PhishScore: Hacking Phishers' Minds

Samuel Marchal*†, Jérôme François*‡, Radu State*, Thomas Engel*

* SnT - University of Luxembourg, Luxembourg – { samuel.marchal | radu.state | thomas.engel }@uni.lu
† TELECOM Nancy - University of Lorraine, France – samuel.marchal@univ-lorraine.fr
‡ INRIA Nancy Grand Est, France – jerome.francois@inria.fr

***Abstract*—Despite the growth of prevention techniques, phishing remains an important threat since the principal countermeasures in use are still based on reactive URL blacklisting. This technique is inefficient due to the short lifetime of phishing Web sites, making recent approaches relying on real-time or proactive phishing URLs detection techniques more appropriate. In this paper we introduce PhishScore, an automated real-time phishing detection system. We observed that phishing URLs usually have few relationships between the part of the URL that must be registered (upper level domain) and the remaining part of the URL (low level domain, path, query). Hence, we define this concept as intra-URL relatedness and evaluate it using features extracted from words that compose a URL based on query data from Google and Yahoo search engines. These features are then used in machine learning based classification to detect phishing URLs from a real dataset.**

I. Introduction

Phishing is currently one of the most lucrative cybercrime activities. Although accurately evaluating the financial loss cause by phishing is difficult, some surveys have been conducted, suggesting losses of several billion dollars every year. In 2007, Gartner Research estimated a $3.2 billion loss due to phishing scams [1]. Javelin Strategy & Fraud published a report [2] that identity theft led to a loss of $54 billion in 2009, mostly due to cybercrime.

Various techniques are used to perform phising attacks, ranging from *technical subterfuges* (DNS cache poisoning, e-mail spoofing, Web server takeover, etc.) to *social engineering*. In addition various goals are sought: data, money or credential stealing through fake Web sites, drive-by download of malware, etc. Despite this diversity, one common feature is the use of obfuscated URLs to misdirect users to fake Web sites or drive-by downloads.

Luring Internet users by making them click on rogue links that seem trustworthy is an easy task because of widespread credulity and unawareness. To cope with this threat, the best strategy is to prevent connection to phishing Web sites by the identification of phishing URLs. Phishing Web site short lifetime [3] makes the protracted process of reactive blacklisting based on user reports inefficient. In addition the use of different variations in URLs for the same phishing campaign [4] complicates the task of blacklisting, as blacklists must provide a perfect match for a URL. Hence real-time malicious URL detection is a better technique for defeating phishing.

In this paper, we propose an automated real-time URL phishingness rating system to protect users against phishing content: PhishScore. The underlying method targets identification of phishing URLs that are based on registered domains (malicious or not) that are not related to their targeted brand. To delude their victims, phishers blend many phishing keywords (famous brand, attractive words) into the remaining parts of the URL. Most Internet users are not aware of the DNS hierarchy. Seeing words like *paypal*, *ebay* or *visa* at any level of a URL will make them feel confident that the rogue link actually leads to the official Web site of these brands.

From observation of phishing URLs, we claim that there are few relationships between the registered domain and the rest of the URL. However, the words that compose the rest of the URL (low level domain, path, query) often have many interrelationships. Therefore, our approach evaluates the relatedness of words that compose a URL and highlights the differences between legitimate and phishing URLs. Previously existing solutions [5], [6], [7], [8], are not well suited to evaluating word similarity or relatedness for the Internet vocabulary. These tools, coming from the natural language processing field, usually have no entries for domain names and most of the words that compose a URL. We leverage search engine query data from Google and Yahoo to compute this relatedness.

Based on this, we define the term of intra-URL relatedness. We extract 12 features from a single URL which are input to machine learning algorithms to identify phishing URLs. Our technique is assessed on ground truth data of 96,018 URLs leading to a correct classification rate of 94,91%. Finally, a phishingness scores is computed for every single URL based on Random Forest classifier.

To summarize the major contributions of this paper:

– We introduce the concept of intra-URL relatedness depicting the relation between a registered domain and the words that compose the rest of a URL.

– We leverage search engine query data to establish relatedness between words and show that this is more suited to Internet vocabulary than existing methods.

– We propose new features based on intra-URL relatedness and build a machine learning based approach relying on these for distinguishing between phishing and non-phishing URLs.

The rest of the paper is structured as follows: we start by presenting URL obfuscation techniques in Section II. We introduce the search engine query data and the metrics used to calculate intra-URL relatedness in Section III. Section IV presents the datasets. PhishScore is assessed in Section V, both

Obf. Type	Example
Type I	http://school497.ru/222/www.paypal.com/29370274276105805/ http://paypal.com.eu.compte.client.update.condst.com.br/
Type II	http://www.quadrodeofertas.com.br/www1.paypal-com/encripted/ssl218 http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run
Type III	http://cgi-3.paypal-secure.de/info2/verikredit.html http://paypal-shopping.co.il/
Type IV	http://69.72.130.98/janaseva/https.paypal.com/uk/onepagepaypal.htm ftp://212.13.144.72/SERVICE/PayPal.com/security/alert/paypal.com
Type V	http://tiny.cc/clientID00858JD8 http://goo.gl/HQx5g

TABLE I
EXAMPLE OF OBFUSCATED URLS FOR THE DOMAIN *paypal.com*

for classification and scoring. Limitations of this technique are identified in Section VI, related works are discussed in Section VII and we conclude in Section VIII.

II. PHISHING URL OBFUSCATION

This paper assumes some knowledge about DNS organisation and operation; the reader is referred to [9], [10], [11] for the necessary background.

