Detection of Hidden Fraudulent URLs within Trusted Sites using Lexical Features

Enrico Sorio, Alberto Bartoli, Eric Medvet
DIA - Engineering and Architecture Dept.
University of Trieste, Italy
enrico.sorio@phd.units.it, bartoli.alberto@units.it, emedvet@units.it

***Abstract*—Internet security threats often involve the fraudulent modification of a web site, often with the addition of new pages at URLs where no page should exist. Detecting the existence of such *hidden* URLs is very difficult because they do not appear during normal navigation and usually are not indexed by search engines. Most importantly, drive-by attacks leading users to hidden URLs, for example for phishing credentials, may fool even tech-savvy users, because such hidden URLs are increasingly hosted within trusted sites, thereby rendering HTTPS authentication ineffective. In this work, we propose an approach for detecting such URLs based only on their lexical features, which allows alerting the user before actually fetching the page. We assess our proposal on a dataset composed of thousands of URLs, with promising results.**

***Keywords*-phishing; web site defacement**

I. Introduction

Internet security threats often involve the fraudulent modification of a web site. Such modifications may consist in: (i) changes to existing pages, or (ii) addition of new pages at URLs at which no page is supposed to exist. Fraudulent changes to existing pages may be useful for a variety of purposes, including *defacement*, addition of links to pages to be promoted in search engine results (*search spam* or *web spam*), exploitation of browser vulnerabilities for spreading of *malware*. Detecting such changes may be relatively easy or very difficult, depending on the specific change: an administrator (or a user) looking at a defaced page realizes immediately that there has been an intrusion, while the addition of a small malicious piece of Javascript may easily remain undetected.

Fraudulent creation of pages at URLs at which no page is supposed to exist may be useful for essentially the same reasons, except that in this case defacements are not visible to all users but only to those users that know the URL of the page containing disturbing content. These forms of defacements are meant to be a proof of ability of their authors and a significant fraction of the defacements hosted on ZoneH (`http://www.zone-h.org`) indeed belong to this category.

Detecting the existence of pages as such *hidden* URLs may be very difficult, because administrators will likely never see any anomaly in the served content and most users will never visit those URLs. For example, a large scale study on defacements hosted on Zone-H showed that the typical reaction time for recovering the defaced page is in the order of several days [1]. Intrusions of this form have become particularly appealing to attackers, for example in the context of illegal drug trade [2], and affect even sites of public interest. For example, a recent analysis of sites belonging to Italian Public Administrations has found that more than 9% of the analyzed domains host content that admittedly should not be there [3].

It is important to point out that phishing campaigns coupled with attacks of this sort may be extremely dangerous and fool even tech-savvy users: HTTPS–the main and ubiquitous line of defense in sensitive web sites—does *not* provide any defense in this respect. HTTPS ensures secrecy, integrity and authentication by means of cryptographic techniques. The problem is, the server site is authenticated as a whole: thus, any page coming from that site appears as being legitimate, from the HTTPS point of view.

The strategies for persuading unsuspecting users to visit hidden fraudulent pages can differ, but all the methods require that the user takes some action, usually by clicking on a link which points to the fraudulent URL. Ideally, when a user clicks on an unknown URL, he should assess the risk associated with her action. This risk assessment is indeed a difficult task for common users and is furtherly exhacerbated by the fact that a URL could refer a fraudulent content which is hosted within a site *trusted* by the user, i.e., an HTTPS-authenticated site routinely accessed by the user and whose administrators perform their best effort to host content that is indeed genuine and not harmful.

In this paper we present an approach for the detection of hidden fraudulent URLs before actually fetching the corresponding page. A system with this ability could be deployed in a variety of ways, for instance, within an e-mail client or within a web browser and trigger an alert to the user before actually accessing the fraudulent page itself. It could also be deployed within a web proxy, at the outside border of an organization, as a layer for a defense in depth strategy—of course, an approach like ours that does not actually analyze the page content is almost not useful for detecting malware-serving pages.

