Modeling Content from Human-Verified Blacklists for Accurate Zero-Hour Phish Detection

Guang Xiang, Bryan A. Pendleton, and Jason Hong
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213-3891, USA
Email: {guangx, bpendlet, jasonh}@cs.cmu.edu

Abstract

Phishing attacks are a significant security threat to users of the Internet, causing tremendous economic loss every year. Past work in academia has not been adopted by industry in part due to concerns about liability over false positives. However, blacklist-based methods heavily used in industry are slow in responding to new phish attacks, and tend to be easily overwhelmed by phishing techniques such as fast-flux and the proliferation of toolkits.

In this paper, we present the design and evaluation of two blacklist-enhanced content-based algorithms. The key insight behind our algorithms is to leverage existing human-verified whitelists and blacklists, and relax them via probabilistic methods to attain high true positive rates while maintaining extremely low false positive rates. Comprehensive experiments over a diverse spectrum of data sources show that our approach currently achieves a false positive rate of 0.0434% *with a true positive rate of* 87.42%. *Our algorithms are able to adapt quickly to new phishing attacks by incremental retraining, and present a new framework that will generalize to evolving attacks.*

1. Introduction

Phishing is a form of identity theft, in which phishers build replicas of target websites and lure unsuspecting victims to give away their financial information like passwords, personal identification numbers (PINs), etc. The annual Internet crime report of IC3 [1] revealed that through the year of 2007, Internet crimes had caused an unprecedented loss of $\$239.09$ million dollars. One particularly infamous group, known as the "rock phish gang", uses phish toolkits to create a large number of unique phishing URLs, which puts additional pressure on the timeliness and accuracy of blacklist-based anti-phishing techniques.

Generally, mainstream phish detection methods employ a domain/URL blacklist, the creation of which usually involves human effort, or apply classification algorithms on heuristics extracted from the pages. While the former has an extremely low false positive rate, human-verified blacklists do not generalize well to future unseen cases. Furthermore, human-verified blacklists can be slow to respond to new phishing attacks. For example, Phishtank [2] statistics in July 2008 show that it took on average 14 hours to verify that a URL was a phishing attack [3]. Finally, human-verified blacklists can be easily overwhelmed by automatically generated URLs. On the other hand, heuristics-based approaches enjoy the flexibility of being able to recognize new phish, but they often lead to a relatively higher false positive rate. The three major phishing blacklists, operated by Google, Microsoft, and Phishtank, all use human verification, in spite of the amount of human labor required, primarily due to concerns over liability for false positives.

The goal of our work is to demonstrate that combining human-verified blacklists with information retrieval and machine learning techniques can yield a system that is fast, highly reliable and adaptive to new attacks. Cova et al observed 583 unique toolkits over a 2 month period beginning in April 2008, suggesting that many phishing attacks were generated by toolkits [4]. This lead us to explore techniques that exploit the semantics of the webpage content in examining inter-webpage similarity. Specifically, we investigate in this paper a detection framework of two algorithms aided by URL blacklists and domain whitelists. Our first algorithm, shingling, which is based on well-known search engine techniques, utilizes finer-grained units of the page content, n-grams or n-shingles, to evaluate inter-webpage similarity. Its best performance is a true positive rate (TP) of 87.42% with a false positive rate (FP) of 0.0434% in our experiment. Exploiting skewed word distributions on typical phishing pages, our second algorithm builds probabilistic language models from the content, and classifies query pages based on the distance between the language models via the Dirichlet-smoothed Kullback-Leibler (KL) divergence, achieving 86.51% TP and 0.286% FP in our experiment.

To evaluate our algorithms, we also proposed two baselines. The first is a hash-based near-duplicate detection method using secure hash functions aiming at capturing identical phishing sites, with a TP of 73.36% and FP of 0% in our experiment. The second, a heuristics-constrained k-centroid clustering algorithm, exploits the patterns in URLs without accessing the page content, achieving 47.3% TP and 0% FP. In addition, evaluation against CANTINA [5], which has been demonstrated to be comparable to state-of-the-art toolbars, shows that our content-based detection approaches perform significantly better in terms of FP while comparable on TP.

