Anomaly Based Web Phishing Page Detection

Ying Pan, Xuhua Ding
School of Information Systems, Singapore Management University
{ypan,xhding}@smu.edu.sg

Abstract

Many anti-phishing schemes have recently been proposed in literature. Despite all those efforts, the threat of phishing attacks is not mitigated. One of the main reasons is that phishing attackers have the adaptability to change their tactics with little cost. In this paper, we propose a novel approach, which is independent of any specific phishing implementation. Our idea is to examine the anomalies in web pages, in particular, the discrepancy between a web site's identity and its structural features and HTTP transactions. It demands neither user expertise nor prior knowledge of the website. The evasion of our phishing detection entails high cost to the adversary. As shown by the experiments, our phishing detector functions with low miss rate and low false-positive rate.

1 Introduction

The goal of phishing attacks is to steal user identities and credentials. Phishing attackers use various tactics to lure or hijack a browser to visit bogus sites. They may choose social engineering, such as phishing emails, or/and technical subterfuge, such as trojan horses. A common factor among all phishing sites is that they maliciously mislead users to believe that they are other legitimate sites. Therefore, detecting phishing pages is essentially an authentication problem between humans and servers. Ideally, when a user visits a website, he or she is able to authenticate the engaged web server, even if the server is accessed for the first time. Unfortunately, no existing technical mechanism fully solves this problem. For example, SSL only authenticates a web server's IP address or hostname to a browser and protects the communication channel as well. Nonetheless, it provides no guarantee the HTML files sent by the web server are not misleading. Some schemes are proposed to enable a user to authenticate a server with a priori security association. However, those schemes are not applicable to websites which a user visits for the first time. Moreover, it requires user awareness of existence of the authentication step. Nonetheless, if the user is already alerted, a simple URL checking can prevent the phishing attack.

Although many anti-phishing schemes have been proposed, none of them effectively solves the authentication challenge. In this paper, we take a novel strategy to detect phishing pages. Our basic idea is the following. *Either explicitly or implicitly, every website claims an identity in the cyberspace. When a phishing site maliciously claims a false identity, it always demonstrates abnormal behaviors compared to a honest site, which are indicated by some web DOM objects in the page and HTTP transactions. We propose to detect a phishing website by capturing those anomalies.* The advantage of this approach includes: 1) it does not rely on any prior knowledge of the server or users' security expertise; 2) the adversary has much less adaptability since the detection is independent of any specific phishing strategy; 3) it causes no changes on users' existing navigation behavior.

We report our study in this paper. In the rest of this section, we discuss the related work. In Section 2, a synopsis of our work is presented. Our phishing detector comprises two components: an identity extractor presented in Section 3; and a web page classifier in Section 4. Our experimental results are reported in Section 5. Section 6 concludes this paper by discussing open problems and future work.

Related Work

Based on the methodologies used in existing anti-phishing schemes, we roughly group them into three categories: server based schemes, browser based schemes, and proactive schemes.

Server based schemes refer to those requiring server authentication to defend against phishing attacks. A seemingly straightforward approach is for clients to verify the credential presented by a web server. A credential is usually issued by a trusted third party and provides assurance on its bearer's identity. Typically, it is displayed as logos, icons, seals of the brand in the browser window to attract users' attention. Exemplary schemes include Content Verification Certificates[1], GeoTrust ToolBar[2], Trustbar [8] etc. Nonetheless, lacking a global wide public key infrastructure, many users just blindly trust or reject the credentials. A variant of web credential is to use a database or list published by a trusted party, where known phishing web sites are blacklisted. Examples include Websense[3], McAfee's anti–phishing filter[4], Netcraft anti-phishing system[5], Cloudmark SafetyBar[6], Microsoft Phishing Filter[7]. The weakness of this approach is its poor scalability and its timeliness. Note that phishing sites are cheap and easy to build and their average lifetime is only a few days. Another authentication approach is to share a secret, e.g. an image or a password, between clients and the server. A user trusts a Web server on the condition that the secret presented by the server matches his local one. For instance, Dynamic Security Skins [5, 6] allows a remote server to prove its identity in the form of an image which is unique for each user and each transaction such that users visually verify whether the image from the server matches its corresponding local image. However, this requires user awareness and prior knowledge. We argue that many phishing victims are not of security concerns. Otherwise, a simple URL checking will foil the attack.

Browser based schemes embed anti-phishing measures into Web browsers. These browsers regulate web pages' visual behaviors to prevent cheating. For example, Trusted Browser [19, 20] uses synchronized random colored boundary to secure the path from users to their browser. The trusted status content is marked in the trusted window whereas the server content is shown in the untrusted window. Moreover, the boundary style of the windows is changed between inset and outset by a random number generator at a certain frequency in concert with a reference window. SpoofGuard [3] uses domain names, URLs, links, and images to measure the similarity between a given page and the pages in the caches or histories. AntiPhish [10] compares the domains for the same sensitive information in web pages to the domains in the caches. PhishHook [16] converts a web page to "normal form" through text, images and hyperlinks transformations. EBay ToolBar[8] monitors the visited web pages and provides a warning in the form of colored tab for eBay's users. Spoofstick[9] just provides basic information about the domain name of the web site. The limitation of browser based schemes is that it requires prior knowledge of the target site, which is unfortunately not always available. More importantly, since phishing attackers are able to update the enticement techniques to get around those schemes, the effectiveness of these schemes is not convincing.

