



# Antiphishing through Phishing Target Discovery

Phishing attacks are growing in both volume and sophistication. The antiphishing method described here collects webpages with either a direct or indirect association with a given suspicious webpage. This enables the discovery of a webpage's so-called "parasitic" community and then ultimately its phishing target — that is, the page with the strongest parasitic relationship to the suspicious webpage. Finding this target lets users determine whether the given webpage is a phishing page.

**Liu Wenyin, Gang Liu,  
Bite Qiu, and Xiaojun Quan**  
City University of Hong Kong

**P**hishing is the criminally fraudulent process of attempting to acquire sensitive information such as user names, passwords, and credit-card details from a victim by pretending to be a trustworthy entity in an electronic communication. Phishing attacks have seen alarming increases in both volume and sophistication. In the first half of 2011, at least 112,472 attacks occurred worldwide using 79,753 unique phishing domain names.<sup>1</sup> During 2008 and 2009, US consumers lost almost \$8 billion to cybercrime, according to the *Consumer Reports* "State of the Net" survey.<sup>2</sup>

In response to these threats, antiphishing researchers have developed various solutions. However, to compare a given suspicious webpage, researchers must first identify the legitimate webpage under attack — that is, the *phishing target*.<sup>3</sup> Unfortunately, this requirement isn't always easy to satisfy for general scenarios. Additionally, a few phishing attacks target less popular

or new webpages, in which case, even system administrators or experienced professionals have difficulty distinguishing between the phishing page and the target. The need to automatically discover a phishing target is an important problem for antiphishing efforts. If we can correctly identify a target, we can confirm which webpages are phishing pages. We can also inform the target owners of phishing attacks so that they can immediately take necessary countermeasures.

Here, we present an antiphishing approach that identifies phishing targets using a suspicious webpage's associated relationships.

## Method Overview

To convince visitors that a phishing page is legitimate, phishers typically use considerable information about targets to make their pages look as similar as possible to those targets. In this sense, phishing webpages aren't isolated but are often associated with their targets.

We refer to this as a *parasitic relationship*. Typically, a strong parasitic relationship exists from a phishing webpage to its target but not vice versa. One potentially useful parasitic relationship is hyperlinks from a phishing webpage to target websites. For example, one source reports that 42 percent of phishing kits contain links pointing to their target sites.<sup>4</sup> Our study on a dataset of 10,005 phishing pages collected from PhishTank (www.phishtank.com) shows that as many as 85.6 percent of such pages contain actual hyperlinks to their targets. However, sometimes the parasitic relationship from the given suspicious webpage isn't direct to its target; rather, we can mine it from a set of webpages – referred to as the *associated page set* – that are highly related to the target webpage. Using search engines, we can find these associated pages on the basis of the suspicious webpage's content.

For a given suspicious page, our method first finds all its directly or indirectly associated webpages. We then construct a Web graph from these associated pages by finding their associated webpages, and we further partition the graph to find a Web community (the *parasitic community*) for the given webpage. We finally use a parasitic coefficient to measure the parasitic relationship's strength from the given page to each page in the community. We regard the page with the strongest parasitic relationship to the given suspicious webpage as the phishing target. If we find such a target, we identify the given suspicious webpage as a phishing webpage. Otherwise, we consider it legitimate.

## Finding a Phishing Target

To detect a given webpage's phishing target, we first mine its associated webpages and build its parasitic community. We then measure the parasitic relationship from the given page to webpages in the community and identify the one with the strongest relationship as the phishing target.

## Constructing the Parasitic Community

A parasitic community is a tightly coupled Web community with stronger relationships inside the community than outside. In general, other webpages don't link to a phishing webpage. Thus, for a given webpage  $P$ , the parasitic community doesn't include  $P$  itself. We first build a Web graph based on the associated page set to find

$P$ 's parasitic community. To keep the webpages with which  $P$  has strong parasitic relationships in the parasitic community and remove those unrelated to  $P$ , we then employ a maximum-flow, minimum-cut method to partition this Web graph and extract an initial parasitic community. (Among the well-known methods for constructing Web communities,<sup>5,6</sup> we followed the s-t minimum-cut-based method<sup>5</sup> as our basic framework.) We then refine this initial community via reinforcement iterations.

Given a webpage  $P$ , we construct the parasitic community as follows. Step 1 is to find all pages that are either directly or indirectly associated with page  $P$  (see Figure 1). All pages found in step 1 constitute  $P$ 's initial associated page set,  $A$ . Once we determine  $A$ , a crawler builds a directed Web graph,  $G$ , of  $A$  (step 2). In step 3, we further partition the graph  $G$  to construct  $P$ 's parasitic community ( $C$ ). Finally, step 4 is to obtain the interrelationships among members in  $C$  to reinforce the associated page set  $A$ . The reinforcement step updates  $A$  by adding active pages and removing inactive ones. Steps 2 through 4 iterate until certain conditions are satisfied; we then output the  $C$  constructed in the final iteration. As Figure 1 shows, the iteration ends if  $P$  is found in  $G$ , if the number of iterations exceeds a preset threshold, or if  $A$  doesn't change after reinforcement.