The major contribution of this paper is two fold.

1) Leveraging human-verified blacklists with a focus on minimizing false positives, we present two novel and practical content-based algorithms, which achieved a low FP of 0.0434% and a TP of 87.42% in our experiment.
2) Being able to operate in an online fashion updated incrementally by a few out of a large number of new phishing instances, our algorithms are able to adapt quickly to the constantly evolving zero-hour phish and toolkits and thus perform robustly over time.

We don't contend that our specific algorithms are unbeatable, rather, we expect that our framework can be easily extended to address new attacks. Discovering and generalizing from patterns of attacks provides a degree of leverage against toolkit and fast-flux techniques, even if our proposed approaches lose efficiency over time.

The structure of this paper is organized as follows. In section 2, we give a brief introduction of related anti-phishing research, which is followed by an elaboration of our approach in section 3. The whole experiment setup is described thoroughly in section 4 and further discussion is given in section 5. Conclusions are drawn in the last section.

2. Related Work

2.1. Existing Detection Methods

Phish detection is under intensive study recently, and a plethora of methods have been proposed to attack this problem.

In the URL signature camp, Garera et al [6] identified a set of fine-grained heuristics from URLs, and combined them with other features including the page rank features, domain-based features and keyword-based features (a summarized set of eight sensitive words that frequently appear in phishing URLs) to detect phish. Specifically, they categorized phishing URLs into four groups, each capturing a phishing pattern. Applying a logistic regression model on 18 features yielded an average TP of 95.8% and FP of 1.2% over a repository of 2508 URLs.

On another frontier, a variety of heuristics have been proposed for phish detection. In [7], the authors came up with a total of 18 properties after investigating page structures, including forms, input fields, links, whitelist references, script tags, suspicious URLs, use of SSL (https), etc. The J48 decision tree algorithm was applied on these features and achieved a TP of 83.09% and a FP of 0.43% over a corpus with 4149 good pages and 680 phishing pages. Pan et al [8] proposed a method to extract the webpage identity from key parts of the HTML via the χ^2 test, and compiled a list of features based upon the extracted identity. Trained with support vector machines (SVM), their features achieved an average FP of about 12%. In another work, Zhang et al [5] proposed a content-based approach named CANTINA applying the TF-IDF metric in information retrieval (IR). A linear classifier was built from a set of features including TF-IDF and achieved 89% TP and 1% FP on 100 phishing URLs and 100 legitimate URLs.

In addition to the research works introduced above, anti-phishing toolbars based on different techniques are also available, many of which exploit blacklists to assure close-to-zero false positive rate. The working mechanism of the EarthLink toolbar [9] is based on a blacklist as well as some other heuristics such as the domain registration information, etc. SpoofGuard [10] examines phishing signatures via a list of heuristics including seen domains, URL obfuscation, non-standard port numbers, image hashes, etc. Alarm is raised if the weighted sum of the heuristics exceeds a threshold. Microsoft Internet Explorer (IE) 7 relies on a domain whitelist and blacklist from Microsoft's servers to judge the legitimacy of an incoming URL.

Similarly, the Google Safe Browsing toolbar [11] and NetCraft [12] also use a URL blacklist to detect phish.

All of the papers above have the view of maximizing true positives while minimizing false positives. Our view in this paper is subtly different, which is to see how high our true positive rate can be while maintaining close to 0% false positives. As we noted in the introduction, industry has not adopted many of those heuristics above due to concerns about poor user experience for false positives as well as reasons of liability. Thus, our work here deliberately takes a conservative approach, though as we will show, we still get reasonably good true positive rates. Furthermore, our work could also be combined later on with more aggressive heuristics, providing an adjustable range depending on a user or provider's willingness to accept false positives.