The peculiarity of our proposal consists in not using any feature related to the domain part of the URL, i.e., the URL portion which identifies the host. The rationale for this requirement is our focus on addressing fraudulent

978-0-7695-5008-4/13 $26.00 © 2013 IEEE
DOI 10.1109/ARES.2013.31

CPS Conference Publishing Services

URLs inserted into trusted web sites. In this scenario, the domain part of the URL is obviously not a discriminant between fraudulent and legitimate URLs belonging to the same web site. For the same reasons, we purposefully avoided using other domain-related features, e.g., `whois` queries or geographic properties.

We use lexical features extracted from the URL to be classified, excluding the domain part of the URL. These features are then input to a Support Vector Machine. We also propose two variants of this method, which augment the features available for classification based on the responses to HTTP requests directed at the site hosting the URL to be classified, but that do not involve fetching the actual content of that URL.

Our approach effectiveness was assessed on two categories of hidden fraudulent URLs: hidden phishing pages and hidden defacements. The two datasets are composed of about 6500 and 2150 URLs respectively. Our approach achieves an accuracy of about 96% for the phishing category and 99% for the defacement one.

II. Related work

The problem of detecting fraudulent URLs is long-established and has been studied from several points of view. As far as we know, though, this is the first approach focussed on detecting fraudulent URLs that are hosted on a trusted (in the sense clarified above) web site. With respect to other existing approaches for detecting fraudulent URLs, we (i) excluded the domain part from the feature used for classifying a URL and (ii) we assessed explicitly our approach ability to discriminate between fraudulent and legitimate URLs belonging to the *same* web site.

Almost all the previous works in this area focused on the detection of phishing URLs and URLs related to spam-advertised web sites. In this paper we also consider hidden *defacements* URLs—a quick look at the on-line defacement archive http://www.zone-h.org shows that exploit of this form occur routinely at organizations of any size. A system for detecting automatically whether a given URL is defaced has been proposed in [4]. The system operates as a cloud-based service and is designed for dynamic web content: it first builds a profile of the web page and then sends an alert whenever the page content deviates from that profile. This system is unable to detect hidden defacements because it must know the URLs to be monitored in advance.

Techniques for detecting phishing attacks can be subdivided in three categories: URL-based, host-based and content-based detection methods. Broadly speaking, URL-based approaches are faster and more scalable than the others since that they can classify a URL based on the URL itself, without collecting any further information.

An approach that relies only on URL-based features is proposed in [5]. The authors of the cited work use structural features of the URLs and some binary features indicating the presence of certain words in the URLs itself. This approach requires a fair amount of domain knowledge (for choosing the words corresponding to features) and, as such, it appears to be difficult to generalize.

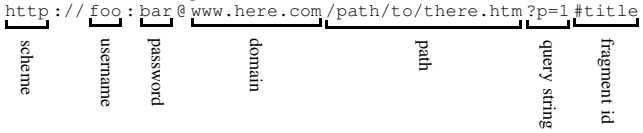
Other studies augment URL-based features by features extracted from the host where the corresponding document resides [6], [7], [8], [9]. The approach proposed in [6] uses lexical features, geographic properties and features extracted from `whois` queries: primary domain name, the registrar, the registrant, and the registration date. Authors of [7], [8], [9] use a larger set of host-based features such as `whois` queries, DNS information, geographic properties, connection speed and membership in blacklists; these features are used with online classification algorithms. In [8] the authors experiment with different classifiers (Support Vector Machine (SVM), logistic regression, Bayes), whereas in [7] the features are processed also using the confidence-weighted algorithm, passive-aggressive algorithm and the perceptron.