**Finding the initial associated page set.** The associated webpages in the set come from two main sources:

- *Directly associated pages* are those webpages that  $P$  links to directly. We can find them by examining  $P$ 's HTML source and extracting all hyperlinks in it.
- *Indirectly associated pages* share the same or similar text or visual information with  $P$ . We can mine such indirectly associated pages by, for example, searching the Web using the representative keywords found in  $P$  as queries. Representative keywords are those in  $P$ 's title, body tags, and other meta tags.

In this step, we assume the simple hypothesis  $P \notin A$ . That is, if  $P$  is directly associated with itself, it will not be included in  $A$ . In addition, if we find that  $P$  is indirectly associated with itself (for example, found in the search results), it also won't be included in  $A$ .

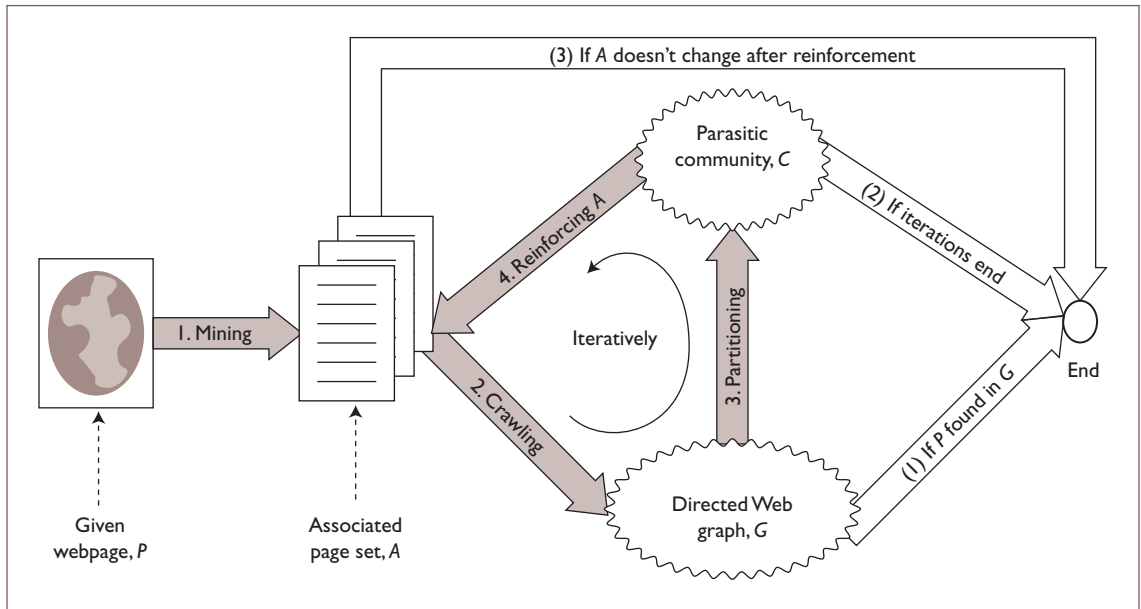


Figure 1. Constructing a parasitic community. All pages found in step 1 constitute  $P$ 's initial associated page set. Steps 2 through 4 iterate until certain conditions are satisfied.

**Building a Web graph.** To expand the initial associated page set to capture more related webpages, we build a Web graph  $G = (V, E)$  via two steps of crawling with different settings. Initially, we assign the associated page set  $A$  to  $V$  as the initial vertex set, with  $E$  as an empty set. The first crawling step finds those pages directly linked with  $A$  through both forward links and back links because we want to keep the important webpages in the associated page set in the first iteration. We add the set of newly found pages,  $N$ , and the set of links,  $L$ , to  $V$  and  $E$ , respectively. This step expands the graph by  $G = (A + N, L)$ . The second step starts from the newly found pages  $N$  and crawls through only  $N$ 's forward links because the pages found in this round are less related to set  $A$ . Incorporating with the newly found pages ( $N^*$ ) and the newly found links ( $L^*$ ) in the second step, we obtain the final graph,  $G = (A + N + N^*, L + L^*)$ .

In reality, we might find  $P$  in the newly found pages during the crawling steps – that is,  $P \in (N \cup N^*)$ . If this condition holds true, we can stop and claim that  $P$ 's parasitic community is empty and  $P$  is a legitimate page.

**Partitioning the Web graph.** Inevitably, the Web graph can contain certain unrelated pages. To purify it, we follow the s-t minimum-cut technique for partitioning a Web graph.<sup>5</sup> This method generates a manageable Web community, usually fewer than 100 webpages. This is

critical to our algorithm because a given phishing page usually has a limited number of potential targets. In addition, a large parasitic community would cause a high outlier rate.

To apply the s-t minimum-cut algorithm to partition the Web graph  $G = (A + N + N^*, L + L^*)$ , we must create a source  $s$  and a sink  $t$ , and add them to the Web graph by linking  $s$  to all pages in  $A$  and linking all pages in  $N^*$  to  $t$  with infinite capacity  $c$  for each new edge. Let  $L_s = \{e = (s, v), c(s, v) = \infty \mid v \in A\}$  and  $L_t = \{e = (u, t), c(u, t) = \infty \mid u \in N^*\}$ . After creating the source vertex and the sink vertex, we obtain the updated  $G = (s + t + A + N + N^*, L_s + L_t + L + L^*)$ .