Phishers usually try to lure their victims into clicking on rogue URLs pointing to phishing sites or drive-by downloads. Different URL obfuscation techniques are used with the aim of hiding the real host, and particularly the registered domain, the only part of the URL that cannot be freely defined. If somebody wants to use a domain *mydomain.tld* and derive several URLs from it: *url1.mydomain.tld*, *url2.mydomain.tld/file*, he has first to register the domain *mydomain.tld* at a domain registrar, ensuring that it cannot be registered by anybody else. Assuming a phisher wants to trap PayPal users, he must use a *domain.tld* other than *paypal.com*, as this domain is already registered by PayPal Inc. The phisher must register a domain name *mydomain.tld* and try to deceive people by blending labels such as *paypal* into the rest of the URL: *login.mydomain.tld/paypal*.

A registered domain consists of two parts: a *main level domain* and a *public suffix*. A *public suffix* (or *ps*) is a domain name suffix under which an Internet user can register a name. It can be just a Top Level Domain like *.com*, *.org* or a combination of level domains like *.co.uk* or *.blogspot.com*. A *main level domain* (or *mld*) is the level domain preceding a *public suffix*. A registered domain is then: *mld.ps*. For instance in *www.paypal.com/login*, *com* is the *ps* and *paypal* is the *mld*.

The different obfuscation techniques consist of blending either the original domain name or phishing keywords into the remaining part of the URL. These keywords are usually the targeted brand, related services of the brand and other attractive words such as *secure*, *login*, *protect*, etc.

Assume a URL formed of a hostname with different level domain (*ld*), a path (*path*) and a query (*key=value*): *http://5ld.4ld.3ld.mld.ps/path1/path2/path3?key1=value1 &key2=value2*. The obfuscation often consists in blending keywords into the path, the query and the low level domain of the hostname (*5ld.4ld.3ld*). In the following we present the most used URL obfuscation techniques [12], with examples given in Table I for the domain *paypal.com*:

- **Type I: URL obfuscation with other domain:** The *mld.ps* is a real domain name, usually registered by the phisher, while the original domain being phished is part of the path, the query or the low level domain.
- **Type II: URL obfuscation with keywords:** Again the *mld.ps* is a real domain name, and the brand being phished and related words are part of the path, the query or low level domain.
- **Type III: Typosquatting domains or long domains:** the *mld.ps* of the URL is the domain being phished but misspelled, with letters or words missing or added, or the domain is pronounced the same way as the original but written differently. The targeted brand can also be combined with other words to create an unregistered domain.
- **Type IV: URL obfuscation with IP address:** the URL's hostname is replaced by an IP address and the brand being phished is part of the path or the query.
- **Type V: Obfuscation with URL shortener:** A URL shortening service is used to hide the name of the real host. Such URLs are not meaningful and are mainly used in phishing attacks targeting services that use this kind of short URL, like Twitter.

We focus on the identification of the four first types of URL obfuscation technique since our technique relies on natural language processing, which is clearly not suited to shortened URLs. The common feature of these obfuscated URLs is that the brand and some related terms are included in the path, the query and low level domain. These terms are related as they have relationships with the targeted brand and have no obvious relation with the *mld.ps* that is used for phishing. This is the opposite of what happens for a legitimate URL, where all the parts of the URL are normally related. To reveal this difference a relatedness analysis of the different part of a URL is performed.

III. Intra-URL Relatedness Analysis

The intra-URL relatedness is the quantification of the relatedness among the words composing the different parts of a URL and more precisely between the registered domain and the rest of the URL. Due to the limitations of existing relatedness calculation techniques, we leverage search engine query data to compute it.

A. URL Word Extraction

The examples of obfuscated phishing URL from Type I to IV highlight a global characteristic in URL obfuscation, namely that there is no relation between the *mld.ps* and the rest of the URL. To reveal this, we split the URL in the two parts that are presumed to have no relationship: extract the *mld.ps* and separate it from the rest. As the *ps* may be composed of multiple level domain, we use Public Suffix List[1] to identify it and then retrieve the immediately preceding level domain as the *mld*. For the rest of the URL, a split according to non-alpha-numeric characters is first performed. From extracted parts composed of several words such as *paypalitlogin* in *http://sezopoztos.com/paypalitlogin/us/...* we use a dictionary-based word splitter [13]. For instance, the three words *paypal*, *it* and *login* are extracted from *paypalitlogin* through this process.

Based on this splitting two sets are composed: one, called RD_{url} (for Registered Domain), consists just of two elements: $RD_{url} = \{mld, mld.ps\}$. The other, REM_{url} (for REMaining part), is composed of all extracted words from the URL except *mld.ps*. Given *http://sezopoztos.com/paypalitlogin/us/webscr.html?cmd=_login-run*, the following sets are extracted:

- $RD_{url} = \{sezopoztos, sezopoztos.com\}$
- $REM_{url} = \{paypal, it, login, us, web, src, html, cmd, login, run\}$

The *mld.ps* is not split like the other part to keep the *mld* unmodified, which can be composed of several words.

Assume a type III obfuscated URL such as *http://cgi-3.paypal-secure.de/info2/verikredit.html* . The word *paypal* would be an element of $RD_{phish} = \{paypal, secure, de\}$. If *http://cgi-3.paypal.de/info2/verikredit.html* is a real PayPal URL, we have $RD_{legit} = \{paypal, de\}$ and $RD_{legit} \cap RD_{phish} = \{paypal, de\}$. However with the proposed decomposition of *mld.ps* we have the two lists $RD_{phish} = \{$*paypal-secure,paypal-secure.de*$\}$ and $RD_{legit} = \{paypal, paypal.de\}$ giving $RD_{legit} \cap RD_{phish} = \emptyset$. Hence our proposed decomposition emphasizes the difference between the two domains.

Once the two sets are built, the next step is to evaluate the relatedness of their components. It is tempting to compute word similarity or word relatedness with existing tools such as Disco [7]. However this tool, even if efficient in most cases and especially for phishing domain names analysis as shown in our previous work [14], it is not necessarily suited to intra-URL relatedness computation.