Approaches that belong to the content-based category [10], [11], [12] are more time consuming than the others, since they involve fetching and analyzing the full page content. The authors of [10] use a number of features coming from the HTML and JavaScript code found in the page in addition to URL and host based features: the obtained features set is analyzed with different classifiers. In [12] the features corresponding to a URL are extracted using a bag-of-word approach while the page content is compared to sets of files from previously confirmed phishing websites using MD5 hashes. The classification is performed using a confidence weighted algorithm and tests are conducted on daily batches of URLs simulating a model updated by a daily URL blacklist/whitelist feed. The approach proposed in [11] uses features extracted from the web page content (the presence of password fields and the external links frequency) in addition to URL and host based features with a proprietary machine learning algorithm: due to their nature, these content-based features fit the phishing URLs detection scenario, while they could not be appropriate for the defacement URLs detection scenario—a phishing page is carefully crafted to resemble genuine content, while a defacement page is usually very different from the original page.

III. Our approach

A URL (Uniform Resource Locator) is a string which identifies a web resource (Figure 1). We say that a URL is *hidden* if the corresponding page is hosted within a site without the administrator being aware of it. We say that a URL is *fraudulent* if the corresponding page is a defacement or a phishing attack (pages devoted to disseminating malware are beyond the scope of this work). The goal of the proposed method is to associate an input URL u with a boolean value which indicates if u is an *hidden fraudulent URL*.

Figure 1. The structure of a URL



We propose three increasingly more complex variants of the method. Each variant makes use of a superset of the information available to the previous variant. The first one, which we call *lexical*, uses only features extracted from the URL u itself. The second one, *lexical+headers*, augments those features with some of the headers obtained as response to an HTTP request for u. Finally, *lexical+headers+age*, uses also some of the headers obtained while fetching the home page of the domain of u—i.e., the web page identified by u without the path and subsequent components. In other words, lexical may be applied for classifying u without issuing any request for u and may thus be applied *offline*; lexical+headers requires one HTTP HEAD request for u; lexical+headers+age requires one HTTP HEAD request for u and another HEAD request for the home page of the domain of u.

Each variant requires a preliminary parameter calibration to be performed only once based on labelled data collected in a *training set*. The training set is transformed into a matrix F of *features*, with one row for each URL in the training set and one column for each feature analyzed—each variant analyzing a superset of the features analyzed by the previous one. We describe the three variants in the next sections.

A. Lexical

The lexical variant uses only the URL string itself, as follows (Figure 1 shows the structure of an example URL..). Let $U = \{u_1, \ldots, u_n\}$ be the training set and $L = \{l_1, \ldots, l_n\}$ the corresponding set of labels: $l_i =$ true if and only if u_i is an hidden fraudulent URL.

For the tuning, we first remove from each URL u_i every character up to the domain (included), and obtain a string p_i. Then, we extract the unigrams (i.e., character occurrences) from each p_i and obtain a matrix F of *features*, where $f_{i,j}$ is the number of occurrences of character c_j in p_i. Finally, we train a Support Vector Machine (SVM) on F using the labels of L. We use a third-degree polinomial kernel with cost parameter $C = 10$.

The classification of a unknown URL u is performed with the same steps as above, i.e.: (i) preprocess u to obtain a string p; (ii) compute the occurrences in p of the characters corresponding to the columns of F, obtaining a feature vector; (iii) apply the SVM to the feature vector.

B. Lexical+headers

The lexical+headers variant uses, in addition to the features of the previous variant, features extracted from the values of some of the HTTP response headers obtained when requesting the url U to be classified:

1) `Server`: name and version of the software running the server;
2) `Content-Type`: MIME type of the content of the named web-resource;
3) `Content-Lenght`: length of the response body in octets;
4) `X-Powered-By`: framework of the web application that produces the content of the web-resource (e.g., ASP.NET, PHP, JBoss);

In order to minimize the traffic, we issue HTTP HEAD requests instead of HTTP GET requests (while a GET asks for a resource, an HEAD asks for a response identical to the one that would correspond to a GET, but without the response body, i.e., without the actual resource).