2.2. Rock Phish

The "rock phish gang", first appearing in 2004, refers to a group of cyber-space criminals that carry out phishing attacks by using compromised machines as proxies, relaying user requests to one of many back-end servers that each serve a large number of phishing websites. Multiple domains are typically registered within a short time frame, based on which long URLs are crafted and assigned to those fake sites, leading to unique URLs for different phishing sites and thus rendering blacklist-based detection methods futile. Rock phish websites are usually created with phishing toolkits covering a variety of target organizations. The rock phish gang is believed to be responsible for more than half of the phishing attacks around the world. An analysis of rock-phishing sites by Moore et al [13] from February to April in 2007 reveals that 52.6% of all Phishtank reports are rock phish during that time period.

Rock phish is constantly evolving, and a recent report [14] by the RSA FraudAction Research Lab provides evidence indicating that the rock phish gang is updating its phishing infrastructure to the sophisticated fast-flux Asprox botnets, which work by infecting more machines and using them as proxies for either additional infection sources or phishing attack hosts. This transition from the previous simplistic proxy clients to the highly-advanced fast-flux network makes rock phishing more effective and harder to cope with.

3. Algorithmic Methods

In this section, we first describe two baselines, based on hard matching techniques. The point of these hard matching methods is to serve as simple baselines. Though demonstrated by our experiment to be rather effective against today's phishing toolkits, we also expect them to be defeated fairly easily. Accordingly, we also give two novel probabilistic detection methods that are designed to be more robust and effective.

In our method, we only used positive examples of phish in the training set and did not train our models on any instances of legitimate pages. A merit of this strategy is that we are extracting signatures that are truly phish, and by means of soft matching techniques and whitelist filtering, we are able to both enlarge the capture scope and further reduce the false positive rate, which addresses one of the major concerns of the anti-phishing community.

We now give some notational conventions that will be used in the remaining of this paper.

Notations *Let q and d represent a query webpage and a phishing page in the phish corpus respectively. Let Q and D represent the set of all query pages and the set of all training phish. We use w to denote a word on the webpage, and $p(w|d)$ is the word distribution on page d. URL host names consist of a number of dot-separated parts, and we define each part as a "segment".*

3.1. Baseline 1: Hash-based Near-duplicate Page Detection

The growth of rock phish as well as phishing toolkits [4] produce an abundance of webpages very similar or identical to each other in terms of page HTML. This property motivated us to use duplicate identification algorithms for phish detection.

The SHA1 security hashing algorithm, a popular method for detecting duplicate documents suggested by NIST [15], is a secure and fast procedure that produces 20-byte or 160-bit hash values and is applicable to text of any length with a low likelihood of hash collisions. We use hashes to identify matching phishing sites, and so the odds of a chance collision with a legitimate site are vanishingly small.

Based on SHA1 hashing, our duplicate detection algorithm proceeds as follows. After filtering out good webpages via a domain whitelist, it removes all the spaces in the page HTML. Subsequently, it identifies all default values to the input fields in the HTML forms and replaces them with empty strings. This is often seen in the value field for email inputs, in which many phishing toolkits insert a random email address. These meaningless addresses almost never point to actual emails, but are sufficient to render the hashing process to give different hash values. After these two steps, we compute a SHA1 hash on the processed HTML, which is then compared with a pool of hash values for phishing webpages, and trigger an alarm if a match is found.

3.2. Baseline 2: Heuristic-constrained k-centroid Clustering

The central idea of the clustering algorithm is to build k clusters out of the training phish URLs, and in each cluster, choose the most representative URL as the centroid. Future query pages are evaluated based on their similarity with the cluster centroids. The average distance to other URLs in the same cluster is our heuristic in electing cluster centroids. In this method, initial clusters are automatically learned without a human-specified k.

Before delving into algorithmic details, we show a few phishing URLs (truncated from behind due to limited space) from our corpus below to illustrate one typical obfuscation trick, i.e., putting the name of the organization being phished in front of the mass-registered domain names.

http://ww5.**chase.com**.bank84.**com/ccp/clientconfirm.jsp**/?siteid=17x

http://ww2.**chase.com**.id746.**com/ccp/clientconfirm.jsp**/?site=18bzkw

http://ww8.**chase.com**.cert83.**com/ccp/clientconfirm.jsp**/?taskid=14zr

http://ww6.**chase.com**.sid36.**com/ccp/clientconfirm.jsp**/?ref=22hknZ

We now define a set of constraints for the clustering algorithm.

