

# Evaluation of Online Resources in Assisting Phishing Detection

Kaigui Bian\*, Jung-Min “Jerry” Park\*, Michael S. Hsiao\*, France Bélanger†, Janine Hiller‡

\*Department of Electrical and Computer Engineering

†Department of Accounting and Information Systems

‡Department of Finance, Insurance and Business Law

Virginia Tech, Blacksburg, VA 24061, USA

{kgbian,jungmin,hsiao,belanger,jhiller}@vt.edu

**Abstract**—Phishing is an attempt to fraudulently acquire users’ sensitive information, such as passwords or financial information, by masquerading as a trustworthy entity in online transactions. Recently, a number of researchers have proposed using external online resources like the Google PageRank system to assist phishing detection. The advantage of such an approach is that the detection capability will gradually evolve and improve as the online resources become more sophisticated and manipulation-resistant. In this paper, we evaluate the effectiveness of three popular online resources in detecting phishing sites—viz, Google PageRank system, Yahoo! Inlink data, and Yahoo! directory service. Our results indicate that these online resources can be used to increase the accuracy of phishing site detection when used in conjunction with existing phishing countermeasures. The proposed approach involves examining the following three attributes of a target site (site being examined): (1) the credibility of the target site’s hosting domain, (2) the credibility of in-neighbor sites that link to the hosting domain, and (3) the correlation between the target site’s web category and its hosting domain’s web category. The aforementioned online resources by themselves are insufficient to address the phishing attack problem. We provide discussions on how each of those resources may be integrated with existing phishing detection techniques to provide a more effective solution.

## I. INTRODUCTION

Web-based information theft, more commonly known as a “phishing” attack, poses a significant threat to Internet users. In a phishing attack, phishers send unsolicited messages (e.g., emails or instant messages) to recipients, and lure them into giving out their private information at spoofed web sites (a.k.a. phishing sites). Phishing attacks cause enormous financial loss. The Gartner group estimates that the phishing-related loss in the U.S. in 2007 was more than \$3 billion [7].

The majority of phishing detection schemes attempt to detect fraudulent sites by observing content-related attributes (e.g., the site images, keywords, etc) [4], [11], and/or comparing the target site against a blacklist [5], [8], [15]. Purely content-based detection schemes are prone to incur a relatively high number of *false negatives* (i.e., failing to identify a phishing site) unless the schemes are highly “tuned”<sup>1</sup> to detect certain attributes of typical phishing

sites. However, such highly tuned schemes may bring about an increase in the number of *false positives* (i.e., falsely identifying a legitimate site as a phishing site). Blacklist-based detection schemes are likely to fail in identifying new phishing sites not included in the blacklist, which results in false negatives.

Recently, researchers have proposed using the Google Search and PageRank system [10] as part of a detection scheme to detect phishing sites [6], [13], [15], [23]. This approach takes advantage of the fact that most phishing sites share a common feature—i.e., their hosting domains have extremely low PageRank scores. Our study on web datasets revealed two other features common to most phishing sites: (1) the hosting domain<sup>2</sup> of a phishing web site generally has no inlinks from credible sites, and (2) the web category of a phishing web site is irrelevant to that of its hosting domain. Note that inlinks of a web site  $x$  are the incoming links from other web sites to this web site, and web sites pointing to  $x$  are called the *in-neighbors* or in-neighbor sites of  $x$ . Thus, it is often possible to detect phishing sites by examining these features using external online resources (i.e., information depositories). One noteworthy consequence of this approach is that the resulting phishing detection scheme “evolves” along with the online resources which evolve over time to become more sophisticated and manipulation-resistant.

Using datasets from reliable sources, we carried out experiments to evaluate three detection heuristics—viz, *domain credibility check*, *credible in-neighbor search*, and *web category comparison*. These heuristics utilize online resources to detect phishing sites. Specifically, the heuristics utilize the Google PageRank system, the Yahoo! Inlink data, and the Yahoo! Directory. From our study, we put forward the following two theses.

- 1) In certain scenarios, detection heuristics using online resources are very effective in identifying phishing sites and validating legitimate sites;
- 2) However, these heuristics by themselves are not sufficient to address the false positive problem. We surmise that a truly effective and practical phishing detection scheme will require a combination of multiple detec-

<sup>1</sup>Phishing detection schemes are often tuned to detect certain characteristics of phishing sites by adding one or more detection heuristics.