The tuning of this variant proceeds as follows. For each $u_i \in U$, we perform a HEAD request to u_i and store each of the received header values $h_i^1, \ldots, h_i^4$ (in case the response does not contain the k-th header, we set the corresponding h_i^k is to the empty string). We pre-process the values for the `Server` and `X-Powered-By` headers so as to keep only the framework name and the major and minor version number (e.g., `Apache/2.2.22-12` becomes `Apache/2.2`). We then transform the header values in numerical features as follows. For each k-th header, we build a matrix F^k based on all the $v_1^k, \ldots, v_{n_k}^k$ observed values, as follows. F^k has a row for each URL and a column for each distinct header value: a matrix element $f_{i,j}^k$ of F^k is 1 if and only if $h_i^k = v_j^k$, 0 otherwise. In other words, F^k is a matrix of the n_k binary features corresponding to the observed values for the k-th header. Finally, the new features in F^1, F^2, F^3, F^4 are added to the original feature matrix F by joining all the columns of the five matrices.

The remaining part of the tuning step and the classification step are the same of the lexical variant.

C. Lexical+headers+age

The lexical+headers+age variant augments the previous features with the difference between the timestamps given by the values of the `Last-Modified` header obtained for the URL u and for the home page corresponding to u. These timestamps correspond to the creation in the tuning of two further columns in F, as follows.

For each $u_i \in U$, we extract the `Last-Modified` value (from the same response to the HEAD request performed for the previous variant) and store it in t_i as a date (in case the response does not contain the `Last-Modified`, we set $t_i = \emptyset$). Next, we remove from u_i the substring which starts from the path (included) and obtain d_i, which is the URL of the home page corresponding to u_i. Then, we perform a HEAD request to d_i, store the value of the `Last-Modified` header in t_i' and set $a_i = t_i - t_i'$ (in seconds). We also set a binary feature a_i' to 0, if both t_i and

t'_i were defined (i.e, $t_i \neq \emptyset$ and $t'_i \neq \emptyset$), or 1, if at least one of them was undefined. Finally, we add two new columns to F, given by the a_i and a'_i.

The rationale is that we try to exploit the information given by the *relative age* of the u resource, i.e, its last modification date compared to the home page last modification date. In other words, if the home page of the web site was modified long before the URL u under analysis, this could be a symptom of an hidden fraudulent URL.

IV. Experimental evaluation

A. Dataset

We assessed the effectiveness of our approach on two categories of hidden fraudulent URLs: (i) hidden phishing pages and (ii) hidden defacements. Due to the lack of publicly available datasets we collected, for each category, a set of real-world URLs as described below.

Concerning hidden phishing pages, we used the data provided by Phishtank[1]. Phishtank is a public web-based archive of phishing attacks: a user can report Phishtank of a possible attack by providing the phishing page URL. A team of voluntary experts may then verify the user's notification, marking the URLs as "valid phish". Moreover, Phishtank periodically verify that each phishing attack is actually still in place—i.e., if a page is served at the corresponding URL—and mark those URLs as "online". We composed a set U^P of about 7500 valid and online URLs extracted from Phishtank; we verified that each URL in U^P was still marked as valid and online for all the duration of our experimental evaluation.

Concerning hidden defacements, we used data provided by Zone-H[2]. The Zone-H Digital Attacks Archive is a public web-based archive of defacement attacks: users or attackers themselves report a URL of a defaced page to Zone-H; later, a human operator verifies and confirms the attack which is then published on the Zone-H web site. We composed a list U^D of about 2500 URLs extracted from Zone-H.

We assumed that both Phishtank and Zone-H are indeed authoritative with respect to URLs being fraudulent, i.e., we assumed that all URLs of U^P and U^D are fraudulent. A key ingredient of our problem is the ability of identifying fraudulent URLs that are hidden . For this reason, we defined 5 sets of URLs: (i) hidden fraudulent URLs: U^P_+ and U^D_+ (phishing and defacement category respectively); (ii) URLs of legitimate pages belonging to sites that host fraudulent URLs: U^P_- and U^D_- (phishing and defacement category respectively), (iii) URLs of legitimate pages belonging to trusted and (as far as we can tell) uncompromised web sites: U_-. In order to populate these sets we proceeded as follows.