In a proactive manner, a set of techniques are designed to capture phishing sites in the Internet. Liu et al. [11] propose an approach to detecting the phishing web pages based on visual similarity (i.e. block level similarity, layout similarity and overall style similarity). Cyveillance Fraud Management[10] uses proprietary Internet monitoring technology to identify phishing-related activity such as suspicious domain registrations, phishing lures, spoofed sites and the post-attack sale of compromised credentials. Others include Internet Identity's Domain Security Audit[11]. These approaches mainly are motivated to protect corporations' interests. Nonetheless, they do not directly defend phishing attacks for users.

2 Synopsis

2.1 Overview

A phishing website usually demonstrates abnormal discrepancies between its web objects or HTTP transaction and its claimed identity. Our anti-phishing scheme is mainly built upon the detection of those identity relevant anomalies. The proposed phishing detector consists of two components described below:

1. Identity Extractor: An *identity* of a web site is defined as a set of words (i.e. character strings) which uniquely identify the web site's ownership in the cyberspace. Typically, the identity is an abbreviation of the organization's full name and/or a unique string appearing in its domain name. In a web page, the identity is indicated in a number of objects or properties. No existing work has addressed the issue of identity retrieval from web objects. In our scheme, we develop an algorithm by making use of keyword extraction techniques from information retrieval literature. Note that an identity is not exactly the keywords appearing in a web page. Therefore, no off-the-shelf keyword extraction scheme is directly applicable to our problem. Section 3 presents the details of our identity extractor.

2. Page Classifier: A phishing web site with its own identity A attempts to claim a false identity B by imitating the visual effects of web site B. Nonetheless, in order to accomplish the attack, some web objects or properties, which are relevant to identity A, can not be freely fabricated. We refer to these objects/properties as *structural features*. One

[1]`http://www.comodo.com`
[2]`http://www.trustwatch.com`
[3]`http://www.websense.com`
[4]`http://www.mcafee.com`
[5]`http://toolbar.netcraft.com`
[6]`http://www.cloudmark.com`
[7]`http://www.microsoft.com/mscorp/safety/technologies/antiphishing/at_glance.mspx`
[8]`http://pages.ebay.com/ebay_toolbar`
[9]`http://www.corestreet.com/spoofstick/`
[10]`http://www.cyveillance.com`
[11]`http://www.internetidentity.com`

source of structural features is those identity related W3C DOM objects in a web page, e.g. URI domain of an anchor. Another source of structural features is HTTP transactions. Examples include domain names in DNS records or cookies. With a high probability, the features extracted from a phishing web site do not match its claimed identity, which are considered as anomalies. The detection of anomalies leads to the judgement of the genuineness of the web site. Given an identity and a set of features, the task of determining the genuineness of a web page is executed by Vapnik's Support Vector Machine (SVM) [17, 4], which is a well known classifier and has been widely employed in pattern recognition. Before its deployment, the SVM is trained by a large data set of identities and features from both authentic websites and phishing sites.

To deploy the proposed phishing detector in a browser, a user downloads a plug-in comprising both an identity extractor and a page classifier from a trusted server. Note that such a server is not necessarily globally trusted. For example, it can be a server within the user's organization. The plug-in should be trained by the server before it is provided to the users. Once downloaded, the plug-in functions independently without involvement of any third party. The architecture of our scheme is shown in Figure 1. When a browser opens a web page, an identity is extracted by the identity extractor and the structural features of the page are extracted and converted into vectors. The SVM-based page classifier, which has been trained *offline*, takes the vectors as an input and outputs a phishing label.

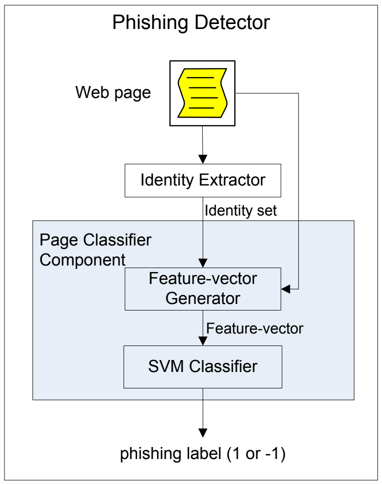


Figure 1. The architecture of the phishing detector

Next, we analyze the properties of W3C DOM objects and phishing pages as the technical basis of our mechanism.

2.2 W3C DOM Objects

A structured web page is composed of W3C DOM objects [1]. Among them, we list below five categories, based on their relevance to the web identity. These types of objects are the main sources which the identity and features are derived from.