In Web graph  $G$ , each edge has a capacity constraint, which we calculate on the basis of the density of the links between the two pages. Because all edges from  $s$  or to  $t$  have infinite capacities, both  $s\#$  (the total capacity of edges from  $s$ ) and  $t\#$  (the total capacity of edges to  $t$ ) exceed the cut set size. Hence, we can cut  $G$  to identify the Web community according to the theorem that Gary Flake and his colleagues proved.<sup>5</sup> After the cut, all vertices that are reachable from  $s$  constitute the community. In our approach, once we add the virtual source  $s$  and the virtual sink  $t$  to the Web graph, we apply the *shortest augment path* algorithm<sup>7</sup> to calculate the maximum flow from  $s$  to  $t$ . Construction of the parasitic community is complete when we collect all vertices in  $G$  that are reachable from  $s$ .

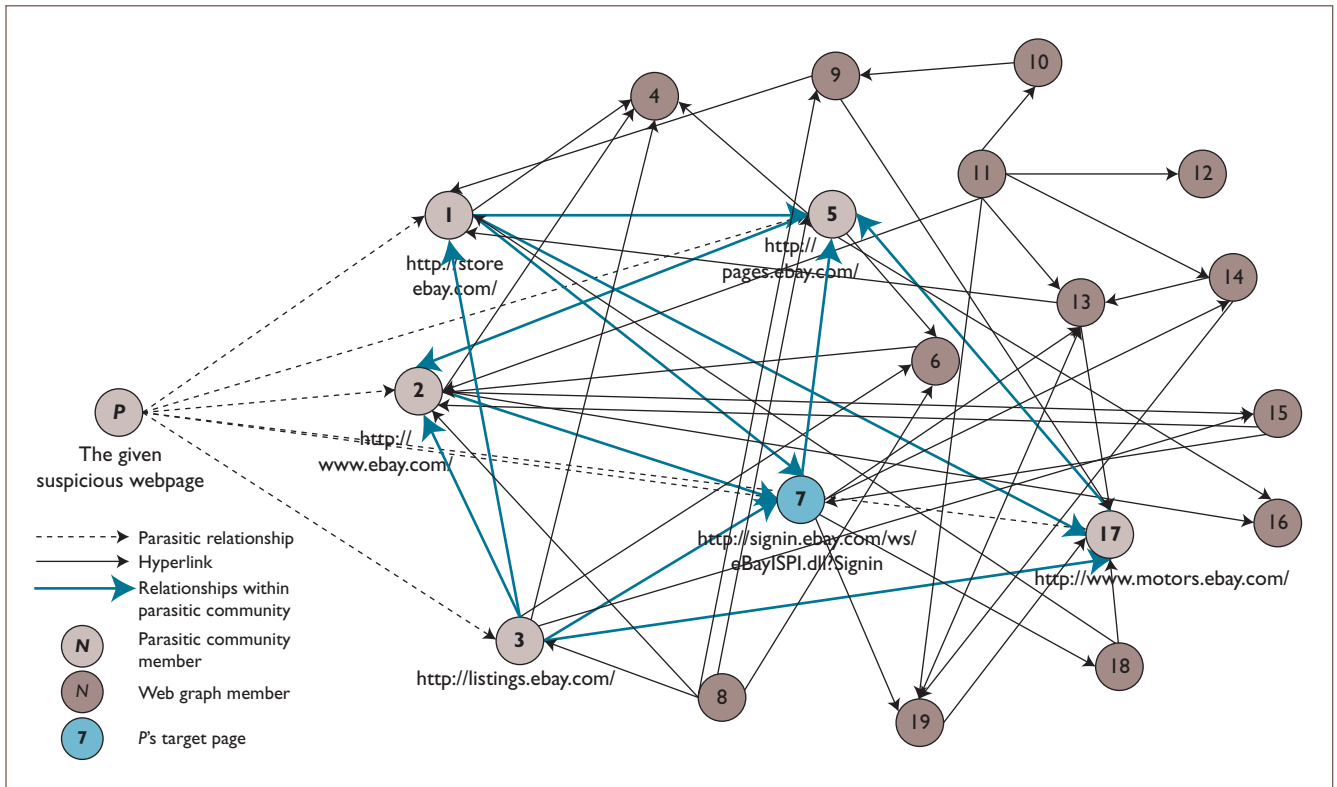


Figure 2. The parasitic community of a phishing webpage  $P$  targeting eBay. Each node represents a webpage. Numbered nodes are  $P$ 's associated webpages, and nodes linked via thick blue lines constitute  $P$ 's parasitic community.

**Reinforcing the associated page set.** To incorporate more associated webpages into the associated page set, we take a reinforcement step in the current iteration, adding active webpages and removing inactive ones. An active page is one that belongs to community  $C$  but doesn't belong to the associated page set  $A$ , and that has either the highest in-degree relative to  $C$  or the highest out-degree relative to  $C$ . A node's in-degree relative to  $C$  is the number of edges pointing to the node from all nodes in  $C$ , and its out-degree relative to  $C$  is the number of edges pointing from the node to all nodes in  $C$ .

An inactive page is one that belongs to the associated page set  $A$  but doesn't belong to community  $C$  and the sum of whose in-degree and out-degree relative to  $C$  is less than a threshold (set to 2 for this article's purposes). By adding the active pages to the associated page set and removing the inactive pages from it, we can reinforce the associated page set gradually.