B. Word Relatedness Evaluation Tools Shortcomings

WordNet [8] is a lexical database containing a collection of english language words. Given a word, WordNet can give a collection of related words. The limitation of this tool is that it is only based on English vocabulary that is likely to appear in an English dictionary, whereas Internet vocabulary includes several different languages and many words that are not contained in dictionaries.

Automated techniques and measures have also been developed to evaluate word relatedness. Latent Semantic Analysis (LSA) proposed by Landauer and Dumais [15] or Pointwise Mutual Information (PMI), introduced by Church and Hanks [5], are examples of these techniques. The Normalized Google Distance (NGD [6]) computes the semantic similarity between two words by querying the Google search engine for these words and counts the number of Web pages where they appear together and individually. Disco [7] relies on mutual information evaluation between two words based on corpora.

To prove the limitations of these existing tools, we tested whether two of them are able to find related words for a set of labels. WordNet and Disco are chosen since these are the only ones that are really usable through an interface. The testing set consisted of the RD_{url} extracted from a set of 94 URLs. These URLs come from PhishTank (described in Section IV-A) *i.e.* phishing URLs present in PhishTank blacklist are categorised according to the brand they target, when we made our evaluation 94 brands and associated URLs were present in this list. A subset of this test set is given in Table II and the result of the test for each tool is given in the two first rows of Table III. The numbers of *mld* and *mld.ps* for which the tested tools can give at least one related word are given in absolute value and percentage terms.

Neither WordNet nor Disco performs well on this test set. These only provide related words for less than 25% of *mld* and never match any *mld.ps*, although the brands and domain names tested are well known. In addition for the *mld*s that match a result, it is usually for brand that is also a meaningful word such as *live* or *visa*. The test proves that current word relatedness tools are not suited to the measure of intra-URL relatedness.

While creating a dedicated corpus to be used with existing methods would be helpful but challenging, word relatedness can be dynamically inferred from search engine query data, as shown in next section.

C. Search Engine Query Data

To perform the evaluation of intra-URL relatedness, we use search engine query data. The reason is that URL obfuscation is a social engineering lure. Phishing URLs target a brand, so clever phishers blend within them the brand and words that Internet users associate with the brand, such as a provided service: *payment* for PayPal. People generally use search engines to access these services. When they make a search, they type some keywords that are typically the brand or the domain name and the service needed like *paypal payment* or *hsbc.com on-line banking*. These word associations reflect

[1] http://publicsuffix.org/list/

Brand	***mld***	***mld.ps***
JPMorgan Chase	jpmorganchase	jpmorganchase.com
TAM Airline	tam	tam.com.br
Visa	visa	visa.com
Windows Live	live	live.com
Poste Italiane	poste	poste.it
Wells Fargo	wellsfargo	wellsfargo.com
Blizzard	blizzard	blizzard.com

TABLE II
SUBSET OF MOST PHISHING TARGETED BRANDS WITH *mld* & *mld.ps*

Tool	***#mld***	***%mld***	***#mld.ps***	***%mld.ps***
WordNet	20	21.3%	0	0%
Disco	23	24.5%	0	0%
Yahoo Clues	87	92.6%	68	72.3%
Google Trends	92	97.9%	76	80.9%
Total	94	-	94	-

TABLE III
NUMBER OF LABELS MATCHING AT LEAST ONE RELATED WORD FOR 4 TOOLS

the cognitive process of users searching for PayPal or HSBC. Consequently, such words are the ones phishers tend to blend into URLs to trap PayPal and HSBC customers [3].

Hence, mining search engine query data for word relatedness measurement is relevant in a phishing context. To achieve this goal, we use search engine query data from two top-ranked search engines: Google and Yahoo. Both offer services, that, given a term, provide some insights on requesting trends concerning it. These services are respectively *Google Trends*[2] and *Yahoo Clues*[3]. In the context of the paper we define a term t as a set of words w. $\{paypal\}$ and $\{paypal, login, secure\}$ are two examples of terms.

Google Trends shows the relative interest of Google users over time in a term. It depicts the geographic interest for this term and provide related terms according to users' related searches. Google Trends provides the top ten related searches over time as well as the ten rising related searches namely those on which interest has increased recently. This allows us to gather up to twenty related terms for one given term.

Yahoo Clues provides the same kind of services as Google Trends. It offers an insight into the search flows, the terms requested just before (5 terms) and after (5 terms) a term. Like Google Trends it also provides a set of related searches, but no rising searches.

Combining both sources can give up to forty related terms for one given term. A result for the queried term $\{paypal\}$ for both tools Google Trends and Yahoo Clues is given in Table IV. The ability of these tools to find related words for phishing targeted *mld* and *mld.ps* is highlighted in Table III. Both tools were tested on the same set of terms used for WordNet and Disco, giving the results in rows 3 and 4. They perform better, with Google Trends being the best at finding related words. However both provide match results for more than 90% *mld* and 70% *mld.ps*, much more than usual similarity evaluation tools tested earlier.

Having a URL *url* and the extracted sets RD_{url} and REM_{url}, Google Trends and Yahoo Clues are automatically requested for each element of the two sets. We define $Term_w$, as the set of terms resulting from the requests of the word w in both Google Trends (related & rising) and Yahoo Clues (related & requests). A subset of $Term_{paypal}$ is given in Table IV with $Term_{paypal} = \{\{paypal, account\}, \{paypal, login\}, ...\}$.

We define four sets of words built from a URL *url*: $REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$.