<sup>2</sup>For example, `www.example.com` is a domain name; and `http://www.example.com/index.html` is the URL of a web site.

Table I  
DEFINITIONS OF SIX TYPES OF TARGET WEB SITES.

Type	Definition
$T_1$	Legitimate sites on credible domains
$T_2$	Phishing sites on credible domains
$T_3$	Legitimate sites on obscure domains
$T_4$	Phishing sites on obscure domains
$T_5$	Legitimate sites on unknown domains
$T_6$	Phishing sites on unknown domains

tion techniques including the aforementioned heuristics.

The online resource-based heuristics proposed in this paper are complementary to existing phishing detection techniques and are unique in the sense that they leverage the power of the online resources in identifying and classifying online information. In this paper, we discuss how the proposed heuristics can be integrated with existing detection techniques (e.g., content-based analysis, white listing, blacklisting, etc.) to improve the overall detection performance.

The rest of this paper is organized as follows. Technical preliminaries are presented in Section II, followed by descriptions of the three heuristics in Sections III, IV, and V. In Section VI, we discuss related work. Finally, we conclude the paper in Section VII.

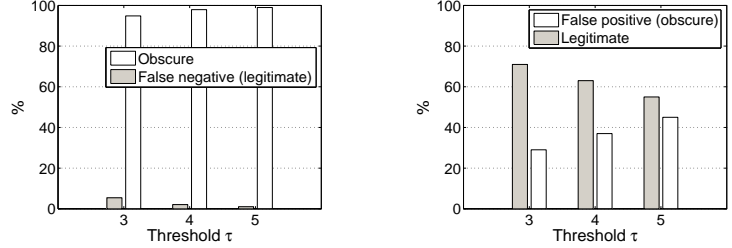
## II. TECHNICAL PRELIMINARIES

### A. Classification of Web Sites

In this paper, we use the PageRank score to quantify the *credibility* of a web site or its hosting domain. First, we classify the hosting domains of web sites into three categories according to the PageRank score: (1) A domain is considered to be *credible* if its PageRank score is equal to or greater than a threshold  $\tau$ ; (2) A domain is considered to be *less credible or obscure*, if its PageRank score is lower than the threshold  $\tau$ . Such domains may host legitimate sites or phishing sites, or both; (3) A domain is considered to be *unknown* if its PageRank score is unavailable or it has no credible in-neighbor sites. Then, we categorize target web sites (i.e., web sites being examined) into six types, as shown in Table I.

### B. The Fundamental Challenges

1) *False Positives*: Today, many of legitimate sites are hosted on obscure (but legitimate) domains and/or they are newly established sites. Some of these legitimate sites have some features in common with phishing sites. Our study of web datasets and existing detection methods confirmed that these legitimate, but “obscure” sites (those that belong in  $T_3$  and  $T_5$ ) are the main cause of false positives in phishing detection schemes. Detection heuristics based on site-content



(a) Results of phishing sites' domains; (b) Results of legitimate sites' domains.  
Figure 1. DCC results. DCC makes one of two decisions: “obscure” (PageRank score is less than  $\tau$ ) and “legitimate” (PageRank score is no less than  $\tau$ ).

analysis and/or blacklists have difficulty distinguishing these sites from actual phishing sites (those that belong in  $T_4$  and  $T_6$ ).

2) *False Negatives*: We also found that a significant number of false negatives are caused by cases in which phishing sites are hosted on legitimate (but obscure) domains ( $T_4$ ) or even on credible domains ( $T_2$ ). Existing detection schemes that check the legitimacy of a URL and the corresponding domain are likely to fail in identifying these web sites.

3) *Drawback of Purely Content-Based Detection*: As discussed before, content-based detection schemes can be circumvented by manipulating a phishing site’s content to minimize the chance of detection. To counter this phishing ploy, detection schemes should examine site attributes that are more difficult for a phisher to manipulate in addition to site content. These attributes could be evaluated by utilizing data in online information repositories that are provided by well-known and trusted companies, such as Yahoo! and Google.

### C. Experiment Datasets

The following two datasets were used to evaluate the proposed detection heuristics. As mentioned previously, those heuristics utilize the information provided by online information repositories.