- **Keyword/Description (KD)** These objects and properties are used to claim ownership or copyright of web pages, or to describe the content. Therefore, they usually explicitly present their web identity. For example, <title>PayPal - Welcome</title> in `http://www.paypal.com`.

- **Request URL (RURL)** External objects (such as images, external scripts, CSS) in a web page are loaded from other URLs. For a normal corporate web site, a large percent of those URLs are in its own domain. Therefore, they are consistent with the web identity in most cases. For instance, <img src="https://home.peoplepc.com/i/60/common/ppco_logo.gif"> in `http://www.peoplepc.com`.

- **URL of Anchor (AURL)** This type of objects has a similar characteristic to RURL. A high portion of anchors in a legitimate web page point to the same domain as the page itself. One example is <a href="http://www.ebay.com/"> in `http://www.ebay.com`.

- **Server Form Handler (SFH)** For security and management reasons, most finance/e-business web portals require usernames and passwords. Therefore, those pages usually contain a server form handler. For example, <form action="/inetSearch/index.jsp" method="post" target="_top"> in `http://www.chase.com`. For phishing sites, the SFH-s usually are void or refer to a different domain.

- **Main Body (MB)** The Main Body of a page is another source of identity relevant information. Many web sites give a description of their corporations in the main body of their homepages.

A summary of examples is presented in Table 1.

2.3 Analysis of Phishing Pages

We provide below a list of characteristics of phishing pages based on our study of about 300 phishing sites from APWG's repository.

Category	DOM objects or properties
KD	title of all elements, TITLE, content of META name = "keywords"/"Description"/"copyright", content of META http-Equiv = "keywords"/"Description"/"copyright", alt of IMG/AREA/INPUT/APPLET /OBJECT, caption/summary of TABLE, ADDRESS
RURL	src of FRAME/IFRAME /IMG /INPUT/SCRIPT, href of BASE/LINK, domain/location of document, background of BODY, code/codeBase of APPLET, baseURI of OBJECT
AURL	host/hostname/href of A/AREA
SFH	action of FORM
MB	text of BODY

Table 1. Identity Relevant Categories of W3C DOM objects and properties.

- A phishing page disguises itself as a real site by showing the identity of the real site in its KD category objects. Noticing the logos or titles displayed in their browsers, users may mistakenly believe the attacker's false claim.

- **Abnormal URL** The hostname in URL or revolved from the IP address of the phishing web server does not match its claimed identity.

- **Abnormal DNS record** A full DNS record usually has identity relevant information. For phishing sites, either the record of the hostname is not found in the WHOIS database or the claimed identity is not contained in the record.

- **Abnormal Anchors** Normally, a web page of a web site does not stand alone. Its anchors usually link to other related pages to provide navigation guidance. The phishing attackers, however, are generally not interested in developing a large number of web pages. Therefore, a web page is suspicious when one of the following cases occurs.
  (a) Domains of most of AURL-s are different from the page's domain and these domains contain the claimed identity; (b) Anchors do not link to any page, e.g., AURL is "file:///E:/" or "#".

- **Abnormal Server Form Handler** SFH-s containing "about:blank" or an empty string("") are suspicious because legitimate web servers always take actions upon a submitted form. SFH-s with a different domain is suspicious, since it is rare that a form is handled by a server in a foreign domain.

- **Abnormal Request URL** A phishing page usually has a high percentage of object references to its target (the real site) in order to show similar appearance. By contrast, a legitimate web page usually requests its objects from its own domain.

- **Abnormal cookie** Cookies are deposited by a web server to its client, so that the client information is sent to the server when the browser visits it again. Therefore, a cookie is usually bound to its web server's domain. For a phishing site, its cookies either point to its own domain, which is inconsistent of the claimed identity, or point to the real site which is inconsistent with its own domain.

- **Abnormal certificate in SSL** For SSL-enabled phishing sites, public key certificates are employed. In many phishing attacks, the Distinguished Names (DN) in their certificates are inconsistent with the claimed identities.

Based on the above analysis, we extract the related web objects from a web page and convert them into a feature vector. The page classifier (see Section 4) takes the feature vector as input and determines whether the page is bogus or not.

3 Identity Extractor

Identity extraction is to acquire an abstraction of the ownership of a web site. To the best of our knowledge, no existing scheme directly deals with this issue. We design an algorithm using techniques in keyword extraction in information retrieval literature.

Many keyword extraction algorithms (e.g. [7, 9, 12, 14]) exploit document information, including the hierarchical structure of documents, the frequency of terms, the co-occurrences of terms etc. The basic idea is that terms with the highest occurrence frequency are of the highest probability of being keywords, assuming that keywords are usually repeated more frequently than other words.

One may argue that the web identity is not exactly the keyword(s) of a web page. Nonetheless, we observe that in some types of DOM objects/properties, the identity related words are more frequently cited than other words. Our identity extractor determines the web identity by considering the following DOM objects:

1. **Title:** the title of one web page (namely, the text between the tag <title> and $< \backslash$title>).