**A parasitic community example.** Figure 2 shows a parasitic community built using our method. We use a phishing page targeting eBay as the

given page,  $P$ , which looks visually identical to eBay's sign-in page. Each node represents a webpage. The nodes with numbers are  $P$ 's associated webpages. Those nodes linked by thick blue lines constitute  $P$ 's parasitic community. After constructing the parasitic community, we develop a parasitic-coefficient calculation method to reveal the strength of the parasitic relations between  $P$  and its community. In the figure, the obtained parasitic community consists of six webpages; we can consider the one with the highest parasitic coefficient to be  $P$ 's target webpage (labeled "7").

### Phishing Target Discovery

We propose a parasitic coefficient for detecting a phishing target within a parasitic community on the basis of the parasitic relationship between the given page and the webpages in the community. We create an extensible framework to incorporate the different metrics of an association relationship to calculate the parasitic coefficient.

**Metrics for a direct association relationship.** As a kind of direct association relationship, forward

links directly imply reference relationships from the source to the target. Phishers frequently use such relationships with legitimate webpages to obtain victims' trust. However, it's impossible for legitimate webpages to link back to phishing webpages. Thus, forward links are considered a very important factor in measuring a parasitic relationship. We define  $L_{ij}$  as the metrics for the forward-link relationship from  $page_i$  to  $page_j$ :

$$L_{ij} = \frac{NL_{ij}}{NL_i}, \quad (1)$$

where  $NL_{ij}$  is the number of forward links from  $page_i$  to any page in  $page_j$ 's website,  $NL_i$  is the total number of links from  $page_i$ , and  $L_{ij}$  is just one of a few kinds of direct association relationships that might exist between webpages. If we can mine  $K$  kinds of direct association relationships, we use a weighted average of their metrics as the overall metric for all direct association relationships  $D_{ij}$ , as follows:

$$D_{ij} = \sum_{k=1}^K (w_k \cdot D_{ij}^{(k)}), \quad (2)$$

where  $D_{ij}^{(k)}$  is the value of the direct association relationship  $k$ , and  $w_k$  is its weight. In this article, we consider only forward links in the direct association relationship, so  $D_{ij} = L_{ij}$ .

#### Metrics for an indirect association relationship.

Indirect association relationships between webpages don't behave like the forward-link relationship, which we can directly measure using the number of links. Measuring indirect association relationships is less intuitive. In this article, we consider only ranking association relationships and similarity association relationships. However, our framework can easily incorporate other indirect association relationships, as the following equation demonstrates:

$$I_{ij} = \sum_{k=1}^K (w_k \cdot I_{ij}^{(k)}), \quad (3)$$

where  $I_{ij}$  is the overall metric for all indirect association relationships,  $I_{ij}^{(k)}$  is the value of the indirect association relationship  $k$ , and  $w_k$  is its weight. To mine indirect association relationships, we extract title information, keywords between the given webpage's body tags, and information in other meta tags to represent the page content.

*The ranking association relationship.* We define the ranking association relationship from  $page_i$  to  $page_j$  on the basis of  $page_j$ 's rank in the search results using  $page_i$ 's content as the query. When we query a powerful search engine with  $page_i$ 's content, if  $page_j$ 's domain name matches the domain name of any item in the top  $N$  search results, then a ranking association relationship exists from  $page_i$  to  $page_j$ . Our technique assumes that legitimate webpages constitute the majority of the search engine results and that a legitimate webpage gets a higher rank than a phishing page. Thus, the ranking association relationship from a phishing webpage to a legitimate webpage is quite different from the reverse. The ranking relationship is asymmetric, and we measure it as follows:

$$R_{ij} = \sum_{k=1}^K (w_k \cdot R_{ij}^{(k)}), \quad (4)$$

where  $R_{ij}$  is the overall ranking association relationship from  $page_i$  to  $page_j$ ,  $w_k$  is the weight of  $R_{ij}^{(k)}$ , and  $R_{ij}^{(k)}$  is the ranking association relationship from  $page_i$  to  $page_j$  using the keywords extracted from the source  $k$  in  $page_i$  as the query. The sources of query keywords include  $page_i$ 's title, meta tags, and body. We can calculate the value of  $R_{ij}^{(k)}$  as follows:

$$R_{ij}^{(k)} = \frac{N_r - (R_s - 1)}{N_r}, \quad (5)$$

where  $N_r$  is the total number of search results we use (for example, the top 50 results), and  $R_s$  is the rank of  $page_j$ 's domain name in the results.

*The similarity association relationship.* Phishing webpages usually use the same or similar text and visual content as true webpages to deceive their visitors. If a given webpage is very similar to a webpage in its parasitic community, the probability that it's a phishing page is high. We measure the similarity association relationship from  $page_i$  to  $page_j$  as follows:

$$S_{ij} = \sum_{k=1}^K (w_k \cdot S_{ij}^{(k)}), \quad (6)$$

where  $S_{ij}$  is the overall similarity association relationship from  $page_i$  to  $page_j$ ,  $w_k$  is the weight of  $S_{ij}^{(k)}$ , and  $S_{ij}^{(k)}$  is the similarity association relationship from  $page_i$  to  $page_j$  using the features extracted from source  $k$  in the webpages. These kinds of features include text content, visual content, and the webpage's layout.



In this article, we take into account only the features of the two webpages' text content.