*Dataset 1: Phishing sites.* We collected 387 “online and valid” phishing sites from PhishTank.com [12]<sup>3</sup>. We acknowledge that there is a probability—although this probability is small—that some of the sites classified by PhishTank.com as “online and valid” may not actually be phishing sites (cf. [9]). Therefore, we manually verified each one of the entries fetched from PhishTank.com to ensure that a collection of actual phishing sites were obtained.

*Dataset 2: Legitimate sites.* 3sharp.com provides a list of URLs of legitimate sites [1], and the Stanford WebBase

<sup>3</sup>PhishTank.com offers a community-based phishing verification system where users report suspected phishing sites and “vote” on whether those sites are actual phishing sites or not. At PhishTank.com, a web site categorized as an “online” site means that it is an active site; a web site categorized as a “valid” site means that it was collectively verified by the users as a phishing web site; a web site categorized as an “invalid” site means that it was determined not to be a phishing site.

project [16] has a crawler for grabbing web pages from a list of domains. We randomly selected 1323 legitimate sites from these two datasets.

### III. DOMAIN CREDIBILITY CHECK (DCC)

The primary function of the Domain Credibility Check (DCC) heuristic is to evaluate the effectiveness of the PageRank system in estimating the legitimacy of a target web site. Note that the DCC does not directly examine the credibility of a target web site since it may be a new web site that has not been archived in the PageRank system; its hosting domain, however, is more likely to have a longer history. Specifically, the DCC is carried out using a simple decision rule:

- If the domain has a PageRank score higher than or equal to  $\tau$ , the DCC determines that the domain hosting the web site is credible and consequently determines that the web site is also “legitimate”.
- If the domain’s PageRank score is lower than  $\tau$  or unavailable, the DCC determines that the web site’s hosting domain is not sufficiently credible and determines the legitimacy of the web site is “obscure”.

#### A. Discussion

From Figure 1(a), we observe that examining the PageRank scores of sites would help detect phishing sites because the hosting domains of most of the phishing sites have PageRank scores lower than the threshold  $\tau$ . In our experiments, the threshold  $\tau$  is set to a value no more than 5, since Dataset 1 suggests that the domains of most phishing sites have PageRank scores lower than 5. However, there are a small number of phishing sites that are hosted on credible domains, i.e., sites of type  $T_2$ . These phishing sites are erroneously determined by DCC as “legitimate”, which causes false negatives.

When running DCC on Dataset 2, we found that DCC has limitations in validating legitimate sites whose hosting domains have low PageRank scores. As can be seen in Figure 1(b), DCC determined that many legitimate sites are “obscure”, which equates to false positives. We can conclude that it is very difficult to distinguish legitimate sites on obscure and unknown domains (sites of type  $T_3$  or  $T_5$ ) from actual phishing sites (sites of type  $T_4$  or  $T_6$ ) by relying exclusively on a detection heuristic such as the DCC that utilizes the PageRank system. Therefore, we suggest integrating DCC with existing anti-phishing techniques that can validate legitimate sites, such as white listing and password management interface [17], [21].

### IV. CREDIBLE IN-NEIGHBOR SEARCH (CIS)

To reduce the false positive instances, we devised a heuristic called Credible In-neighbor Search (CIS); this heuristic examines the credibility of in-neighbor sites to the hosting domain. A web site usually has inlinks that associate in-neighbor sites with the current web site. We

have observed that the domains of legitimate sites have inlinks from credible web sites in the vast majority of the cases. In contrast, most phishing sites’ domains have no inlinks from credible web sites; in fact, some do not have any inlinks at all. The CIS heuristic determines that a web site is “legitimate” if its domain has inlinks from credible web sites; otherwise, the web site is determined to be “suspect”.

To implement the above technique, one problem needs to be addressed: acquiring a list of a web site’s in-neighbors. Yahoo! provides services for acquiring the in-neighbors of a web site. Specifically, they provide the “link:” command that can be issued through a web browser’s interface. This command returns a list of in-neighbors of a given web site, ordered according to the relevancy of those in-neighbors to the web site. For example, the command “link:www.example.com” returns a list of in-neighbors for “www.example.com”. Yahoo! site developer API [20] provides inlink data for supporting user applications. Using the Yahoo! site developer API, the CIS algorithm can readily obtain a list of in-neighbors of a given web site.

#### A. The CIS Algorithm

Let  $k(x) \in [0, 10]$  denote the PageRank score of a web site  $x$ . A web site,  $x$ , is considered to be *credible*, if  $k(x) \geq \tau$ , where  $0 < \tau \leq 10$ .