1) **Minimum host name length constraint**. Host names must have at least 4 segments. URLs with shorter host names will not be added into any cluster.
2) **Mandatory path constraint**. URLs must have at least one directory level in the path part. For example, an eligible URL candidate *http://www.9x9x.web.ve/sc/saw-cgi/eBayISAPI.dll/login.htm* has a 3-level directory (*/sc/saw-cgi/eBayISAPI.dll/*), while URL *http://wellsfargo-online3.4t.com/login.htm* has no directory hierarchy in the path part and is excluded from being clustered.
3) **Loose match host name constraint**. Two host names must have the same length in terms of segments, and among them, at most 2 corresponding segment pairs are allowed to be different, and the remaining must either be identical, or share a common subsequence [1] with 2 or more characters. For example, among the 4 segments in the following two host names *webexpress1.tdbanknorth.com.b06.su* and *webexpress6.tdbanknorth.com.asp8.su*, pair *webexpress1* and *webexpress6* counts as one match since they share a 10-character subsequence *webexpress*, while pair *b06* and *asp8* does not.
4) **Exact match path constraint**. The directory hierarchy in the URL path part must be identical for two phishing URLs to be clustered together.
5) **Minimum cluster size constraint**. Each phishing URL cluster must have at least 2 URLs, which is the optimal value according to cross validation.

Algorithm 1 LearnInitialClusters

```
 1: Input: training phish URL corpus $D$, constraint set $S$
 2: Output: phish URL clusters $C$
 3: shuffle $D$
 4: $C \leftarrow \phi$
 5: for all $d_1 \in D$ do
 6:     $c \leftarrow \{d_1\}$, remove $d_1$ from $D$
 7:     for all $d_2 (d_2 \neq d_1) \in D$ do
 8:         if $d_1, d_2$ satisfy constraint 1 to 4 in $S$ then
 9:             $c \leftarrow c \cup \{d_2\}$, remove $d_2$ from $D$
10:         end if
11:     end for
12:     if cluster $c$ satisfies constraint 5 in $S$ then
13:         $C \leftarrow C \cup c$
14:     else
15:         restore $\forall d \in c$ back into $D$, next iteration starts with the URL after $d_1$
16:     end if
17: end for
18: return $C$
```

Algorithm 1–3 show the procedures of initial cluster learning, cluster centroid computation, and query page evaluation respectively. Initial clusters are formed by one scan of the phish URLs, which might be dependent on the order of the URLs, and we apply random shuffling beforehand to mitigate its influence on cluster formation.

We now define the formula for computing distance between host names in Eq (1)(2), in which h_j denotes

1. In this context, a subsequence is defined to be the longest contiguous sequence of dashes and alphabetic letters. For instance, 'online-business9online' has two such subsequences 'online-business' and 'online'.

Algorithm 2 LearnClusterCentroids

```
1: Input: training phish URL corpus $D$, constraint set $S$
2: Output: centroids of phish clusters $M$
3: $M \leftarrow \phi$
4: $C \leftarrow$ LearnInitialClusters($D$, $S$)
5: for all cluster $c \in C$ do
6:     $distances \leftarrow \phi$
7:     for all URL $d \in c$ do
8:         $dis \leftarrow$ compute the average distance of $d$ to all other URLs in $c$ by Eq (1)
9:         $distances \leftarrow distances \cup \{dis\}$
10:    end for
11:    $centroid \leftarrow \underset{d}{\text{argmin}}\{distances\}$
12:    $M \leftarrow M \cup centroid$
13: end for
14: return $M$
```