$REL_{set}(url)$ consists of all the words *related* to the words of set *i.e.* words included in terms that are results of requests for elements of set. Here set is either RD_{url} or REM_{url}. The formulas for these sets are given in Equations (1) and (2).

$$REL_{rd}(url) = \{w \in t \mid t \in Term_{w'}, w' \in RD_{url}\} \tag{1}$$

$$REL_{rem}(url) = \{w \in t \mid t \in Term_{w'}, w' \in REM_{url}\} \tag{2}$$

$AS_{set}(url)$ is the set of words that are *associated* with the words of *set i.e.* the words that appear in a common single term. Assuming a term t composed of three words $\{w_1, w_2, w_3\}$, there is a symmetric *association* relationship between w_1 and w_2, w_1 and w_3, w_2 and w_3. The two sets $AS_{rd}(url)$ for RD_{url} and $AS_{rem}(url)$ for REM_{url} are defined in Equations (3) and (4) respectively.

$$AS_{rd}(url) = \{w \in t \mid \exists w' \in RD_{url}, w' \in t, w' \neq w\} \tag{3}$$

$$AS_{rem}(url) = \{w \in t \mid \exists w' \in REM_{url}, w' \in t, w' \neq w\} \tag{4}$$

These four sets are extracted to quantify the relationship between and inside each set RD_{url} and REM_{url}. Assume a URL *www.paypal.com/login*:
$RD_{www.paypal.com/login} = \{paypal, paypal.com\}$. Consider the subset of three terms resulting from querying *paypal* in Google Trends (*rising*) given in Table IV: $Term_{paypal} = \{\{amazon, paypal\}, \{paypal, fees\}, \{ebay, uk\}\}$
We have:
$REL_{rd}(www.paypal...) = \{amazon, paypal, fees, ebay, uk\}$
$AS_{rd}(www.paypal...) = \{amazon, fees\}$

D. Feature Calculation

Based on the sets defined in the previous subsection, we introduce 12 features characterising intra-URL relatedness and URL popularity. The popularity criteria is based on the search count for components of a URL (registered domain, *mld*, etc.). These features are described in Table V.

The features 1-6 define intra-URL relatedness by calculating the Jaccard index pairwise between the four sets defined in Section III-C ($REL_{rd}(url)$, $REL_{rem}(url)$, $AS_{rd}(url)$ and $AS_{rem}(url)$). The Jaccard index is a long-established metric used to calculate similarity and diversity between two sets A

[2] http://www.google.com/trends/

[3] http://clues.yahoo.com/analysis

Google related	**Google rising**	**Yahoo related**	**Yahoo requests**
$\{paypal, account\}$	$\{amazon, paypal\}$	$\{bill, me, later\}$	$\{paypal, login\}$
$\{paypal, login\}$	$\{paypal, fees\}$	$\{netspend\}$	$\{paypal.com\}$
$\{paypal, credit, card\}$	$\{ebay, uk\}$	$\{suntrust\}$	$\{paypal, buyer, credit\}$
$\{paypal, email\}$	$\{paypal, login\}$	$\{regions\}$	$\{paypal, customer, service\}$

TABLE IV
EXAMPLE OF TERM RESULTS FROM GOOGLE TRENDS AND YAHOO CLUES FOR $\{paypal\}$

	Features	**Description**
1	$J_{RR} = \frac{\lvert REL_{rd}(url) \cap REL_{rem}(url)\rvert}{\lvert REL_{rd}(url) \cup REL_{rem}(url)\rvert}$	Jaccard index b/w $REL_{rd}(url)$ and $REL_{rem}(url)$
2	$J_{RA} = \frac{\lvert REL_{rd}(url) \cap AS_{rem}(url)\rvert}{\lvert REL_{rd}(url) \cup AS_{rem}(url)\rvert}$	Jaccard index b/w $REL_{rd}(url)$ and $AS_{rem}(url)$
3	$J_{AA} = \frac{\lvert AS_{rd}(url) \cap AS_{rem}(url)\rvert}{\lvert AS_{rd}(url) \cup AS_{rem}(url)\rvert}$	Jaccard index b/w $AS_{rd}(url)$ and $AS_{rem}(url)$
4	$J_{AR} = \frac{\lvert AS_{rd}(url) \cap REL_{rem}(url)\rvert}{\lvert AS_{rd}(url) \cup REL_{rem}(url)\rvert}$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rem}(url)$
5	$J_{ARrd} = \frac{\lvert AS_{rd}(url) \cap REL_{rd}(url)\rvert}{\lvert AS_{rd}(url) \cup REL_{rd}(url)\rvert}$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rd}(url)$
6	$J_{ARrem} = \frac{\lvert AS_{rem}(url) \cap REL_{rem}(url)\rvert}{\lvert AS_{rem}(url) \cup REL_{rem}(url)\rvert}$	Jaccard index b/w $AS_{rem}(url)$ and $REL_{rem}(url)$
7	$card_{rem} = \lvert REM_{url}\rvert$	number of words in REM_{url}
8	$ratio_{Arem} = \frac{\lvert AS_{rem}(url)\rvert}{\lvert REM_{url}\rvert}$	ratio of associated words for words in REM_{url}
9	$ratio_{Rrem} = \frac{\lvert REL_{rem}(url)\rvert}{\lvert REM_{url}\rvert}$	ratio of related words for words in REM_{url}
11	$mld_{res} = \begin{cases} 0 & \text{if } \lvert Term_{mld}\rvert = 0 \\ 1 & \text{else} \end{cases}$	whether there is search engine results or not for the *mld* of the URL
11	$mld.ps_{res} \begin{cases} 0 & \text{if } \lvert Term_{mld.ps}\rvert = 0 \\ 1 & \text{else} \end{cases}$	whether there is search engine results or not for the *mld.ps* of the URL
12	$ranking$	Alexa ranking for $mld.ps$

TABLE V
URL FEATURE DESCRIPTIONS

and B. The closer $J(A, B)$ is to 1 the more similar are A and B. These six features quantify the relatedness between the two parts of the URL (*mld.ps* and the rest) through J_{RR}, J_{RA}, J_{AA} and J_{AR}, as these compute Jaccard indexes between sets extracted from different parts (RD_{url} and REM_{url}). These also measure the relatedness inside each part through J_{ARrd} and J_{ARrem}, as these features are calculated from sets extracted from the same part of a URL.