Let  $x_0$  denote the target web site’s domain, and let  $x_{i-1}$  for  $i \geq 1$  denote the “test site” in the  $i^{\text{th}}$  iteration of the CIS algorithm. In an attempt to determine the credibility of the target site’s hosting domain, the CIS algorithm goes through multiple iterations, where in each iteration, a credible in-neighbor search is performed for a site (or the domain  $x_0$ )—this is called a test site. In the first iteration,  $x_0$  is the test site. Below, we explain the steps of the CIS algorithm for each iteration.

In the  $i^{\text{th}}$  ( $i \geq 1$ ) iteration, the CIS algorithm collects a set of  $r$  in-neighbors of  $x_{i-1}$  whose PageRank scores are higher than or equal to  $k(x_{i-1})$ . This set is denoted as  $\mathbf{N}_i$ . The notation  $r$  is called the *search breadth*, and it represents the number of in-neighbors collected in each iteration. Note that  $\mathbf{N}_i$  may be an empty set, since  $x_{i-1}$  may have no in-neighbors with PageRank scores no less than  $k(x_{i-1})$  or  $x_{i-1}$  may have no in-neighbors at all. Thus,  $|\mathbf{N}_i| \in [0, r]$ . Figure 2 depicts an example of the CIS process carried out by the CIS algorithm. In this example,  $\tau = 5$  and  $r = 3$ . Note that the PageRank scores of the web sites in  $\mathbf{N}_1$  are no less than  $k(x_0)$ .

If  $\mathbf{N}_i \neq \emptyset$  and none of the sites in  $\mathbf{N}_i$  is credible, then the CIS algorithm selects a site that has the greatest PageRank score among the sites in  $\mathbf{N}_i$ . That is, the algorithm selects the test site for the next iteration,  $x_i$ , that satisfies the following:

$$x_i = \operatorname{argmax}_{y \in \mathbf{N}_i} \{k(y)\}. \quad (1)$$

The algorithm now starts the  $(i + 1)^{\text{th}}$  iteration and repeats the same steps followed in the  $i^{\text{th}}$  iteration, except the CIS process is carried out using  $x_i$  instead of  $x_{i-1}$ .

If one or more of the web sites in  $\mathbf{N}_i$  is credible, then the decision variable of the CIS algorithm,  $\mathbf{u}$ , is set to one and the algorithm stops. This signifies the CIS's determination that  $x_0$  is a legitimate but less credible (i.e., obscure) domain. In the example of Figure 2, the CIS algorithm stops in the 3<sup>rd</sup> iteration, since a credible web site  $E$  with  $k(E) = 5 \geq \tau$  is found. Note that there is a chain of links,  $((E, x_2), (x_2, x_1), (x_1, x_0))$ , linking the credible site  $E$  to the domain  $x_0$ , indicating that  $x_0$  is credible.

The number of iterations of the CIS algorithm is limited to  $d$ , where  $d$  is called the *maximum search depth*. If  $\mathbf{N}_i = \emptyset$  or no credible web site was found in  $\mathbf{N}_i$  after  $d$  iterations, then the CIS algorithm sets  $\mathbf{u}$  to zero and terminates. This signifies the CIS's determination that  $x_0$  is a suspect domain. A pseudo code of the CIS algorithm is given by *Algorithm 1*.

---

**Algorithm 1** Credible In-Neighbor Search (CIS)

---

**Input:**  $x_0, d, r$ , and  $\tau$ .

**Output:**  $\mathbf{u}$ .

```

1:  $\mathbf{u} \leftarrow 0, i \leftarrow 1$ 
2: while  $i \leq d$  do
3:   Determine  $\mathbf{N}_i$  using  $x_{i-1}$ 
4:   if  $\mathbf{N}_i = \emptyset$  then
5:     goto 13
6:   end if
7:   for each  $y \in \mathbf{N}_i$  do
8:     if  $k(y) \geq \tau$  then
9:        $\mathbf{u} \leftarrow 1, \text{goto } 13$ 
10:    end if
11:  end for
12:  Determine  $x_i, i \leftarrow (i + 1)$ 
13: end while

```

---

1) *Test Site Selection Rule:* Obviously, the execution time is heavily dependent on the number of CIS iterations that is executed. If one or more credible web sites are found in  $\mathbf{N}_i$ , the CIS algorithm stops at the  $i^{\text{th}}$  iteration. However, if no credible web site is found in  $\mathbf{N}_i$ , then the CIS algorithm selects the web site with the greatest PageRank score in  $\mathbf{N}_i$  as the test site for the next iteration (as specified by (1)). In the following discussions, we claim that such a test site selection rule maximizes the probability of executing a minimum number of algorithm iterations.