To measure the asymmetric content-based similarity from  $page_i$  to  $page_j$ , which is more compatible with human perception, we employ a model that measures the similarity between objects in terms of their common and distinctive features.<sup>8</sup> We simplify and normalize this model to  $[0, 1]$ , as the following formula shows:

$$S_{ij}^{(k)} = \frac{|T_i(k) \cap T_j(k)|}{|T_i(k)|}, \quad (7)$$

where  $T_i(k)$  is the set of words extracted from source  $k$  in  $page_i$ ,  $|T_i(k) \cap T_j(k)|$  is the number of common words they share, and  $|T_i(k)|$  is the number of words in  $T_i(k)$ .

**Parasitic coefficient.** Finally, we calculate the parasitic coefficient  $Para_{ij}$  of  $page_i$  to  $page_j$ , considering the weighted sum of the direct and indirect association relationships, as follows:

$$Para_{ij} = \alpha D_{ij} + (1 - \alpha) I_{ij}, \quad (8)$$

where  $D_{ij}$  is the direct association relationship from  $page_i$  to  $page_j$ ,  $I_{ij}$  is the indirect association relationship from  $page_i$  to  $page_j$ , and  $\alpha$  is a control parameter.

On the basis of  $Para_{ij}$ , we define the parasitic coefficient ratio  $PCR_{ij}$  as follows:

$$PCR_{ij} = \frac{Para_{ij}}{Para_{ji}}. \quad (9)$$

Assume that  $page_i$  is the given webpage, and  $page_j$  is a page in the parasitic community. If  $Para_{ij}$  is smaller than a threshold (0.1 in this article), then  $page_i$  has almost no association relationship with  $page_j$ . After a careful study of the experimental results, we assume that  $PCR_{ij} = 0$  in such a case. The higher  $PCR_{ij}$ 's value, the stronger the parasitic relationship is from  $page_i$  to  $page_j$ . Hence, we conclude that the website of  $page_j$  with the highest  $PCR_{ij}$  (where, for this article's purposes,  $PCR_{ij} > 2$  should hold true) is the phishing target.

## Implementation and Evaluation

We implemented our method as a service at [www.SiteWatcher.cn](http://www.SiteWatcher.cn). Any end user can test a suspicious webpage by submitting its URL as the given webpage. We also posted our latest phishing reports to our Twitter (<http://twitter>.

[com/sitewatcherhk](http://com/sitewatcherhk)) and Sina Weibo (<http://t.sina.com.cn/sitewatcher>) pages to draw users' attention. We integrated our solution in Microsoft Outlook as a plug-in email filter ([www.sitewatcher.cn/download/emailwatcher.rar](http://www.sitewatcher.cn/download/emailwatcher.rar)) that can verify all embedded URLs in an email at the time of receipt. Similarly, we can integrate this solution into Web browsers as a lightweight plug-in to alert users to phishing attacks. We also implemented a class library (with APIs) for enterprise users to build their own application systems so as to check suspicious websites.

## Detection Accuracy

To test our method's detection accuracy, we collected 10,005 phishing pages that target 154 well-known companies or brands as our phishing dataset. We collected these phishing URLs from PhishTank using a specialized script with succedent manual verification to eliminate biased data, and classification to establish benchmarks. In our experiment, we set  $\alpha$  from Equation 8 to 0.5, and the threshold of  $PCR_{ij}$  in Equation 9 as 2. We determined these thresholds empirically through large-scale experiments to achieve the best performance for phishing target identification. We use a strict criterion for target evaluation: the target confirmed as correct must be the one with the highest parasitic coefficient ratio (to the given webpage) among all members in the parasitic community. Otherwise, we consider the result incorrect. Thus, phishing detection accuracy (the accuracy of detecting a webpage as a phishing page) is 99.2 percent, and the phishing target identification accuracy (the accuracy of correctly finding the phishing target) is 92.1 percent.

## False-Alarm Rate

To test our method's false-alarm rate, we conducted an experiment on 1,000 legitimate webpages obtained from the Random Yahoo Link (<http://random.yahoo.com/bin/ryl>). We repeated the experiment 10 times, and the false-alarm rates ranged from 1.6 percent to 2.4 percent, with an average of 2.2 percent. This rate is acceptable for most applications because human administrators can verify false alarms. In many applications, detection accuracy is more critical because missing a phishing website during detection could cause significant losses. However, applications that are less tolerant to false alarms might desire a better rate.

Table 1. PageRank and back-link effects on false-negative and false-positive rates.\*

PageRank (PR) rating	Number of back-links (BL) (%)	BL ≥ 1 (%)	BL ≥ 3 (%)	BL ≥ 5 (%)	BL ≥ 7 (%)	BL ≥ 10 (%)	BL ≥ 15 (%)
N/A	0.8, 2.2	2.4, 0.08	6.7, 0.1	5.2, 0.13	4.6, 0.16	4, 0.21	3.6, 0.33
PR ≥ 1	2.5, 0.44	2.4, 0.49	2.4, 0.49	2.2, 0.5	2.1, 0.52	2, 0.53	1.9, 0.59
PR ≥ 2	1.9, 0.54	1.9, 0.59	1.8, 0.59	1.74, 0.6	1.66, 0.6	1.6, 0.62	1.5, 0.67
PR ≥ 3	1.5, 0.77	1.5, 0.84	1.4, 0.84	1.4, 0.85	1.4, 0.85	1.3, 0.86	<b>1.27, 0.9</b>
PR ≥ 4	<b>1.05, 1.2</b>	1, 1.31	1, 1.31	1, 1.31	1, 1.32	1, 1.32	1, 1.33
PR ≥ 5	0.87, 1.6	0.88, 1.8	0.88, 1.8	0.88, 1.8	0.88, 1.8	0.88, 1.8	0.88, 1.8

\*Promising combinations are in bold font.