Features 7-12 reflect the popularity of a URL and its components with the number of words that compose it ($card_{rem}$) and the number of related and associated words found in search engine query data based on these words with $ratio_{Arem}$ and $ratio_{Rrem}$. These two features are weighted by $card_{rem}$. Features $mld.ps_{res}$ and mld_{res} represent the popularity of the registered domain by giving boolean values describing whether the $mld.ps$ and mld match results while queried in Google Trends and Yahoo Clues. The final feature (*ranking*) is the ranking of the *mld.ps* according to the Alexa[4] Web site ranking list. Alexa gives a ranking for the top 1,000,000 most visited Web sites; if a particular *mld.ps* is not in the list, the value 10,000,000 is considered.

Features 10, 11 and 12 can be considered as relying on the reputation of a domain and not on the intra-URL relatedness. Even if features 10 and 11 are new — *ranking* has been used already in state of the art work — we compare in Section V classification results with and without these three features to assess the relevancy of intra-URL relatedness features.

IV. DATASETS

To assess the ability of the proposed feature set to be used in supervised classification, we use two datasets. One of these is a malicious dataset, the *phishing dataset*; the other is the *legitimate dataset*. These sets are composed from different sources — as in several phishing detection work [12], [16], [17] — already used in [18], [19].

A. Phishing Dataset

We used PhishTank to build a phishing dataset. PhishTank[5] is a collaborative project to which people can submit phishing e-mails and Web sites. Suspected phishing URLs are further checked by several people before being confirmed as malicious and added to a blacklist. PhishTank provides lists of valid and active phishing URLs.

We downloaded this list on a daily basis between October 11th and November 10th, 2012 and built a phishing ground truth dataset of 53,089 unique URLs. URLs consisting only of *mld.ps*, *www.mld.ps* or IP addresses without path or query were discarded because it is impossible to calculate the intra-URL relatedness for such URLs, as $REM_{url} = \emptyset$. In addition we

[4] http://www.alexa.com/

[5] http://www.phishtank.com/

already addressed the identification of such phishing domains in [14]. After this selection we had 48,009 extended phishing URLs in the *phishing dataset* meaning less than 10% phishing URLs discarded.

B. Legitimate Dataset

To provide additional learning instances for legitimate URLs, we selected URLs from the Open Directory Project (DMOZ[6]). DMOZ is a directory of the Web containing more than two million URLs. We first discarded URLs consisting only of *mld.ps* or *www.mld.ps*, as for the *phishing dataset*. Then a uniform random selection was made on the rest to keep 48,009 legitimate URLs.

We constructed a balanced dataset (half malicious/half legitimate) of ground truth data composed of 96,018 URLs. We acknowledge that a half/half ratio for phishing and legitimate URLs does not reflect real world repartition. However this dataset is used to assess the efficiency of the search engine query data and the features extracted therefrom, in distinguishing phishing from legitimate URLs through ten-fold cross-validation. And as presented in [20], imbalanced dataset in cross-validation provides misleading results. This URL set with extracted features is publicly available for research purpose[7].

V. Phishing URL Detection

To automatically detect phishing URLs, we use supervised classification techniques. We build a feature vector matrix from the dataset presented in previous section. Each feature vector is composed of 12 elements, namely the 12 features described in Section III-D. The predicted variable is 0 for a legitimate URL and 1 for a phishing URL. This gives a matrix of 96,018 feature vectors representing the 96,018 URLs of the testing dataset.

A. URL Classification

Since there is a wide range of supervised classification algorithms, we assessed our dataset according to several classifiers using Weka [21]. Seven classifiers were tested divided between tree-based (Random Tree, Random Forest, C4.5, LMT) rule-based (PART, JRip) and function-based (SVM). The classification was made without parameters tuning through a ten-fold cross-validation as a first step to select the most promising approach. Results for accuracy, true positives and true negatives are given in Figure 1 for each classifier. For sake of clarity we define for URLs:

- Phishing classified as phishing: true positives (TP) and $TP_{rate} = \frac{TP}{TP+FN}$
- Legitimate classified as phishing: false positives (FP) and $FP_{rate} = \frac{FP}{TN+FP}$
- Legitimate classified as legitimate: true negatives (TN) and $TN_{rate} = \frac{TN}{TN+FP}$
- Phishing classified as legitimate: false negatives (FN) and $FN_{rate} = \frac{FN}{TP+FN}$

and the accuracy: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

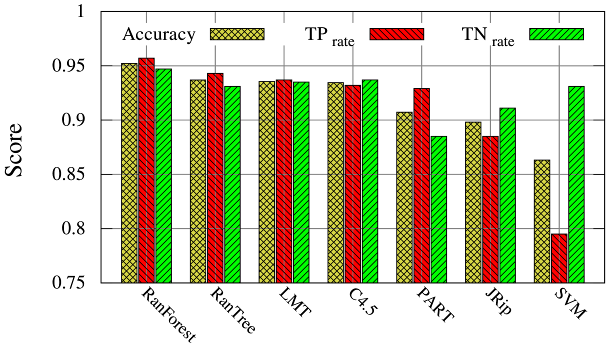

Fig. 1. Classification results for 7 classifiers

Among the tested classifiers, SVM yields the worst accuracy (86.31%) while being efficient in identifying legitimate URLs (93.1%). Rule-based classifiers have approximately the same performance (around 90%) with disproportionate true positives and true negatives. The best performers are tree-based classifiers, with Random Forest, correctly classifying 95.22% of URLs, being the best.