2. **Description:** the content property of the META whose name or http-equiv is “description”.
3. **Copyright:** the content property of the META whose name or http–equiv is “copyright”.
4. **ALT/title:** the alt and title properties of the DOM objects such as IMG, AREA, INPUT, APPLET, OBJECT.
5. **Address:** the text of address objects.
6. **Body:** the text in the main body or the images in a web page. There are many technologies to recognize texts from one image, such as Optical Character Recognition (OCR)[12].

Taking a web page as input, the identity extractor first forms an identity relevant object set, denoted by $\mathcal{D}$. By processing $\mathcal{D}$, it initiates a word set $\mathcal{W}$ which comprises all identity candidates. The selection of web identity from $\mathcal{W}$ is based on the observation that the distribution of identity words deviates from ordinary words' distributions. In specific, we choose χ^2-test to measure the statistical significance of deviation, which is widely adopted in keyword extraction schemes [7, 12, 18]. The algorithm of our identity extractor is shown below:

Identity Extraction Algorithm

Input: Web page P; Output: Identity set $\mathcal{I}$

1. Construction of object set $\mathcal{D}$: Initialize $\mathcal{D} = \emptyset$. Parse P to obtain object title, description, copyright,alt/title, address,and body. Insert all of them to set $\mathcal{D}$, except body object. The reason for excluding body is that body object is not necessarily identity related. Moreover, it is text rich and its own keywords, e.g. ”online”, might have also frequent occurrence.
2. Construction of word set $\mathcal{W}$: Initialize $\mathcal{W} = \emptyset$. For $\forall d \in \mathcal{D}$, extract all words from d separated by a stop mark[13]. If a word w is not a stop word [15](such as 'about', 'after', 'back', etc), set $\mathcal{W} = \mathcal{W} \bigcup \{w\}$. Words with same stem, according to Porter algorithm [13], are treated as one element in $\mathcal{W}$. For $\forall d \in \mathcal{D}$, if no word from d is in $\mathcal{W}$, $\mathcal{D} = \mathcal{D} - \{d\}$.
3. Calculation of the occurrences: For $\forall w \in \mathcal{W}, \forall d \in \mathcal{D}$, set $C_{w,d}$ as the number of occurrences of w in d. Set $C_w = \sum_{d \in \mathcal{D}} C_{w,d}$ as the total number of occurrences of w.
4. Supplement of body object: If $\forall w_1, w_2 \in \mathcal{W}, C_{w_1} = C_{w_2}$, then the body object is added into set $\mathcal{D}$ and recount C_w in the same way as above, for all $w \in \mathcal{W}$.
5. Calculation of the term frequency $\alpha_{w,d}$ for w with respect to d: For all $d \in \mathcal{D}$, let H_d denote the number of words appearing in d. Therefore,
$$\alpha_{w,d} = \frac{C_{w,d}}{H_d}$$
6. Calculation of expected probability β_d for all $d \in \mathcal{D}$: Compute $N_d = \sum_{w \in \mathcal{W}} C_{w,d}$ and $\beta_d = \frac{N_d}{\sum_{d \in \mathcal{D}} N_d}$. We assume that all ordinary words have the same distribution. Therefore, given a word w, β_d is the expected probability of the event that w occurs in d.
7. Calculation of χ^2 value: The statistical value of χ^2 for word w is defined as $\chi^2(w)$:
$$\chi^2(w) = \sum_{d \in \mathcal{D}, C_{w,d} \neq 0} \frac{(\alpha_{w,d} - C_w \cdot \beta_d)^2}{C_w \cdot \beta_d}$$
8. Output an identity set: The word(s) with the largest χ^2 value is/are inserted into the identity set $\mathcal{I}$.

[12]http://jocr.sourceforge.net/

[13]Stop marks include '.', ',', '?', '!',' ' etc.

The produced $\mathcal{I}$ may comprise a single word or multiple words having the same highest χ^2 score. In either cases, the co-occurrence of all words in $\mathcal{I}$ is considered as the web identity of the page P. The order of their occurrence is ignored. Based on $\mathcal{I}$, the page classifier described in next section extracts structural features from page P and judges its trustworthy.

4 Page Classifier

4.1 Support Vector Machine

Our page classifier employs Support Vector Machine (SVM)[17], a well-known algorithm for classification. We consider a binary classification problem. Let $\Omega = \{x_i\}_{i=1}^n$ be a set of n training vectors, where x_i is a m-dimension vector labelled by $y_i \in \{1, -1\}$, with $y_i = 1$ and $y_i = -1$ indicating membership of x_i to the Class 1 and Class 2 respectively. SVM classifier has the capability of locating a hyperplane in the m-dimension space, which separates Ω into two classes such that all vectors on one side of the hyperplane have label 1 while those on the other side have label -1. Given this property, SVM is used to classify vectors without prior knowledge of their distribution. The training process enables SVM to develop a classification model, based on which a trained SVM is able to label a vector from a test set.