To make our method more adaptable to various application scenarios, we adopted a way of combining two webpage metrics – Google PageRank and the back-link count – as a pre-filtering strategy to make predictive decisions about whether a given page is legitimate. Table 1 shows the false-negative and false-positive rates under different combinations of PageRank (PR) and back-link count (BL). The table doesn't display combinations with PR > 5 because they were monotonous in their results. We found two promising combinations. First, with only PR ≥ 4, the false-negative rate and false-positive (false-alarm) rate are 1.05 percent (corresponding to a detection accuracy of 98.95 percent) and 1.2 percent, respectively. Second, with PR ≥ 3 and BL ≥ 15, the false-negative rate and false-positive rate are 1.27 percent (corresponding to a detection accuracy of 98.73 percent) and 0.9 percent, respectively. We discovered that the prefiltering strategy remains robust because it's difficult for short-lived phishing pages to possess a high PageRank or have many back-links.

Comparison with Other Solutions

We also compared our method with other anti-phishing solutions in terms of both functionality and accuracy. These compared solutions include the black/white list method, WOT (www.mywot.com), a method based on the Semantic Link Network,<sup>9</sup> Ma's method,<sup>10</sup> Cantina,<sup>11</sup> and Xiang's method<sup>12</sup> (see the "Related Work in Antiphishing" sidebar for more about these approaches). The detailed description of every comparison raises the following questions: Does it identify a phishing webpage manually or automatically? If a given webpage is a phishing page, can it discover that page's target? What is the phishing page detection accuracy? And what is the legitimated page detection accuracy?

As Table 2 shows, our method has more advantages over other solutions. According to experimental results, our method has achieved both higher accuracy and a lower false-alarm rate on large datasets.

We also tested the effectiveness and efficiency of a few commercial antiphishing solutions. In that study,<sup>13</sup> PhishTank identified 89.2 percent of phishing websites with an average of 16.4 hours for each site, Google's Safe Browsing API identified 65.7 percent in 10.1 hours on average, and Microsoft's IE browser identified 40.4 percent in 24.5 hours on average. Because most victims of a phishing scam fall for it within eight hours from when the attack begins, the time these solutions require to verify the attack demonstrates that they don't effectively protect users.

As demonstrated, our proposed method can identify phishing pages effectively (by more than 98 percent) and efficiently (in a few minutes). We can also discover phishing targets. Potential applications include phishing detection for Google Adwords (or other companies' similar functions). If the advertising link is detected as phishing, and its target is found, the target owner can be notified and take necessary action. We're currently trying to further speed up our method. If we can report phishing within a few seconds, end users can be warned before submitting any private information to a phishing website. □

Acknowledgments

Our work was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (project number 142210) and the Natural Science Foundation of China (under grant number 91024012).

Table 2. Comparison among antiphishing methods.

Antiphishing methods	Manual or automatic identification	Phishing target discovery	Phishing page detection		Legitimated page detection	
			Accuracy rate (%)	Size of dataset (pages)	False-alarm rate (%)	Size of dataset (pages)
Our method (with prefilter)	Automatic	Yes	98.73–98.95	10,005	0.9–1.2	1,000 × 10 (random)
Black/white list	Manual	No	N/A	N/A	N/A	N/A
WOT	Manual	No	N/A	N/A	N/A	N/A
Semantic Link Network method	Automatic	Yes	83.40	1,000	15.90	1,000
Ma's method	Automatic	No	95–99	20,500	0.8	15,000
Cantina	Automatic	No	90	100	1	100
Xiang's method	Automatic	No	90.06	7,906	1.95	3,543

## Related Work in Antiphishing

Recent antiphishing work falls into several categories. *Black/white lists* are the most straightforward. Representative systems based on black/white lists include PhishTank Site-Checker (<https://addons.mozilla.org/en-us/firefox/addon/phishtank-sitechecker/>), Google Safe Browsing ([www.google.com/tools/firefox/safebrowsing/index.html](http://www.google.com/tools/firefox/safebrowsing/index.html)), FirePhish (<https://addons.mozilla.org/en-US/firefox/addon/firephish-anti-phishing-extends/>), and CallingID Link Advisor ([www.callingid.com/Default.aspx](http://www.callingid.com/Default.aspx)). A white list is a list of reputable companies and their formal domain names. Domain name registration is highly dynamic and frequently updated. Thus, maintaining this list is labor-intensive and difficult. In addition, the method has its limits and could fail to detect a webpage whose domain name isn't on the white list. A blacklist, on the other hand, provides a list of known phishing websites. However, it is certainly vulnerable to a phishing attack during the time window between when the phishing site is launched and when it's added to the blacklist.

Liu Wenjin and his colleagues proposed a *visual-similarity-based strategy* for detecting phishing webpages.<sup>1</sup> They first asked users or system administrators to register with their system the true webpages that those users wanted to protect. They then employed a few algorithms to compute visual similarity, including block similarity, layout similarity, and overall style similarity between a suspicious webpage and a protected one. A webpage could be reported as a phishing suspect if the visual similarity was higher than a preset threshold.