Hence, Random Forest is selected for classification. Random Forest [22] is a classification method that creates a multitude of decision trees during training. During prediction, it outputs a *hard decision* for the class of an instance as the class that has been predicted by the most individual trees. However a *soft prediction* can also be deduced from the combination of results given by individual trees. This *soft prediction* is bounded on $[0, 1]$ and gives a confidence score for the prediction. It is then compared to a discrimination threshold to give the *hard decision*.

We tuned the parameters of Random Forest training in order to achieve better classification. The number of trees to be generated during training was set to 100. The ROC (Receiver Operating Characteristic) curve describing the classification results for the tuned classifier in true positive rate and false positive rate is illustrated in Figure 2. The ROC curve typically corresponds to the variation of true positives and false positives while varying the discrimination threshold from 0 to 1. To minimize the number of legitimate URLs classified as phishing (false positives) we adjust the discrimination threshold from 0.49 (the value giving the best accuracy) to 0.76. This reduces the accuracy from 95.66% to 94.91% but also decreases the FP_{rate} from 4.13% to 1.44%.

The detailed classification metrics for the Random Forest algorithm with a 0.76 discrimination threshold are given in Table VI. The two first columns represent the rate of well-classified and misclassified instances for each class: TP_{rate}, FP_{rate}, FN_{rate} and TN_{rate}. The $Precision$ corresponds to the ratio of phishing URLs classified as phishing with respect to the total URLs classified as phishing such that: $Precision = \frac{TP}{TP+FP}$. The *F-measure* is defined as:
F-measure $= 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$, where $Recall = TP_{rate}$

To show the relevancy of intra relatedness features, we classified with different features the set of URLs. Using only

[6] http://www.dmoz.org/

[7] http://secan-lab.uni.lu/images/stories/samuel_marchal/urlset.csv

Class	Class. as phish.	Class. as leg.	*Precision*	*F-measure*	*Accuracy*
Phishing	91.27% (TP)	8.73% (FN)	98.44%	94.72%	94.91%
Legitimate	1.44% (FP)	98.56% (TN)			

TABLE VI
DETAILED CLASSIFICATION RESULTS FOR RANDOM FOREST (THRESHOLD = 0.76)

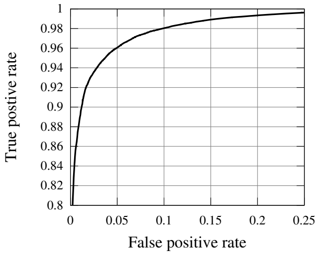

Fig. 2. ROC curve for Random Forest classification

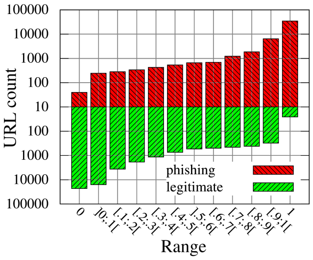

Fig. 3. Phishing and legitimate URL partition according to rating ranges

features 1-9 for classification yields an accuracy of 93.48% whereas using reputation based features 10-12 yields 83.97%. Feature 12 ($ranking$) and other state of the art features are not sufficient to distinguish between phishing and non-phishing URLs alone. However we show that the proposed feature set yields good results in doing this task. In addition, combining the new proposed features with reputation based features can lead to an improvement in the classification accuracy making this work complementary to the state of the art.

Even though this technique, which gives a *hard decision* for URL class, is proved efficient, correctly classifying 94.91% of URLs with only 1.44% of legitimate URLs classified as malicious, we further leverage machine learning to build a reputation system.

B. URL Rating

The *soft prediction* value provided by Random Forest is defined on the range $[0; 1]$. In the previous section a discrimination threshold was fixed to give a *hard decision* on the phishingness degree of a URL. However *soft prediction* values are not uniformly distributed over the range $[0; 1]$ and some sub-ranges of values may be more suitable to providing a highly reliable decision on the phishingness of a URL. Hence, we analysed the *soft prediction* distribution regarding phishing or legitimate URLs.

The *soft prediction* range of value $[0; 1]$ is divided in 12 sub-ranges, two being the exact value 0 and 1 and the ten remaining being ranges of length 0.1: $]0; 0.1[$, $[0.1; 0.2[$, ... , $[0.9; 1[$. The *soft prediction* provided by the tuned Random Forest was computed for all 96,018 URLs of the dataset through a ten-fold cross-validation. We counted the URLs having a score belonging in each sub-range. Figure 3 depicts this count according to the set (phishing at the top / legitimate at the bottom) the URLs come from. The 12 different sub-ranges are on the x-axis and the URL count is on the y-axis in a log scale centered on 10 and increasing in both directions for each class (phishing up/legitimate down).

We can observe that most of the URLs are grouped in each extremity of the range and mostly in the sub-ranges 0, $]0; 0.1[$, $[0.9; 1[$ and 1, which contain a total of 80,630 URLs out of 96,018. In addition the middle values of *soft prediction* have few of either kind of URLs, usually less than 1,000. This confirms that the *soft prediction* is not uniformly distributed over its range of definition. Considering the two extreme values, very few phishing URLs (40) have the score 0, whereas 22,863 legitimate URLs do. The same happens for a *soft prediction* of 1 where 34,790 phishing URLs have this score against only 26 legitimate. Given that 0 corresponds to legitimate and 1 to phishing, we are able to classify 60.11% of the dataset (57,719 URLs) with an accuracy of 99.89%. URLs getting a *soft prediction* of 0 or 1 are very likely to be either legitimate or phishing URLs respectively. This proves that some ranges of *soft prediction* values are more suited to making a reliable prediction. If we extend the analysis to the range $[0; 0.1]$, it contains 38,741 legitimate URLs and only 288 phishing ones. The range $[0.9; 1]$ is composed of 41,260 phishing URLs and 341 legitimate URLs. Considering these two sub-ranges, these contain 83.97% of the testing dataset and their components are correctly identified as legitimate or phishing with an accuracy of 99.22%.