First, we dropped from the set of fraudulent URLs U^P and U^D: (a) URLs whose domain is an IP address; (b) URLs whose path is empty or equal to `index.html`. The rationale for dropping these items is that they are not intended to be hidden. In the former case, we assume that the whole web site is involved in the attack (since it has no DNS name): in other words, it is not a legitimate web site to which an illegitimate content has been added. In the latter case, the attack is manifestly not hidden, since it resides at the root of the domain.

Second, we attempted to identify hidden URLs (i.e., URLs unknown to the administrators) by assuming that an URL is hidden if it is never reachable while crawling the site. Clearly, crawling an entire site is often not feasible. We hence marked an URL as hidden if it is not found by crawling the corresponding web site within up to the third level of depth. In order to justify this choice, in particular the choice of the third level of depth, we performed the following quantitative analysis. We selected a set W of 20 web sites extracted from the top 500 web sites ranking provided by Alexa[3]. We excluded from this selection: (i) web sites providing different content depending on whether the user is authenticated, (ii) social network web sites, (iii) search engines. Table I shows the 20 selected web sites. For each web site $w_i \in W$, we crawled the site up to the 10th level of depth and saved all the URLs obtained in the list U_-. We also determined, for each level l, the number of URLs $n_{i,l}$ found by crawling up to that level. Then, we computed the crawling coverage $C_{i,l} = \frac{n_{i,l}}{n_{i,10}}$ as the fraction of URLs found by crawling up to level 10 which were also found by crawling up to level l. Figure 2 shows crawling coverage, averaged across all 30 web sites, vs. the level l: it can be seen that the curve has a clear slope change at $l = 3$ and tends to be flat for highest values of l. In other words, crawling up to the third level is an acceptable compromise between coverage (which is, on the average, 88% for $l = 3$) and easiness of obtaining the data.

edition.cnn.com	www.steampunk.dk
www.bbc.com	www.weather.com
www.nytimes.com	www.godaddy.com
www.microsoft.com	www.nbcnews.com
www.whitehouse.gov	www.foxnews.com
www.adobe.com	www.bankofamerica.com
www.huffingtonpost.com	www.spiegel.de
espn.go.com	www.aweber.com
www.mediafire.com	www.chase.com
www.1advice.com	amazonaws.com

Table I
LISTS OF 20 DOMAINS USED TO SELECT THE MAXIMUM CRAWLING DEPTH.

We now describe how we populated the sets necessary for our evaluation. Concerning the sets U^P_- and U^D_- of legitimate pages belonging to sites that host fraudulent URLs, we procedeed as follows. For each $u_i \in U^P$, we

[1] http://www.phishtank.com
[2] http://www.zone-h.org
[3] http://www.alexa.com/topsites

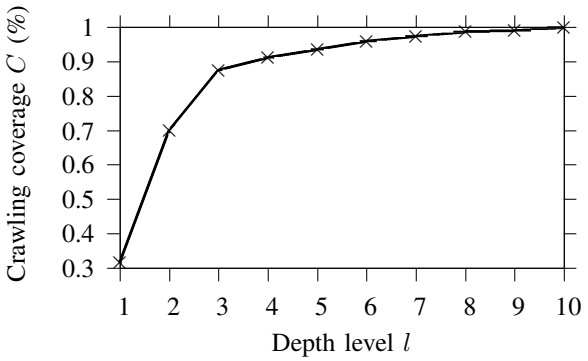


Figure 2. Crawling coverage C vs. depth level l.

crawled the corresponding domain d_i up to the third level of depth and added all the obtained URLs to the (initially empty) set U_-^P. That is, we (somewhat arbitrarily) assumed that all the pages in a crawling up to the third level are indeed legitimate. We repeated the same procedure on U^D and obtained U_-^D.

Concerning the sets U_+^P and U_+^D of hidden fraudulent pages, we procedeed as follows. For each $u_i \in U^P$, we added u_i to the (initially empty) set U_+^P if, and only if, it was not found in the crawl described above. We repeated the same procedure on U^D and obtained U_+^D.