Since a web page is either faked or authentic, phishing detection is by nature a binary classification problem. We make use of SVM as the page classifier. Its input is a 10-dimension vector representing a web page's 10 structural features. It outputs a label 1 indicating a phishing page or a label -1 indicating an authentic one. Now, we proceed to show how to derive a structural feature vector from a web page.

4.2 Feature Vector Generation

Section 2.3 summarizes a group of features of phishing pages. Although the tactics of visual imitation vary wildly, those identity relevant structural features are difficult to hide and are independent of visual effects. To facilitate SVM-based classification, we quantify those features into vectors. Throughout the rest of this section, we consider a web page P at URL L. The output from the execution of the identity extractor on P is $\mathcal{I} = \{s_0, s_1, \cdots, s_k\}$, where s_i is a character string, $0 \leq i \leq k$, k is the number of extracted identity words. The details of the feature vector initialization of P are discussed below.

Feature 1: URL address (F_1) A web page's URL (Uniform Resource Location) is unique in the cyberspace. For a regular web site, its identity is usually part of its URL. For a phishing site, its true URL is usually similar but different from its claimed identity. Specifically, we consider three cases: 1) The URL address L is an alias of another host of address L' and no s_i is a substring of L' for all $0 \leq i \leq k$. 2) The URL address L is an IP address, which is resolved into a host name L'. However, no s_i is a substring of L' for all $0 \leq i \leq k$. 3) No s_i is a substring of L for all $0 \leq i \leq k$. In any of these three cases, $F_1 = 1$. If one page only uses the IP address which can not be resolved into any hostname, $F_1 = 0$. Otherwise $F_1 = -1$.

Feature 2: DNS record (F_2) Many websites register the information of their domains (such as the contact information of the person or organization) in the WHOIS databases [14]. Using the IP address of L as a keyword, we retrieve a DNS record R, if any, from the whois database. If all s_i is a substring of R for $0 \leq i \leq k$, $F_2 = -1$. If no record is returned from the database, $F_2 = 0$. Otherwise, $F_2 = 1$.

Features 3.1 − 3.3: URL of anchor Anchors in a normal web page usually point to pages in the same domain. For phishing pages, there are three possible abnormalities listed below. In the following, let A_a be the total number of anchors in page P.

1. Feature 3.1: Nil anchor (F_{31}) An anchor is called a *nil anchor* if it points to nowhere. Examples are <a href="#">, <a href="#skip">, <a href="javascript::void(0)">, etc. The percentage of nil anchors in a page reflects the degree of suspiciousness. The higher the percentage, the more likely that P is a phishing page. F_{31} is assigned as follows:

$$F_{31} = \begin{cases} 0 & A_a = 0 \\ A_{nil}/A_a & A_{nil} > 0 \\ -1 & \text{otherwise} \end{cases}$$

   where A_{nil} is the number of nil anchors in P.

2. Feature 3.2: ID anchor (F_{32}) An anchor in P is called an *ID anchor* if it points to a domain which is different from L and contains all s_i in $\mathcal{I}$ for $0 \leq i \leq k$. This feature is suspicious since the phishing pages usually use too many ID anchors to have the same appearance as the real site. F_{32} is assigned as follows:

$$F_{32} = \begin{cases} 0 & A_a = 0 \\ A_{id}/A_a & A_{id} > 0 \\ -1 & \text{otherwise} \end{cases}$$

   where A_{id} is the number of ID anchors in P.

3. Feature 3.3: Domain anchor (F_{33}) An anchor in P is called a *domain anchor* if it points to a foreign domain, i.e. a domain different from the one in L. In general, an anchor pointing to a foreign domain is not an abnormality. Nonetheless, an exceptionally high percentage of domain anchors in a corporate web portal make it less trustworthy. Therefore, F_{33} is assigned in a conservative way as shown below:

$$F_{33} = \begin{cases} 0 & A_a = 0 \\ -A_l/A_a & A_l > 0 \\ A_f/A_a & \text{otherwise} \end{cases}$$

   where A_l is the number of the anchors pointing to the local domain and A_f is the number of domain anchors. Note that $A_l + A_f \leq A_a$ since some anchors could be invalid.

Feature 4: Server form handler (F_4) Note that the ultimate goal of phishing attacks is to steal users' private information, such as user name and password. Phishing pages often contain forms requesting user inputs. However, the handler of such a form in a phishing page usually refers to the real site or simply is void. We set $F_4 = 1$, if there is an occurrence of any void handler (e.g. <form action="#">, <form action="about: blank">, <form action="javascript:true">), or any handler referring to a foreign domain. If there is no handler in P, $F_4 = 0$; and $F_4 = -1$ for other cases.

Feature 5.1 − 5.2: Request URL Web pages are object rich, containing numerous objects including images, CSS files, scripts etc. For a regular web page of a corporate portal, a large percent of objects are loaded from its own domain. Only a small portion of them are from foreign domains. In those phishing pages, most objects are copied

[14]There are some famous whois databases, such as `http://www.whois.net/`, `http://networksolutions.com/`, etc.

or loaded from the real sites since the attackers intend to reduce their cost of faking. Therefore, a request URL is an indication of phishing attacks. Let R_a denote the total number of request URLs in P.