The *soft prediction* can be used as a *risk score* for a URL. The closer it is to 1, the higher the risk; the closer to 0 the safer the URL. PhishScore automatically provides a phishingness score for requested URLs to inform users of risks. We have demonstrated that such a rating system is reliable in 99.22% of the cases for most of the URLs (83.96%).

While performing our experiments, we timed the process from labels extraction, requesting search engines, features computation to classification decision. It took around 112 hours for the set of 96,018 URLs on a single machine (Intel Core i7 processor and 4Gb memory). This gives an average time per URL of 4.2 seconds, most of the time being taken by the requests to search engines. Direct access to search engine query data would highly decrease this time making the delay introduced by this method shorter.

VI. DISCUSSION

This paper introduces PhishScore a system relying on search engine query data for phishing URL identification. This technique has some limitations that we identify in this section.

Our technique is not applicable to all types of obfuscated URLs as described in Section II. URLs composed of only a malicious domain, URLs based on shortening services or URLs algorithmically generated are kind of malicious URLs that can bypass our detection technique. This kind of URLs and malicious domains are however widely used in botnet communication (C&C) or spamming activities [23], such activities does not rely on a social engineering process as phishing does. The main part of URLs used for phishing are meaningful and composed of many terms [3], this is why our technique is relevant in a phishing context.

A limitation of the implementation is that data publicly available through Google Trends and Yahoo Clues is limited. For each requested term only the ten related most popular terms are returned by these tools. Related terms less requested by search engine users do not appear in results while being relevant for intra-URL relatedness computing. For the same reason, some unpopular terms blended in URLs do not match any results. The reason is that Google and Yahoo do not provide data that is not representative enough *i.e.* for terms that are not requested enough by their users. These facts limit the accuracy of intra-URL relatedness computing and is one of the reason why extra features such as *ranking* are included in the feature set. A full access to Web search logs would highly improve the relevancy of intra relatedness metrics and, as a result, the classification performance as well. Despite this limited access to data, the results presented in this paper provide strong hints regarding the relevancy of using search engine query data for phishing detection.

A last issue of using on-line tools for inferring intra-URL relatedness is the delay implied by multiple HTTP requests making real-time analysis difficult. As presented in Section V the average processing time per URL is 4.2 seconds. A solution to speed up the process would be again direct access to search engine query data or temporary local cashing.

VII. Related Work

For most related work the datasets used for assessment and the implemented techniques of phishing URL detection are not publicly available, making quantitative comparisons impossible.

In recent years, many techniques have been developed to cope with phishing and have focused on the real-time identification of this threat. One approach is to compare the content of presumed phishing Web pages with the original Web page being phished as in [24], [25], [26], [27]. The main shortcoming of such a method is that the site being phished must be first identified. Another is that this approach is limited to rogue Web sites which is just one of several types of phishing (*i.e.* drive-by download attacks). PhishScore relying only on URL analysis covers a larger scope than the latter.

Passively captured DNS traffic is used to recognise malicious domains in [28], [29]. The technique relies on machine learning algorithms applied to DNS-based features. The limitation of this approach when applied to phishing is that it identifies malicious domains instead of malicious URLs. For obfuscated phishing using URLs based on popular domain names, it is inefficient.

Consequently automated techniques to identify phishing URLs have been developed. Most rely on the extraction of phishing heuristics based only on the URL components. Features such as the length of the level domain, the path, the tokens, and the number of tokens at each level are considered in [18]. Some domain name-related information like ranking, WHOIS information, AS number, blacklisted status, etc. are used in [12], [17], [30]. A lexical analysis is performed to create binary features from each label observed in the URL in [12], [17], [18], [30]. In these techniques, label extraction is performed by splitting according to basic separators (/, ., =, -, etc.) whereas our method is more sophisticated [13]. In addition these approaches need previous knowledge about the exact labels being used in URLs. PhishScore although relying on labels that compose URLs, only computes from these labels and analyses numerical relatedness metrics. This relatedness can be calculated from previously unseen labels as long as these appear in search engine query data. Moreover none of these methods consider the semantic dimension of labels composing URLs as we do.

More predictive approaches have been developed to cope with phishing. In [31], a tailor made blacklist suited to single machine is proposed, this blacklist is built based on the machine's logs and historical attacks from other machines that are considered as similar. In [19], several phishing URLs are grouped according to common pattern in order to extract a common regular expression. Then new potential phishing URLs are generated according to the variable part of regular expressions.

Recently some solutions considering the semantic dimension of phishing attacks have been proposed. In [32], the content of phishing Web pages is mined. A semantic concept is extracted from every sentence composing the Web page. These concepts are then compared with templates learned from known phishing pages through a machine learning algorithm to determine if the Web page is a phishing one. Approximately the same technique is used in [33] to block phishing e-mails based on semantic content analysis. The common aspect with our approach is the leveraging of semantic information for phishing detection, a concept close to word relatedness. However previous research targets phishing e-mails and Web pages. Semantic relatedness analysis is performed on such content with existing similarity metrics, whereas in our work it is applied to URLs with new similarity metrics based on search engine query data. Our work is also complementary to [14], where phishing domain names likely to be registered by phishers are generated in a predictive process based on a natural language model to build predictive blacklists. Here we focus on real-time identification of full phishing URLs based on machine learning and new features, regardless they are based on malicious or legitimate domains.