In [3], the authors showed that in a random web graph, the PageRank scores of web sites adheres to a specific distribution (e.g., a power-law distribution). Based on this result, we assume that the PageRank scores of web sites are independent and identically distributed (i.i.d.) random variables. With this assumption, we can prove that the following proposition is true.

**Proposition 1.** *Selecting the web site with the greatest PageRank score in  $\mathbf{N}_i$  for every iteration results in maximizing the probability of executing a minimum number of algorithm iterations.*

*Proof:* In the  $i^{\text{th}}$  iteration of the CIS algorithm ( $1 \leq i < d$ ), if no credible site is found in  $\mathbf{N}_i$ , the CIS algorithm selects a new test site,  $x_i$ , from  $\mathbf{N}_i$  and continues on to the  $(i + 1)^{\text{th}}$  iteration.

To facilitate our discussions, we will call the CIS algorithm's test site selection rule as *Rule 1*. Recall that the CIS algorithm always selects the site in  $\mathbf{N}_i$  with the greatest PageRank score as the test site—let  $y$  denote such a site.

Now suppose we define another test site selection rule, called *Rule 2*, in which one selects an arbitrary site in  $\mathbf{N}_i$  such that its PageRank score is lower than that of  $y$ —let  $z$  denote such a site. In other words, Rule 2 selects a site,  $z$ , such that  $k(z) < k(y)$ . Note that  $\tau > k(y) > k(z)$ .

Now let us compare the two selection rules.

- 1) If Rule 1 is adopted, the CIS algorithm selects  $y$  as the new test site, i.e.,  $x_i = y$ . Then the algorithm formulates  $\mathbf{N}_{i+1}$  by collecting the in-neighbors of  $y$  whose PageRank scores are no less than  $k(y)$ . Let  $a$  denote an arbitrary in-neighbor of  $y$  included in  $\mathbf{N}_{i+1}$ . Thus,  $k(a) \geq k(y)$ .
- 2) If Rule 2 is adopted, the CIS algorithm selects  $z$  as the new test site, i.e.,  $x_i = z$ . Then the algorithm formulates  $\mathbf{N}'_{i+1}$  by collecting the in-neighbors of  $z$  whose PageRank scores are no less than  $k(z)$ . Let  $b$  denote an arbitrary in-neighbor of  $z$  included in  $\mathbf{N}'_{i+1}$ . Thus,  $k(b) \geq k(z)$ .

In the  $i^{\text{th}}$  iteration, we can see that  $\tau > k(y) > k(z)$ ,  $k(a) \geq k(y)$ , and  $k(b) \geq k(z)$ . In the following discussions, we show that the probability that  $a$  is credible (when Rule 1 is used) is greater than the probability that  $b$  is credible (when Rule 2 is used).

The probability that  $a$  is credible is given by the following conditional probability:

$$Pr\{k(a) \geq \tau \mid k(a) \geq k(y)\} = \frac{Pr\{k(a) \geq \tau\}}{Pr\{k(a) \geq k(y)\}}.$$

On the other hand, the probability that  $b$  is credible is given by the following conditional probability:

$$Pr\{k(b) \geq \tau \mid k(b) \geq k(z)\} = \frac{Pr\{k(b) \geq \tau\}}{Pr\{k(b) \geq k(z)\}}.$$

Since  $k(a)$  and  $k(b)$  are i.i.d. random variables,

$$Pr\{k(a) \geq \tau\} = Pr\{k(b) \geq \tau\}.$$

Moreover, it is easy to see that

$$Pr\{k(a) \geq k(y)\} < Pr\{k(b) \geq k(z)\}$$

since  $k(y) > k(z)$ .