1. Feature 5.1: ID Request URL (F_{51}) A request URL $\hat{L}$ in P is called an *ID request URL*, if $\hat{L}$ is in a different domain from L and all $s_i \in \mathcal{I}$ are substrings of $\hat{L}$. We observe that more ID request URLs in P indicates a higher probability of P being faked.

$$F_{51} = \begin{cases} 0 & R_a = 0 \\ R_{id}/R_a & R_{id} > 0 \\ -1 & \text{otherwise} \end{cases}$$

   where R_{id} is the number of ID request URLs.

2. Feature 5.2: Domain request URL (F_{52}) A request URL $\hat{L}$ in P is called a *domain request URL* if $\hat{L}$ is in a different domain with L. Similar to domain anchors, exceptionally high percentage of domain request URL implies a possibility of a phishing attack. We also assign this feature in a conservative manner:

$$F_{52} = \begin{cases} 0 & R_a = 0 \\ -R_l/R_a & R_l > 0 \\ 1 & \text{otherwise} \end{cases}$$

   where R_l is the number of request URLs in the local domain.

Feature 6: Domain in cookie (F_6) The main purpose of cookies is to maintain the state of a user in the server side. Thus, for a normal web page, its own domain appears in its cookies. In some phishing pages, the domain in cookies is set as the domain of the real site in order to imitate the real site. Therefore, $F_6 = 1$ if any foreign domain exists in cookies of P; $F_6 = 0$ if no domain is claimed in cookies or P has no cookies; $F_6 = -1$ for other situations.

Feature 7: Certificate in SSL (F_7) In a SSL transaction, the web client usually requests the server to present a public key certificate. For a legitimate web site, the presented certificate contains identity relevant information, e.g. the Distinguished Name (DN). Moreover, a certificate for web usage usually defines its serving URL explicitly. A phishing site may choose to use the same certificate as its victim's one. Otherwise, its own certificate would not match the identity it attempts to impersonate. So, $F_7 = 1$ if one of the claimed identities does not appear in the certificate attached to P or the URL specified in the certificate is different from L; $F_7 = 0$ if the SSL is not applied; and $F_7 = -1$ for other cases.

Consequently, the features of P are quantified into

$$V_P =< F_1, F_2, F_{31}, F_{32}, F_{33}, F_4, F_{51}, F_{52}, F_6, F_7 >$$

When in training process, feature vectors are derived from pages in the training set. We label each vector 1 or -1 by manually determining its genuineness. When in classification process, the page classifier takes as input the feature vector of the web page under investigation and outputs a label.

5 Experiments

We make experiments with real data collected from the web to validate the proposed schemes. We download web pages from 100 most frequently attacked web sites according to a report from Millersmiles[15]. Our phishing data source is from the repositories maintained by Millersmiles[16], Castlecops[17], and phishing emails in our own mail boxes. Since most phishing sites are ephemeral, we only manage to collect 279 sites so far. We also notice that those 279 phishing attacks only have 49 different targets. Most victims are finance institutes, e.g. Bank of America, Chase, and e-commerce companies, e.g. eBay, Paypal. These 279 spoofy pages and 100 authentic pages constitute the page pool used in our experiments.

5.1 Experiments of Identity Extractor

We apply our identity extraction algorithm to all pages in the pool. For each page, we manually retrieve its identity and use it as the baseline to verify the output of the identity extractor. An output is correct if it equals to the manually derived identity. Let $\lambda = \frac{n}{N}$ denote the success rate of the identity extractor, where N is the number of web pages in test and n is the number of pages whose identities are correctly extracted by the algorithm.

We provide three examples in Table 2. For each of them, we list the words with five largest χ^2 value. Those labelled by an asterisk are manually selected identities. Those with the highest scores in each column are the outputs from the extractor.

Phishing page #1504[18] is at http://62.123.219.27/.x/.paypal-sk/login.htm, impersonating http://www.paypal.com. Phishing page #1822[19] is at http://www.adword-planet.com/onlineid-sessionload/, impersonating http://www.bankofamerica.com. Clearly, the identity extractor made correct choices in Example 1 and 3. The success rate for all pages in the pool is shown in Table 3.