Search engine query data has already been used for the analysis of search interests over time [34]. By monitoring the variation of interest for terms related to influenza, Ginsberg et al. [35] estimate the magnitude of flu infection for a given geographic region. In [36], Web search logs are used to improve search engine results. However, to the best of our knowledge we are the first to use this data for the purpose of word-relatedness evaluation.

VIII. Conclusion and Future Work

This paper introduces PhishScore, an efficient phishing URL detection system relying only on URL lexical analysis. The approach is based on the intra-URL relatedness. This relatedness reflects the relationship among the words blended into a URL and particularly into the part of the URL that can be freely defined and the registered domain. We leverage search engine query data in order to extract 12 features from a URL characterizing its intra relatedness and its popularity. The proposed features were used in supervised classification on a ground truth dataset of 96,018 phishing and legitimate URLs. This experiment yielded a classification accuracy of 94.91% with a low false positive rate of 1.44%. This experiment was extended to introduce a URL rating system, PhishScore, to dynamically compute a *risk score* for URLs. The *risk score* on the testing dataset is able to correctly identify 99.22% of the legitimate and phishing URLs for 83.97% of the dataset.

Future work will consist in merging the technique proposed in [14], which is complementary to that introduced in this paper, to create a phishing detection system with a larger scope of action. Due to delay issues the applications for real time phishing URLs are limited. A solution to apply this technique is as a centralized phishing e-mail detection system. Every link/URL embedded in incoming e-mail can be extracted and analysed by PhishScore to rate the risk of an e-mail before forwarding it to mail clients as a spam filter does.

Acknowledgements.

This work is supported by the Fonds National de la Recherche, Luxembourg (Project ID: 3967419).

References

[1] "Gartner survey shows phishing attacks escalated in 2007," Gartner Research, Tech. Rep., 2007. [Online]. Available: http://www.gartner.com/newsroom/id/565125

[2] "2010 Identity Fraud Survey Report," Javelin Strategy & Research, Tech. Rep., 2010.

[3] "Global Phishing Survey: Trends and Domain Name Use," APWG, Tech. Rep. 2H2011, 2012.

[4] "Phishing Activity Trends Report," APWG, Tech. Rep. 3rd Quarter 2012, 2013.

[5] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[6] R. L. Cilibrasi and P. M. Vitanyi, "The Google similarity distance," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 3, pp. 370–383, 2007.

[7] P. Kolb, "Disco: A multilingual database of distributionally similar words," in *Proceedings of KONVENS-2008*, 2008.

[8] G. A. Miller *et al.*, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[9] P. Mockapetris, "RFC 1034: Domain Names - Concepts and Facilities," 1987.

[10] ——, "RFC 1035: Domain Names - Implementation and Specification," 1987.

[11] P. Mockapetris and K. Dunlap, "Development of the domain name system," in *Proceedings of the 1988 ACM SIGCOMM*. IEEE Computer Society, 1988.

[12] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the 2007 ACM workshop on recurring malcode*. ACM, 2007, pp. 1–8.

[13] T. Segaran and J. Hammerbacher, *Beautiful data: the stories behind elegant data solutions*. O'Reilly Media, Incorporated, 2009.

[14] S. Marchal, J. François, R. State, and T. Engel, "Proactive discovery of phishing related domain names," in *Research in Attacks, Intrusions, and Defenses*. Springer, 2012, pp. 190–209.

[15] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge," *Psychological Review; Psychological Review*, vol. 104, no. 2, p. 211, 1997.

[16] M. Khonji, Y. Iraqi, and A. Jones, "Lexical url analysis for discriminating phishing and legitimate e-mail messages," in *2011 International Conference for Internet Technology and Secured Transactions (ICITST)*, 2011, pp. 422–427.

[17] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: an application of large-scale online learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 681–688.

[18] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: URL names say it all," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 191–195.

[19] P. Prakash, M. Kumar, R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *Proceedings of INFOCOM*. IEEE, 2010, pp. 1–5.

[20] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 49–57, 2010.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[22] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. ACM, 2010.

[24] T.-C. Chen, S. Dick, and J. Miller, "Detecting visually similar web pages: Application to phishing detection," *ACM Transactions on Internet Technology*, vol. 10, no. 2, 2010.

[25] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proceedings of the 4th international conference on security and privacy in communication netowrks*. ACM, 2008, p. 22.

[26] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 571–580.

[27] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 639–648.

[28] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in *19th Usenix Security Symposium*, 2010.

[29] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive DNS analysis," in *Proceedings of NDSS*, 2011.

[30] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009.

[31] J. Zhang, P. Porras, and J. Ullrich, "Highly predictive blacklisting," in *Proceedings of the 17th conference on Security symposium*. USENIX Association, 2008.

[32] J. Zhang, Q. Li, Q. Wang, T. Geng, X. Ouyang, and Y. Xin, "Parsing and detecting phishing pages based on semantic understanding of text," *Journal of Information & Computational Science*, no. 9, pp. 1521–1534, 2012.

[33] V. Ramanathan and H. Wechsler, "phishGILLNET phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training," *EURASIP Journal on Multimedia and Information Security*, vol. 2012, no. 1, pp. 1–22, 2012.

[34] J. Rech, "Discovering trends in software engineering with Google Trend," *ACM SIGSOFT Software Engineering Notes*, vol. 32, no. 2, pp. 1–2, 2007.

[35] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant, "Detecting influenza epidemics using search engine query data," *Nature*, vol. 457, no. 7232, pp. 1012–1014, 2008.

[36] H. Liu, J. He, Y. Gu, H. Xiong, and X. Du, "Detecting and tracking topics and events from web search logs," *ACM Trans. Inf. Syst.*, vol. 30, no. 4, pp. 21:1–21:29, Nov. 2012.