Yue Zhang and his colleagues implemented a *content-based approach* to detecting phishing websites on the basis of the term frequency-inverse document frequency (TF-IDF) information retrieval algorithm.<sup>2</sup> The authors first calculated the TF-IDF score of each term in a given webpage and generated a lexical signature by taking the five terms with the highest scores. They then input those lexical signatures into Google

to get the top  $N$  search results. If the given webpage's domain name didn't fall into the search results, they classified it as a phishing website.

Guang Xiang and Jason Hong developed a hybrid phishing detection approach that uses *identity discovery* and *keyword retrieval*.<sup>3</sup> They first used an identity-based component to acquire a webpage's identity and then employed search engines using the identity as queries to detect phishing. If the given webpage's domain name wasn't listed in the search results, the authors identified it as a phishing page.

Wenjin and his colleagues used the *Semantic Link Network (SLN)* to automatically identify a given webpage's phishing target.<sup>4</sup> They first found the given webpage's associated pages and then constructed an SLN from those webpages. They exploited a mechanism of reasoning on the SLN to identify whether the given webpage was a phishing page, and discovered its target if so. Although this method can detect a phishing webpage and find its target as well, the computational cost of building an SLN is expensive, and the false positive rate of such a method is somewhat higher than that of our approach.

## References

1. W. Liu et al., "An Antiphishing Strategy Based on Visual Similarity Assessment," *IEEE Internet Computing*, vol. 10, no. 2, 2006, pp. 58–65.
2. Y. Zhang, J.I. Hong, and L.F. Cranor, "Cantina: A Content-Based Approach to Detecting Phishing Websites," *Proc. 16th Int'l World Wide Web Conf. (WWW 07)*, ACM Press, 2007, pp. 639–648.
3. G. Xiang and J.I. Hong, "A Hybrid Phish Detection Approach by Identity Discovery and Keywords Retrieval," *Proc. 18th Int'l World Wide Web Conf. (WWW 09)*, ACM Press, 2009, pp. 571–580.
4. W. Liu et al., "Discovering Phishing Target Based on Semantic Link Network," *Future Generation Computer Systems*, vol. 26, no. 3, 2010, pp. 381–388.



### References

1. *Global Phishing Survey: Trends and Domain Name Use in 1H2011*, tech. report, Antiphishing Working Group, 2011; [www.antiphishing.org/reports/APWG\\_GlobalPhishingSurvey\\_1H2011.pdf](http://www.antiphishing.org/reports/APWG_GlobalPhishingSurvey_1H2011.pdf).
2. "Boom Time for Cybercrime," *Consumer Reports*, June 2009; [www.consumerreports.org/cro/magazine-archive/june-2009/electronics-computers/state-of-the-net/overview/state-of-the-net-ov.htm](http://www.consumerreports.org/cro/magazine-archive/june-2009/electronics-computers/state-of-the-net/overview/state-of-the-net-ov.htm).
3. W. Liu et al., "An Antiphishing Strategy Based on Visual Similarity Assessment," *IEEE Internet Computing*, vol. 10, no. 2, 2006, pp. 58–65.
4. M. Cova, C. Kruegel, and G. Vigna, "There is No Free Phish: An Analysis of 'Free' and Live Phishing Kits," *Proc. 2nd Usenix Workshop Offensive Technologies (WOOT 08)*, Usenix Assoc., 2008, pp. 1–8.
5. G.W. Flake, S. Lawrence, and C.L. Giles, "Efficient Identification of Web Communities," *Proc. 6th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, 2000, pp. 150–160.
6. D. Gibson, J. Kleinberg, and P. Raghavan, "Inferring Web Communities from Link Topology," *Proc. 9th ACM Conf. Hypertext and Hypermedia*, ACM Press, 1998, pp. 225–234.
7. T.H. Cormen et al., *Introduction to Algorithms*, 2nd ed., MIT Press, 2001.
8. A. Tversky, "Features of Similarity," *Psychological Rev.*, vol. 84, no. 4, 1977, pp. 327–352.
9. W. Liu et al., "Discovering Phishing Target Based on Semantic Link Network," *Future Generation Computer Systems*, vol. 26, no. 3, 2010, pp. 381–388.
10. J. Ma et al., "Beyond Blacklists: Learning to Detect Malicious Websites from Suspicious URLs," *Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, 2009, pp. 1245–1254.
11. Y. Zhang, J.I. Hong, and L.F. Cranor, "Cantina: A Content-Based Approach to Detecting Phishing Websites," *Proc. 16th Int'l World Wide Web Conf. (WWW 07)*, ACM Press, 2007, pp. 639–648.
12. G. Xiang and J.I. Hong, "A Hybrid Phish Detection Approach by Identity Discovery and Keywords Retrieval," *Proc. 18th Int'l World Wide Web Conf. (WWW 09)*, ACM Press, 2009, pp. 571–580.
13. G. Liu et al., "Smartening the Crowds: Computational Techniques for Improving Human Verification to Fight Phishing Scams," *Proc. 7th Symp. Usable Privacy and Security (SOUPS 11)*, ACM Press, 2011.

**Liu Wenyin** is an assistant professor at the City University of Hong Kong. His research interests include antiphishing, question answering, graphics recognition, and performance evaluation. Wenyin has a DSc from the Technion, Israel Institute of Technology, Haifa. He's a fellow of the International Association for Pattern Recognition and a senior member of IEEE. Contact him at [liuwenyin@gmail.com](mailto:liuwenyin@gmail.com).