Algorithm 3 QueryPageEvaluation

```
1: Input: centroids of the phish clusters $M$, testing URL $q$, constraint set $S$
2: Output: prediction on $q$
3: $prediction \leftarrow$ good page
4: for all $d \in M$ do
5:     if $d, q$ satisfy constraint 1 to 4 in $S$ then
6:         $prediction \leftarrow$ phish
7:         break
8:     end if
9: end for
10: return $prediction$
```

the j-th host name, $p_j(i)$ denotes the i-th segment of h_j, $|\cdot|$ is the length operator, denoting length in terms of segments in $|h_j|$ and characters in other occasions. LCS returns the longest common subsequence (length $\geq$ 2) of two strings. One rationale behind this is that close host names may indicate similar webpages.

$$Distance(h_1, h_2) = \sum_{i=1}^{|h_1|} dis(p_1(i), p_2(i)) \quad (1)$$

$$dis(p_1(i), p_2(i)) = \begin{cases} 0.0 & \text{if } p_1(i) = p_2(i) \\ 1 - \dfrac{|LCS(p_1(i), p_2(i))|}{\max(|p_1(i)|, |p_2(i)|)} & \text{if not} \end{cases} \quad (2)$$

Initial cluster formation has a time complexity of $O(|D|^2)$, centroid learning is $O(|C| \times Len_{avg}^2)$, and the detection procedure is $O(|C|)$, where Len_{avg} is the average size of the initial clusters.

3.3. Shingle-based Soft Matching

Phishing pages typically have a login form requesting financial information, and the textual content usually shows a certain syntactic and semantic regularity, such as the vocabulary used ("welcome to", "sign in to your account", "user ID", etc.). Intuitively, these linguistic-level pieces could be used to identify phishing webpages. In this section, we give a soft matching method in light of this semantic uniformity, which allows more flexibility in classification than rigid URL/domain matching and thus is more robust to random noise often seen in the HTML. Further, this method is based on a well-known technique used by search engines to find duplicates and should perform particularly well against fast-flux and toolkit based attacks, which are evident in the large number of very similar sites that result [4].

In an effort to capture the webpage semantics on a finer-grained level, we used the notion of n-gram or n-shingle, and measured the inter-page similarity based on these basic units. n-gram, a term from natural language processing (NLP) community, is a subsequence of n contiguous tokens. For example, sample text ***connect with the eBay community*** has 3-grams $\{$***connect with the, with the eBay, the eBay community***$\}$. The shingling method [16] employs a metric named resemblance to calculate the percent of common n-grams between two webpages. Let $S(p)$ denote the set of unique n-grams in p and the similarity metric resemblance $r(q, d)$ is defined as

$$r(q, d) = \frac{|S(q) \cap S(d)|}{|S(q) \cup S(d)|} \quad (3)$$

Our soft matching approach first breaks each $d \in D$ into a set of unique n-grams, and saves them in memory to speedup running. After excluding good pages whose domains appear in the whitelist, we compute resemblance $r(q, d)$ $\forall d \in D$ for a query page q, and fire an alarm whenever $r(q, d)$ exceeds a threshold t. Cross validation (CV), a standard machine learning procedure, is adopted to choose the value of t that yields the best detection result.

3.4. Phish Detection via Probabilistic Language Modeling

In light of the nature of the phishing sites, the probability distribution of the words on webpages is usually skewed, with terms related to sensitive customer information like "account", "password" occurring more often than other words such as "algorithm", "IPv6", etc. This observation offers insight into new possibilities, and in this section, we present a method exploiting the relative frequencies of the words in the page content via language modeling techniques for phish detection.

In particular, we build a unigram language model for each webpage, and evaluate page similarity based on this probabilistic representation of the webpages. Unlike higher-order language models, unigram model $p(w|d)$ is memory-less, only considering individual words while ignoring the context.

To measure the similarity of the unigram models, we used KL divergence, a measure of distance between two probability distributions in information theory

$$KL(q \parallel d) = \sum_w p(w|q) \log \frac{p(w|q)}{p(w|d)} \tag{4}$$

which is non-negative and takes a zero value only when distributions q and d are identical.

