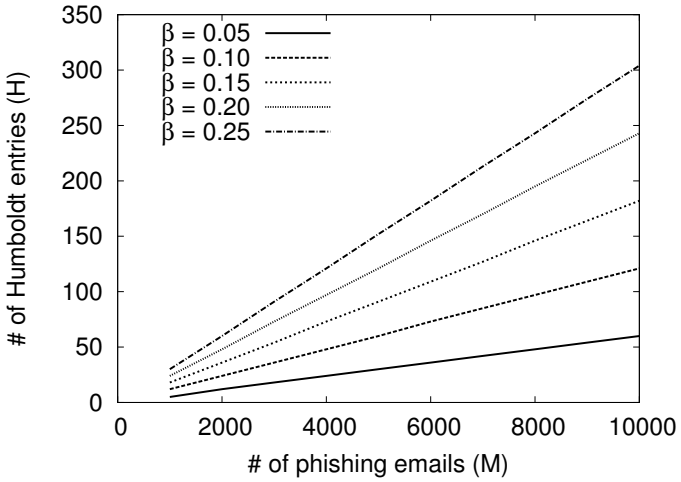


Fig. 8. Estimation of number of Humboldt entries needed for detection probability of 0.9 ($k = 1$ and $n = 20$).

performance expectations. On the Humboldt server side, we mainly studied the rate that new phishing sites appear and the speed at which Humboldt processes them; on the Humboldt client side, we mainly studied the extra network traffic due to a client's participation in Humboldt.

It is crucial for the Humboldt server to be able to keep up with the pace of the appearance of new phishing web sites. By fetching the hourly updates of PhishTank, we observe that although the number of online phishing sites at a given moment is high (over 10000 as of June 2009), the number of new online sites on an hourly basis is quite low. From a period of 27 hours in March 2009, we observed 99 new phishing sites reported by PhishTank through a 27-hour period, an average of 3.7 sites per hour. (Although PhishTank is not complete in listing all the phishing sites, it provides one of the most popular listings, and the number of phishing sites it provides is on the same order as its counterparts.) This low value is advantageous for Humboldt since it only needs to profile each phishing site once, and the profiling stage is the most CPU intensive stage for the Humboldt server.

Since PhishTank only updates the feed each hour, we evaluate the Humboldt server's overhead on an hourly basis as well. First, with our prototype, processing a new phishing site—including downloading its web page and profiling it—will cost on average 1.769 seconds. Here, the latency for downloading a web page will depend on the locations of the Humboldt server, the phishing site, the relevant DNS servers, as well as network congestion level; the cost of profiling a web page can also depend on the contents of the page. In addition, every hour the server will incur an overhead that costs on average 28.269 seconds, including deciding if certain phishing

URLs in the current hour still point to the same web pages that have been processed before. As a result, theoretically our prototype of Humboldt can process an average of 2019 new sites per hour (i.e., (3600-28.269)/1.769), a number that is much larger than the real number of new sites appearing each hour (which is 3.7 on average as mentioned above).

On the client side, we want to minimize network traffic that Humboldt caused on the client machines. Since instructions from Humboldt to each client are minimal, we focus on the size of phishing web pages since each client must pretend to visit the phishing page before a submission is made. After studying 130 phishing sites we found that phishing sites have a mean size of 33145 bytes with a standard deviation of 53842 bytes, including their HTTP headers, images, CSS, external JavaScript, and associated files. Given on average there are only 3.7 new phishing sites per hour to handle as calculated above, a Humboldt client would thus incur a reasonable amount of traffic overhead with today's technology.

VI. Related Work

Several prevention mechanisms and techniques exist to help users not become phished. These solutions are typically deployed as web browser or email client plug-ins. For example, one such type of web browser plug-ins defends against phishing through password obfuscation and password management [4], [5], [6]. Some browser plug-ins use heuristic-based approaches to detect phishing web sites, using machine-readable properties of a web site or a browser's data such as its browsing history [7], [8], [9], [10], [11]. Email client plug-ins can also identify phishing emails if they contain links to fraudulent sites [12], [13].

Prevention can also be done at legitimate web sites. Solutions such as SiteKey rely on mutual authentication where the legitimate web site can provide an image that a phishing site does not have access to [14], [15], [16]. This challenge-response technique is somewhat effective, but an inexperienced user can still be phished if the phishing site does not show the cue image at all, even if the legitimate version of the web site has such component. Other more advanced cueing techniques also exist. One example is an authentication solution that uses random visual keypads for password entry [17].

There are also techniques that can be deployed on both a user's machine and a legitimate web site. The most common method is web site identification using whitelists or blacklists. The web browser (or a plug-in of the browser) can check if each site navigated to exists on a whitelist or a blacklist [18], [19], [20]. In order for blacklisting to be effective, the blacklist database has to update itself quickly and frequently. We cannot blacklist just the IP addresses since sometimes many sites share one same IP address in commercial hosting services. In recent years new phishing techniques, such as rock-phish web sites and fast-flux domains, begin to appear getting around blacklisting. As a result, blacklisting is effective against identified phishing sites (the browser can give highly visual alerts to the user), but it is not exhaustive and leaves many phishing sites unaffected.