It seems that the extractor performs better in dealing with phishing pages than with legitimate ones. This is due to the

[15]http://www.millersmiles.co.uk/scams.php

[16]http://www.millersmiles.co.uk

[17]http://castlecops.com/modules.php?name=Fried_Phish&fp=phish

[18]http://www.millersmiles.co.uk/report/1504

[19]http://www.millersmiles.co.uk/report/1822

Example 1: Phishing Page #1504		Example 2: Phishing Page #1822		Example 3: www.chase.com	
Word	χ^2	Word	χ^2	Word	χ^2
paypal*	12.4873	bank	5.1600	chase*	8.9979
welcome	2.4694	America*	2.9143	security	0.8589
money	2.4475	online	1.9533	online	0.8589
send	1.6518	high	0.7462	account	0.8589
let	0.5369	standard	0.7462	com	0.7848

Table 2. Identity Extraction Results of Three Web Pages

	Phishing pages	Legitimate Pages	Total
N	279	100	379
n	246	71	317
λ	88%	71%	84%

Table 3. Success Rate(λ) of the Identity Extractor: n is the number of correctly extracted pages, N is the total number of pages

different distributions of the legitimate page set and phishing page set. Those official sites are independent from each other while some of the phishing sites are correlated since all of those 279 attacks only have 49 different targets. The similarity among phishing sites of the same target makes the success rate much higher. Thus, we argue that the success rate for legitimate pages more accurately reflects the performance of our identity extractor, since the test samples are independent of each other.

5.2 Experiments of Page Classifier

We run experiments to validate the page classifier and selection of features. LibSVM [2] is selected as an implementation of SVM. The experimental data set, including a training data set and a test data set, is randomly drawn from the page pool comprising 279 phishing pages and 100 official pages collected from the world wide web.

As a rule of thumb, half of the training set are phishing pages and the other half are legitimate pages. The balance of two types of data maximizes the training effect of SVM. In our experiments the training set consists of 50 false pages and 50 authentic pages. From the remaining pages in the page pool, the test data set is generated at random. Note that the size and the composition of the test data affect the classifier's false positive rate and miss rate. Our tests simulate the practical deployment of the scheme. The test data set contains 50 pages in total, based on the estimation that an ordinary user visits 50 different web sites per day in maximum. Seven groups of testing sets are used with different portions of phishing pages, i.e., $2\%, 6\%, 10\%, 20\%, 30\%, 40\%, 50\%$ respectively. After the SVM-based classifier is trained, it is challenged to label all pages in the test set. A positive label 1 indicates a phishing page while a negative label -1 indicates a legitimate page.

The performance of our phishing detector is measured by two metrics: miss rate (denoted by ρ) and false positive rate (denoted by σ), defined as

$$\rho = \frac{n_m}{N'} \ , \ \sigma = \frac{n_f}{N''}$$

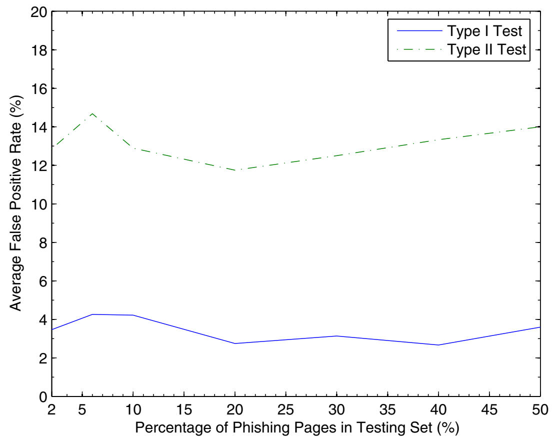
where N' is the number of phishing pages in the testing set; n_m is the number of phishing pages which are mistakenly labelled as negative; N'' is the number of real pages in the testing set; n_f is the number of legitimate pages which are mistakenly labelled as positive.

To precisely study the effectiveness of the classifier, we design two types of experiments. Type I test measures the independent performance of the classifier without the involvement of the identity extractor. In this type of tests, the web page identities are manually selected and fed to the classifier. Type II test measures the performance of the entire detection scheme, whereby the page classifier makes use of the identities produced by the identity extractor. In both types of experiments, we run ten times for each group of testing sets. In each round of execution, the classifier is trained from the scratch using different training sets. Figure 2(a) shows the average false-positive rate and Figure 2(b) shows the average miss rate with respect to the different percentage of phishing pages.

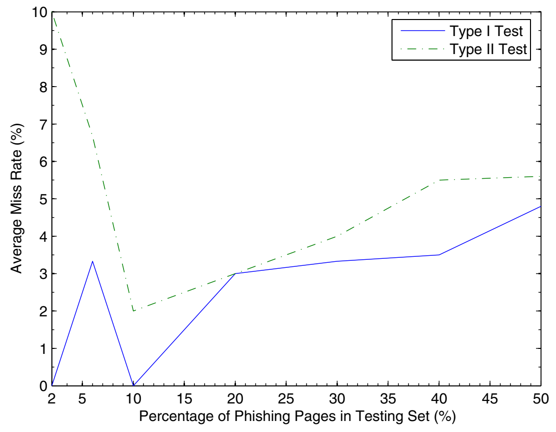
The above figures show that the performance of the identity extractor has impact on the classifier's performance, especially on its false positive rate. This is due to the fact that the identity extractor has lower success rate when processing legitimate pages as shown in Table 3. Therefore, a wrongly extracted identity from a legitimate web page consequently cause the classifier to produce a wrong label.