Finally, the set U_- of legitimate pages belonging to trusted and (as far as we can tell) uncompromised sites, consisted of all the URLs found while crawling the 20 web sites in Table I up to the 10th depth level.

We collected this data during months July to November, 2012 and obtained 6564, 78388, 2144, 94370, 3713 URLS respectively in U_+^P, U_-^P, U_+^D, U_-^D and U_-. The dataset is publicly available on our lab web site[4].

B. Settings and results

We performed two suites of experiments in order to evaluate our approach effectiveness separately on the two hidden URLs categories, i.e., phishing and defacement.

For the purpose of the experimental evaluation, we built, for each category, a labeled and balanced dataset, as follows. For the phishing category, we set U_e^P to the set containing all U_+^P URLs, 3282 URLs randomly extracted from U_-^P and 3282 URLs randomly extracted from U_-; we also set the corresponding labels L^P accordingly. For the defacement category, we set U_e^D to the set containing all U_+^D URLs, 1072 URLs randomly extracted from U_-^D and 1072 URLs randomly extracted from U_-; we also set the corresponding labels L^D accordingly.

We assessed the our approach effectiveness in terms of the following performance indexes (for the sake of clarity, we here show index names only for the phishing category):

- false negative rate (FNR on U_+^P);
- false positive rate on U_-^P (FPR on U_-^P);
- false positive rate on U_- (FPR on U_-).
- accuracy, i.e., the ratio between correctly classified URLs and all the processed URLs;

For each variant of our method, we repeated 5 times the following procedure, collecting for each execution the four performance indexes explained before (for brevity, we describe the procedure for the phishing category only): (i) we randomly split U_e^P in a training set and testing set, 90% of URLs used for training and the remaining 10% for testing (we preserved the proportion of URLs coming from U_+^P, U_-^P and U_-), (ii) we trained the SVM classifier on the training set and, finally, (iii) we applied the trained classifier on each URL of the testing set.

Tables II and III report the performance indexes, averaged across the five repetitions, for the three variants of our method applied on the phishing and defacement categories, respectively.

The accuracy of our system, using the lexical+headers+age variant, is greater than 99% and 95% for defacement and phishing categories, respectively.

As expected, the off-line variant (lexical) has a lower accuracy (98% for defacement and 92% for phishing): on the other hand, this variant requires just less than a millisecond to classify an URL, whereas the other on-line variants require to perform one or two HTTP requests, which can result in several seconds of elapsed time.

Another key finding is that FPR on U_- is lower than 1% for both categories, when using the on-line variants of our method. Only for the phishing category FPR on U_-^P is slightly higher (about 4%): this result is somewhat justified because U_-^P are negative URLs belonging to compromised web sites and could hence resemble fraudulent URLs.

Concerning FNR, the experimental evaluation shows that it is higher for the phishing category (about 5%) than for the defacement category (less than 1%). The reason is because an attacker who puts in place a phishing attack will purposely shape all components of the attack (i.e., including the URL of the fraudulent page) so as to make it as much unnoticed as possible. An attacker which hides a defacement page will likely not care too much of whether the attack URL is easily detectable.

The time needed for the URL classification is smaller than 1 msec, 800 msec and 1600 msec respectively for lexical, lexical+headers and lexical+headers+age, on average. Note, however that the lexical variant can work off-line (i.e., without any access to the Internet). The actual computation time is < 1 msec for classifying an URL; 2 sec and 60 sec are required for the initial tuning, respectively on a training set of about 4300 defacement and about 13000 phishing URLs. We executed all our experiments on a machine powered with a quad-core Intel Xeon X3323 (2.53 GHz) and 2GB of RAM, with an high-speed academic connection

[4]The link is obscured to comply with the double blind review process.