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, as discussed in [3], [21]. It is also possible to stop DNS resolutions earlier on in the DNS resolution chain so that the domain name of a phishing site is not resolved at all. Take-down is more of a reactive strategy rather than preventative, so usually by the time a web site is indeed taken down, some damage has already been done and the phisher is able to obtain personal information from victim users up until that point. Another problem with this take-down strategy is that it lacks incentive for everyone except for the victim site itself. Users who can identify a site as a phishing site do not achieve personal gain by notifying the bank or hosting company; and similar argument stand for hosting companies.

VII. Conclusions

Phishing as one of the most severe security threats on today's Internet has caused widespread identity theft and immense finance loss. Solutions to phishing, however, have mostly been passive and are often about detecting phishing sites and preventing users from accessing them. Although offensive approaches have been taken, such as submitting bogus credentials to phishing sites or simply requesting to take down a phishing site, such approaches are either easy to thwart or require too many manual operations.

We present a solution in this paper, called Humboldt, that can inject poisonous data into phishing sites to disrupt phishing activities. It can profile phishing sites to decide what fake data to submit, and have its clients do so without looking different from real phished users. Humboldt not only causes phishers to have difficulty in obtaining sensitive data from real victim users, but also makes it possible to track phishers. We have also evaluated Humboldt and shown it is both effective and efficient; the evaluation of the Humboldt prototype also shows the system has a low performance overhead.

REFERENCES

[1] C. Yue and H. Wang, "Anti-phishing in offense and defense," in *ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 345–354.

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

[3] T. Moore and R. Clayton, "An empirical analysis of the current state of phishing attack and defence," in *Workshop on the Economics of Information Security, 2007*, June 2007.

[4] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions," in *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2005, pp. 2–2.

[5] E. Kirda and C. Kruegel, "Protecting users against phishing attacks with AntiPhish," in *Proceedings of the International Computer Software and Applications Conference*, 2005, pp. 517–524.

[6] K.-P. Yee and K. Sitaker, "Passpet: Convenient password management and phishing protection," in *Proceedings of the Symposium on Usable Privacy and Security*, 2006, pp. 32–43.

[7] 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*, February 2004.

[8] CallingID Ltd., "CallingID Toolbar," visited October 10, 2007. [Online]. Available: http://www.callingid.com/DesktopSolutions/CallingIDToolbar.aspx

[9] Mozilla Foundation, "Firefox phishing protection," visited October 10, 2007. [Online]. Available: http://www.mozilla.com/en-US/firefox/phishing-protection/

[10] Microsoft Corp., "Anti-phishing technologies overview," visited October 10, 2007. [Online]. Available: http://www.microsoft.com/mscorp/safety/technologies/antiphishing/overview.mspx

[11] Netcraft, "Netcraft anti-phishing toolbar," visited October 10, 2007. [Online]. Available: http://toolbar.netcraft.com/

[12] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proceedings of the international conference on World Wide Web*, 2007, pp. 649–656.

[13] B. Adida, S. Hohenberger, and R. L. Rivest, "Fighting phishing attacks: A lightweight trust architecture for detecting spoofed emails," in *Proceedings of DIMACS Workshop on Theft in E-Commerce: Content, Identity, and Service*, April 2005.

[14] N. Agarwal, S. Renfro, and A. Bejar, "Yahoo!'s Sign-in Seal and current anti-phishing solutions," in *Proceedings of Web 2.0 Security & Privacy Workshop*, May 2007.

[15] Bank of America, "How Bank of America SiteKey works for online banking security," visited October 10, 2007. [Online]. Available: http://www.bankofamerica.com/privacy/sitekey/

[16] The Vanguard Group, "Learn how Vanguard protects your accounts," visited October 10, 2007. [Online]. Available: https://personal.vanguard.com/VGApp/hnw/help/SecurityVGProtectsAcctsContent.jsp

[17] Tricerion, "Tricerion SMA product overview," visited October 10, 2007. [Online]. Available: http://www.tricerion.com/products/productOverview.php

[18] Earthlink, Inc., "Earthlink tools for Firefox browser," visited October 10, 2007. [Online]. Available: http://www.earthlink.net/software/nmfree/firefox/faq/

[19] eBay, Inc, "eBay: Buyer's tools: Toolbar," visited October 10, 2007. [Online]. Available: http://pages.ebay.com/ebay_toolbar/

[20] Google, Inc., "Google safe browsing for Firefox," visited October 10, 2007. [Online]. Available: http://www.google.com/tools/firefox/safebrowsing/

[21] T. Moore and R. Clayton, "Examining the impact of website take-down on phishing," in *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*. New York, NY, USA: ACM, 2007, pp. 1–13.