**Gang Liu** is a postdoctoral fellow in the Department of Chinese, Translation, and Linguistics at the City University of Hong Kong. His research interests include artificial intelligence approaches to computer security and privacy, poetics of translation, Web document analysis, information retrieval, and natural language processing. Liu has a PhD in computer science from the City University of Hong Kong. Contact him at [gangliu@cityu.edu.hk](mailto:gangliu@cityu.edu.hk).


**Bite Qiu** is pursuing a PhD in the Department of Computer Science at the City University of Hong Kong. His research interests include antiphishing, information retrieval, and Web data mining. Qiu has a BE in software engineering from Tongji University, Shanghai. Contact him at [biteqiu@student.cityu.edu.hk](mailto:biteqiu@student.cityu.edu.hk).

**Xiaojun Quan** is pursuing a PhD in the Department of Computer Science at the City University of Hong Kong. His research interests include data mining, information retrieval, question answering, and antiphishing. Quan has an ME in computer science from the University of Science and Technology of China. Contact him at [xiaoquan@student.cityu.edu.hk](mailto:xiaoquan@student.cityu.edu.hk).



stay connected.  
IEEE Computer Society

	@ComputerSociety   @ComputingNow
	facebook.com/IEEE ComputerSociety   facebook.com/ComputingNow
	IEEE Computer Society   Computing Now
	youtube.com/ieeecompetersociety

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

## IEEE MultiMedia: Call for Papers

# Web-Scale Near-Duplicate Search: Techniques and Applications

### Submissions due: 29 June 2012

As bandwidth accessible to average users is increasing, audiovisual material has become the fastest growing data type on the Internet. The impressive growth of the social Web where users can exchange user-generated content contributes to the overwhelming number of multimedia files available. Among these huge volumes of data, there exist large numbers of near-duplicates and copies. File copies are easy to detect using hashes. Near-duplicates are based on the same original content, but have been edited and postprocessed, resulting in different files. Another type of near-duplicate relates to footage about the same event or scene. The detection of near-duplicates poses a challenge to multimedia content analysis, especially when speed, scale, and copied fragment length are pushed to operational levels. Near-duplicates carry both informative and redundant signals, for example, providing rich visual clues for indexing and summarizing videos from different sources. On the other hand, the excessive amount of near-duplicates makes browsing Web videos streamed over Internet an extremely time-consuming task. As a result, there is strong interest from industry, academia, and governmental agencies in Web-scale search, elimination, detection, and use of near-duplicates for various multimedia applications.

This special issue seeks innovative contributions dedicated to the theme of Web-scale near-duplicate search. Topics of interests include, but are not limited to, the following:

#### Techniques and algorithms

- near-duplicate and/or partial near-duplicate detection;
- cross-media search of near-duplicates;
- semantic-based detection of near-duplicates;
- framework and algorithm for real-time, near-duplicate detection;
- semantic indexing and hashing techniques;
- similarity and perception learning;
- multimedia fingerprint extraction; and
- instance search, matching, and localization.

#### Applications

- search results ranking and diversification;
- novelty detection;
- topic detection, tracking, and threading;
- data-driven applications;

- Internet media management and service;
- Web-scale multimedia mining;
- Web-scale summarization and browsing of multimedia data; and
- multimedia archaeology mining.

### Submission Procedures and Deadlines

Submit your paper at <https://mc.manuscriptcentral.com/cs-ieee>. When uploading your paper, please select the appropriate special issue title under the category "Manuscript Type." If you have any questions regarding the submission system, please contact Andy Morton at [mm-ma@computer.org](mailto:mm-ma@computer.org). All submissions will undergo a blind peer review by at least two expert reviewers to ensure a high standard of quality. Referees will consider originality, significance, technical soundness, clarity of exposition, and relevance to the special issue topics. All submissions must contain original, previously unpublished research or engineering work. Papers must stay within the following limits: 6,500 words maximum, 12 total combined figures and tables with each figure counting as 200 words toward the total word count, and 18 references.

To submit a paper to the July-September 2013 special issue, please observe the following deadlines:

- 29 June 2012: Full paper must be submitted using our online manuscript submission service and prepared according to the instructions for authors (please see the Author Resources page at <http://www.computer.org/multimedia/author.htm>).
- 28 September 2012: Authors notified of acceptance, rejection, or needed revisions.
- 30 November 2012: Revisions due
- 31 January 2013: Final versions due.

### Questions?

Direct any correspondence before submission to the guest editors:

- Chong-Wah Ngo, City University of Hong Kong, [cwngo@cs.cityu.edu.hk](mailto:cwngo@cs.cityu.edu.hk)
- Changsheng Xu, Chinese Academy of Sciences, [csxu@nlpr.ia.ac.cn](mailto:csxu@nlpr.ia.ac.cn)
- Wessel Kraaij, TNO and Radboud University Nijmegen, [w.kraaij@cs.ru.nl](mailto:w.kraaij@cs.ru.nl)
- Abdulmotaleb El Saddik, University of Ottawa, Ontario, [abed@mcrlab.uottawa.ca](mailto:abed@mcrlab.uottawa.ca)

**[www.computer.org/multimedia](http://www.computer.org/multimedia)**