5.3 Discussion

In contrast to existing schemes, our scheme is not designed to neutralize a particular type of phishing attack. The strength of our scheme is independent of attack methods. In our scheme, the phishing detector systematically inspects the structural features of a web page. The traces



(a)



(b)

Figure 2. False positive rate (σ) and Miss rate (ρ) of Type I test and Type II test.

of attacks are captured by anomaly detection. Remarkably, our scheme has the virtue that the adversary has much less adaptability to evade the detection, compared to dealing with other anti-phishing schemes. Note that the anomalies in phishing pages are mostly due to the attackers' intention to save their attack cost, in specific, the time cost for designing and storage cost for hosting phishing pages. For example, the attacker simply uses links to images on the victim's website, instead of providing them in its own domain. Therefore, to avoid being detected by our scheme, the attackers are forced to make extra efforts in order to reduce the number of anomalies. Although it is an open problem to build a quantified model to measure the amount of effort, the intuition is that the fewer anomalies, the higher the attack cost and the closer the attacker's expense to the cost in building the same legitimate web site. In fact, the current low cost of phishing is one of the main factors causing the proliferation of phishing sites. A prohibitively higher cost in building phishing sites discourages a large portion of potential attackers. We envisage our approach as an efficient means to curb the trend of phishing attacks.

6 Conclusion

In this paper, we propose a new anti-phishing approach based on DOM object anomalies detection. Our scheme neither requires online interactions with a third party, nor requires users to change their navigation behavior. The experimental results show that both its false-positive rate and miss rate are reasonably low. Moreover, this approach is resistent to adaptive phishing attackers. A complete evasion of our phishing detection tolls the adversary a prohibitively high cost.

Our future effort will focus on more intensive experimental validation and algorithm improvements. A further study on the phishing attacks and web structural features is desired to improve the effectiveness of the phishing detector. We observe that the identity extraction is critical to a successful detection. However, it remains as an open problem how to extract the identity from web pages with an overwhelming success probability. A possible improvement is to explore information from logo images in web pages.

Another dimension of future work is to combine the anomaly-based approach with other techniques. For instance, it may co-work with a search engine. Given a web page, the phishing detector first extracts its identity, which is then fed into a search engine as a keyword. If the search engine returns a URL with the same domain, this web page under checking is genuine with an overwhelmingly high probability.

References

[1] http://www.w3.org/tr/dom-level-2-html/.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client–side defense against web–based identity theft. In *Proceedings of 11th Annual Network and Distributed System Security Symposium*, 2004.

[4] C. Cortes and V. Vapnik. Support–vector networks. *Machine Learning*, 20:273–297, 1995.

[5] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, pages 77–88, 2005.

[6] R. Dhamija and J. D. Tygar. Phish and hips: Human interactive proofs to detect phishing attacks. In *Human Interactive Proofs: Second International Workshop*, pages 127–141, 2005.

[7] F. Fukumoto, Y. Suzukit, and J. Fukumoto. An automatic extraction of key paragraphs based on context dependency. In *Proceedings of the 5th conference on Applied natural language processing*, pages 291–298, 1997.

[8] A. Herzberg and A. Gbara. Trustbar: Protecting (even nave) web users from spoofing and phishing attacks. Cryptology ePrint Archive, Report 2004/155, 2004. `http://eprint.iacr.org/`.

[9] Y. H. Kerner. Automatic extraction of keywords from abstracts. In *Proceedings of the 7th International Conference on Knowledge–Based Intelligent Information and Engineering Systems, KES 2003*, pages 843–849, 2003.

[10] E. Kirda and C. Kruegel. Protecting users against phishing attacks with antiphish. In *Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC)*, pages 517–524, 2005.

[11] W. Liu, G. Huang, X. Liu, M. Zhang, and X. Deng. Detection of phishing webpages based on visual similarity. In *Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, May 10-14, 2005 - Special interest tracks and posters*, pages 1060–1061, 2005.

[12] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co–occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[13] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[14] M. Rossignol and P. Sébillot. Combining statistical data analysis techniques to extract topical keyword classes from corpora. *Intell. Data Anal.*, 9(1):105–127, 2005.

[15] G. Salton. *Automatic Text Processing.* Addison–Wesley, 1988.

[16] M. Stepp. Phishhook: A tool to detect and prevent phishing attacks. In *DIMACS Workshop on Theft in E-Commerce: Content, Identity, and Service*, 2005.

[17] V. N. Vapnik. *The nature of statistical learning theory.* Springer, New York, 1995.

[18] Y. Watanabe, M. Murata, M. Takeuchi, and M. Nagao. Document classification using domain specific kanji characters extracted by x2 method. In *Proceedings of the 16th conference on Computational linguistics*, pages 794–799, 1996.

[19] Z. E. Ye and S. Smith. Trusted paths for browsers. In *Proceedings of the 11th USENIX Security Symposium*, pages 263–279, 2002.

[20] Z. E. Ye and S. Smith. Trusted paths for browsers. *ACM Transactions on Information and System Security*, 8(2):153–186, 2005.