Method	Accuracy (%)		FNR on U_+^P (%)		FPR on U_-^P (%)		FPR on U_- (%)	
	Avg.	Dev. Std.	Avg.	Dev. Std.	Avg.	Dev. Std.	Avg.	Dev. Std.
Lexical	92.50	0.62	9.80	0.85	5.20	0.84	4.93	1.21
Lexical+headers	95.34	0.48	4.93	1.05	4.38	0.62	0.79	0.70
Lexical+headers+age	95.57	0.37	4.87	1.06	3.98	0.52	0.73	0.73

Table II
RESULTS FOR THE PHISHING CATEGORY.

Method	Accuracy (%)		FNR on U_+^D (%)		FPR on U_-^D (%)		FPR on U_- (%)	
	Avg.	Dev. Std.	Avg.	Dev. Std.	Avg.	Dev. Std.	Avg.	Dev. Std.
Lexical	98.37	0.68	0.93	0.74	2.32	0.87	2.41	1.92
Lexical+headers	99.35	0.3	0.37	0.61	0.93	0.57	0.93	0.65
Lexical+headers+age	99.26	0.3	0.47	0.66	1.02	0.61	0.93	0.65

Table III
RESULTS FOR THE DEFACEMENTS CATEGORY.

to the Internet.

V. CONCLUDING REMARKS

Fraudulent creation of pages at URLs at which no page is supposed to exist has become an increasingly attractive and common kind of web intrusions. Detecting the existence of pages as such hidden URLs may be very difficult, because most users will never visit those URLs or observe any evident anomaly in the site content. Phishing campaigns coupled with attacks of this sort may be extremely dangerous and fool even tech-savvy users, because any page coming from an HTTPS-protected site appears as being legitimate.

We have proposed and evaluated an approach for the detection of hidden fraudulent URLs before actually fetching the corresponding page. The peculiarity of our proposal consists in not using any feature related to the domain part of the URL, i.e., the URL portion which identifies the host. Our proposal could be deployed in a variety of ways, either on end-user platforms or within a border web proxy as a layer for a defense in depth strategy.

The experimental results are very promising and have been obtained by a one-time training of our methods on a few thousands of samples. We speculate that they could be improved significantly by a one-time training with a much larger set of samples, e.g., in the order of millions, as routinely used by large service providers (a scale that we cannot presently afford).

REFERENCES

[1] A. Bartoli, G. Davanzo, and E. Medvet, "The reaction time to web site defacements," *Internet Computing, IEEE*, vol. 13, no. 4, pp. 52–58, 2009.

[2] N. Leontiadis and T. Moore, "Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade," *Proc. USENIX Security*, 2011.

[3] E. Sorio, A. Bartoli, and E. Medvet, "A look at hidden web pages in italian public administrations," in *Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on*, pp. 291–296, IEEE, 2012.

[4] A. Bartoli, G. Davanzo, and E. Medvet, "A framework for large-scale detection of web site defacements," *ACM Transactions on Internet Technology (TOIT)*, vol. 10, no. 3, p. 10, 2010.

[5] H. Huang, L. Qian, and Y. Wang, "A SVM-based Technique to Detect Phishing URLs," *Information Technology Journal*, 2012.

[6] A. Le and A. Markopoulou, "PhishDef: URL Names Say It All," in *Proceedings IEEE INFOCOM*, 2010.

[7] L. K. Saul, S. Savage, G. M. Voelker, and L. Jolla, "Identifying Suspicious URLs: An Application of Large-Scale Online Learning," in *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[8] J. Ma, L. Saul, S. Savage, and G. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," *Proceedings of the 15th ACM ...*, 2009.

[9] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Learning to detect malicious URLs," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, 2011.

[10] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages," in *World Wide Web Conference*.

[11] C. Whittaker, B. Ryner, and M. Nazif, "Large-Scale Automatic Classification of Phishing Pages," in *Network and IT Security Conference: NDSS 2010*, 2008.

[12] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing URL detection using online learning," *Proceedings of the 3rd ACM workshop on Artificial intelligence and security - AISec '10*, p. 54, 2010.