Thus, the following must be true:

$$Pr\{k(a) \geq \tau \mid k(a) \geq k(y)\} > Pr\{k(b) \geq \tau \mid k(b) \geq k(z)\}.$$

The above equation means that the probability that  $a$  is credible is greater than the probability that  $b$  is credible, which, in turn, means that the CIS algorithm is more likely

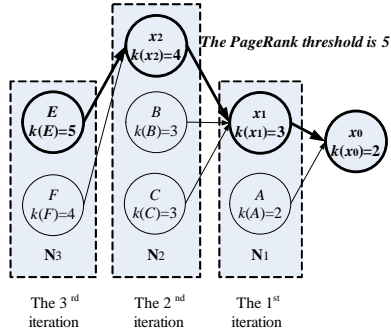
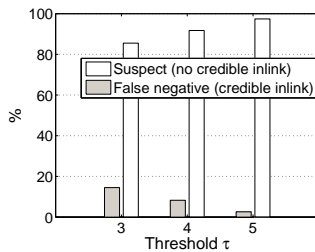
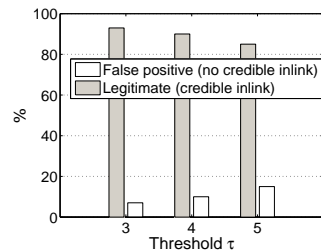


Figure 2. An illustration of the CIS algorithm.



(a) Results of phishing sites' domains;



(b) Results of legitimate sites' domains.

Figure 3. CIS results. CIS makes one of two decisions: “suspect” (no credible inlink found) and “legitimate” (credible inlink found).

to terminate in the  $(i+1)^{th}$  iteration if Rule 1 is used instead of Rule 2. From this true premise, we can infer that one can maximize the probability of executing the minimum number of CIS algorithm iterations if Rule 1 is adopted. ■

### B. Discussion

From Figure 3(b), we observe that the CIS heuristic can address the main drawback of the DCC heuristic which is the relatively high false positive rate. CIS, compared to DCC, can better distinguish legitimate sites and phishing sites hosted on obscure domains—i.e., better distinguish sites of type  $T_3$  from sites of type  $T_4$ .

Unfortunately, CIS reduces the number of false positives at the expense of moderately increasing the number of false negatives (Compare results in Figures 3(a) and 1(a)). The CIS failed to detect phishing sites that were hosted on domains with credible inlinks. For instance, it is possible for attackers to purchase domains with credible inlinks and then deploy their phishing sites on them in order to circumvent heuristics such as CIS. To counter this strategy, CIS can be combined with existing phishing detection techniques, such as the latest dynamic blacklists [5], [8], to detect this type of phishing sites. However, the CIS algorithm still has limitations in evaluating the legitimacy of sites hosted on unknown domains; in other words, CIS is unable to distinguish sites of type  $T_5$  from sites of type  $T_6$ .

## V. WEB CATEGORY COMPARISON (WCC)

Our study suggests that the vast majority of phishing sites hosted on less credible domains belong to web categories that are irrelevant to the domain's web category. Thus, checking whether a web site's web category matches its domain's web category can be used as a phishing detection heuristic.

### A. Web Category

To obtain the web category of a web site, queries are made to Yahoo! Directory [18]. When the keywords of a given web site are provided as inputs, Yahoo! Directory returns a list of web sites with web categories that are relevant to the input

keywords<sup>4</sup>. For example, given two query keywords “web” and “search”, the first web site returned by Yahoo! directory belongs to the following web categories: *Computers and Internet* > *Internet* > *World Wide Web* > *Searching the Web* > *Search Engines and Directories* > *Yahoo!*. The top web category in this example is *Computers and Internet*. This is also the top web category of the keywords “web” and “search”. For ease of implementation, we only takes into account the top web category of the target web site's keywords. The keywords of a web site can be obtained from online keyword extraction tools, such as the one provided by SEO (search engine optimizers) [14]. This tool accepts the web site URL as input and returns the top keywords of the web site ranked by frequency.

### B. Comparison

The WCC heuristic compares the top web categories of a target web site and its domain. It equates the web category of a web site to the web category of the site's keywords. The WCC obtains the top web category of the target site, denoted as  $C_1$ , and the top web category of the target site's hosting domain, denoted as  $C_2$ . By comparing  $C_1$  and  $C_2$ , the WCC algorithm makes a decision in the following way:

- It determines that the target web site has a web category “mismatch” with its hosting domain if  $C_1 \neq C_2$  or  $C_1$  or  $C_2$  is unavailable;
- It determines that the target web has a web category “match” with its hosting domain if  $C_1 = C_2$ .

### C. Discussion

Table II shows the false negative and false positive rates that were obtained by running WCC on the two datasets. The high false positive rate can be explained by the observation that the keywords extracted from a legitimate sites sometimes do not correlate with the web category of that site's hosting domain. Such instances of web category mismatch (between the site and its hosting domain) can happen to all types of legitimate sites (sites of type  $T_1$ ,  $T_3$  and  $T_5$ ).

<sup>4</sup>The top web category relevant to the keywords of a web site  $x$  (returned by Yahoo!) is considered as the web category of  $x$ .

Similar to DCC, we conclude that WCC cannot be used as a standalone scheme for phishing detection as it also generates a large numbers of false positives.

Table II shows that WCC incurred no false negatives when running on Dataset 1, but this result is somewhat misleading. It is possible for an attacker to circumvent WCC (i.e., cause false negatives) by embedding keywords that cause a phishing site’s web category to match its hosting domain’s web category. If those keywords cause the spoofed web site to look “less authentic”, then the attacker may employ well-known ploys such as hiding keywords—e.g., by matching the font color of the keywords with the site’s background color. On the other hand, the attacker may construct a phishing site that does not include any text and relies exclusively on images. This tactic would circumvent WCC and similar heuristics that need to extract text keywords from a site to analyze its content. To counter this tactic, a detection heuristic would need to analyze a site’s content using other methods, such as the existing techniques that recognize the images embedded on a site [4], [15].

Table II  
WCC RESULTS. WCC MAKES ONE OF TWO DECISIONS:  
“ILLEGITIMATE” (WEB CATEGORY MISMATCH) AND “LEGITIMATE”  
(WEB CATEGORY MATCH).

False negative rate	0 %
False positive rate	36 %

## VI. RELATED WORK

*Automated Phishing Detection Schemes:* SpoofGuard [4] applies multiple tests on the suspected page and combine these test results using a scoring mechanism, which yields a total spoof score for determining the spoofed page. CANTINA [23] is another content-based anti-phishing scheme using the TF-IDF (term frequency-inverse document frequency) algorithm and Google web search. Microsoft’s phishing filter [8] analyzes web sites that the user is browsing and determines if they have any characteristics that might be suspicious. It also checks the sites that a user is visiting against an up-to-the-hour, dynamic blacklist of reported phishing sites. The FireFox phishing protection tool [5] also employs a blacklist to check the history of suspicious web sites. The accuracy of such detection algorithms is somewhat limited by the updating frequency and coverage of the blacklist. A comprehensive evaluation on these anti-phishing tools can be found in [22].

*Anti-Phishing User Interfaces:* Anti-phishing user interfaces are typically implemented as browser plug-ins or toolbars. Web Wallet [17] is a browser extension designed to prevent a user from typing personal information directly into a web site. iTrustPage [13] is another anti-phishing interface scheme, which examines whether the web site that a user is trying to access has unverified  $\langle$ input $\rangle$  forms. PassPet [21] is a browser extension that makes it easier to log into known web sites, simply by pressing a single button.

PassPet requires people to memorize only one password, and generates a unique password for each site.

*Server Side Web Authentication Schemes:* A third type of anti-phishing schemes that require server side support. Bank of America’s SiteKey [2] is an example of a scheme that relies on site authentication image checking. This scheme works by having the bank users verify the SiteKey (site authentication image) before entering their password. Another example of a scheme based on site-authentication images is Yahoo!’s sign-in seal [19]. In this scheme, cookies and/or Macromedia Flash objects are placed on a user’s computer, and a Yahoo! server associates a site-authentication image with an individual computer. Protection against phishing is provided by enabling a user to verify the Yahoo! sign-in seal (image) before entering his/her “Yahoo! ID” and password.

## VII. CONCLUSIONS

In this paper, we evaluated three phishing detection heuristics that rely on the information provided by online resources (information depositories), viz the Google PageRank system, the Yahoo! Inlink data, and the Yahoo! directory service. Our study showed that those online resources are very useful in detecting phishing sites. The online resources are constantly evolving to cope with the continuous growth and metamorphosis of the Web. Note that as the online resources evolve to become more sophisticated and manipulation-resistant (e.g., resistant to link optimization), the detection capability of the proposed heuristics will also improve and evolve. However, our study also indicated that detection heuristics solely based on online resources are insufficient to adequately address the phishing attack problem. We have provided discussions on how each of the proposed heuristics can be complemented with other detection techniques to improve the overall detection performance.

## ACKNOWLEDGMENT

This work was partially sponsored by NSF through grant CNS-0524052.

## REFERENCES

- [1] 3Sharp, “Gone Phishing: Evaluating Anti-Phishing Tools for Windows”, September 2006. Available at: <http://www.3sharp.com/projects/antiphishing/gone-phishing.pdf>
- [2] Bank of America, “SiteKey: Online Banking Security”, November, 2006. Available at: <http://www.bankofamerica.com/privacy/sitekey/>
- [3] L. Becchetti and C. Castillo, “The Distribution of PageRank Follows A Power-Law Only for Particular Values of the Damping Factor”, in *Proc. of the 15th international conference on World Wide Web (WWW '06)*, May 2006, pp. 941–942.
- [4] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, “Client-Side Defense against Web-Based Identity Theft”, in *Proc. of the Network and Distributed System Security Symposium (NDSS '04)*, February 2004.
- [5] FireFox, “Phishing Protection”. Available at: <http://www.mozilla.com/en-US/firefox/phishing-protection/>

- [6] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A Framework for Detection and Measurement of Phishing Attacks", in *Proc. of the 2007 ACM Workshop on Recurring Malcode (WORM '07)*, November 2007, pp. 1–8.
- [7] T. McCall, "Gartner Survey Shows Phishing Attacks Escalated in 2007; More than \$3 Billion Lost to These Attacks". Available at: <http://www.gartner.com/it/page.jsp?id=565125>
- [8] Microsoft, "Microsoft Phishing Filter: A New Approach to Building Trust in E-Commerce Content", white paper, 2005.
- [9] T. Moore and R. Clayton, "Evaluating the Wisdom of Crowds in Assessing Phishing Websites", in *Proc. of the 12th International Financial Cryptography and Data Security Conference (FC '08)*, January 2008, pp. 16–30.
- [10] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", Stanford Digital Library Technologies Project, 1998.
- [11] Y. Pan and X. Ding, "Anomaly Based Web Phishing Page Detection", in *Proc. of the 22nd Annual Computer Security Applications Conference (ACSAC '06)*, December 2006, pp. 381–392.
- [12] PhishTank. Available at: <http://www.phishtank.com/>
- [13] T. Ronda and S. Saroiu, "iTrustPage: Pretty Good Phishing Protection", Technical Report, University of Toronto, December 2006.
- [14] Search engine optimizers (SEO), "Keyword Analysis Tool". Available at: <http://www.webmaster-toolkit.com/keyword-analysis-tool.shtml>
- [15] M. Sharifi and S.H. Siadati, "A phishing sites blacklist generator", in *Proc. of IEEE/ACS International Conference Computer Systems and Applications (AICCSA 2008)*, March 2008, pp. 840–843.
- [16] Stanford WebBase project. Available at: <http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/>
- [17] M. Wu, R. C. Miller and G. Little, "Web Wallet: Preventing Phishing Attacks by Revealing User Intentions", in *Proc. of Symposium on Usable Privacy and Security (SOUPS '06)*, July 2006, pp. 102–113.
- [18] Yahoo! Directory. Available at: <http://dir.yahoo.com/>
- [19] Yahoo! Inc., "Personalized Sign-In Seal", 2006. Available at: <https://protect.login.yahoo.com/>
- [20] Yahoo! Inc., Site Explorer Inbound Links API. Available at: <http://developer.yahoo.com/search/siteexplorer/V1/inlinkData.html>
- [21] K.-P. Yee, and K. Sitaker, "Passpet: Convenient Password Management and Phishing Protection", in *Proc. of Symposium on Usable Privacy and Security (SOUPS '06)*, July 2006, pp. 32–43.
- [22] Y. Zhang, S. Egelman, L. F. Cranor and J. I. Hong, "Phishing Phish: Evaluating Anti-Phishing Tools", in *Proc. of the 14th Annual Network & Distributed System Security Symposium (NDSS '07)*, 28th February–2nd March, 2007.
- [23] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A Content-Based Approach to Detecting Phishing Web Sites", in *Proc. of 16th International World Wide Web Conference (WWW '07)*, May 2007, pp. 639–648.