With some derivation, we obtain the final scoring function in measuring inter-page distance in our language modeling detection method in Eq (5)

$$KL(q \parallel d) \propto -\Big(\sum_{\substack{w: tf(w,q)>0 \\ tf(w,d)>0}} p(w|q) \log \big(1 + \frac{tf(w,d)}{\mu p(w|\mathcal{C})}\big) + \log \frac{\mu}{\mu + |d|} \Big) \tag{5}$$

Due to the limitation of space, we do not show the steps in arriving at the scoring function in Eq (5) here and refer readers to the appendix for details.

Our detection algorithm in this section builds a language model for each webpage by learning a unigram distribution, and evaluates the discrepancy between each phishing page and the query page via the Dirichlet-smoothed KL divergence in Eq (5). The algorithm classifies the query as phish, once the KL score is below a threshold t, which is also learned by cross validation. As in our other methods, domain whitelist fulfills the function of reducing false positives.

Two things deserve mentioning before we close this section. First, although Eq (4) is always positive, the derivation (see appendix) leading to Eq (5) dropped a positive term irrelevant to phishing documents, thus allowing both positive and negative values for the scoring function in Eq (5). In light of this, we also experimented with negative threshold t in our evaluation. Second, a set of words called stopwords that occur very often yet bear little actual meaning ("the", "of", etc.) introduce tremendous noise to the language modeling process and usually lead to highly-skewed distributions. We removed them from the page text beforehand.

4. Experiment

4.1. Evaluation Metric

We adopted the standard true positive rate (TP) (also called recall), false positive rate (FP) and precision (Prec) in our evaluation defined below, in which $p2p, p2n, n2p, n2n$ stand for the number of phishing webpages correctly classified as phish, the number of phishing pages wrongly classified as good pages, the number of legitimate pages wrongly classified as phish (false positive) and the number of legitimate instances correctly classified as legitimate respectively. We will use shorthand notations $TP, FP, Prec$ to represent these three metrics in the rest of this paper.

$$\text{TP} = \frac{p2p}{p2p + p2n} \tag{6}$$

$$\text{FP} = \frac{n2p}{n2p + n2n} \tag{7}$$

$$\text{Prec} = \frac{p2p}{p2p + n2p} \tag{8}$$

4.2. Data Source and Usage

Phishing sites are usually ephemeral, and most pages won't last more than a few days. To fully study our approach over a larger corpus, we downloaded the phishing pages when they were still alive and conducted our experiment in an offline mode. Our downloader employed the Internet Explorer to render the webpages and execute Javascript, so that the DOM of the downloaded copy truly corresponds to the page content and thus gets around phishing obfuscations. We also downloaded images to allow us to use CANTINA [5] for comparison.

Known good domains are a crucial part of our detection framework, and we collected such domains from three sources. Google safe browsing provides a publicly-available database [17] with 2770 white domains by the time of our experiment, and after duplicate removal, we obtained a total of 2682 unique domains. Millersmiles [18] maintains an archive of the most common spam targets such as ebay, and we extracted 424 unique domains out of 732 entries after mapping organization names to domains and removing duplicates. Moreover, we also employed an online white domain service [19], which uses DNS lookup to determine if the query domain is on the whitelist.

Like other whitelists, this online database's coverage is rather limited.

Our webpage collection consists of phishing cases from one source, and good webpages from six sources. To eliminate the influence of language heterogeneity on our content-based methods, we only downloaded English webpages in our experiment.

For phishing instances, we used the XML feed of Phishtank [20], a large community-based anti-phishing service with $29,245$ accounts [2]. Phishtank depends on web users from around the world to submit suspicious phish, which are then verified via a simple threshold voting mechanism. Genuine phishing URLs are added into a downloadable blacklist after verification, and so far, Phishtank has $369,905$ verified phish. We started downloading the feed in early May of 2008 and manually examined the downloaded webpages to remove legitimate cases, 404 errors, and other types of noisy pages, collecting a total of 6944 phishing webpages during a four-month period.

Alexa.com maintains a top 100 websites list [21] for a variety of languages, and we crawled the homepages of the top 100 English sites to a limited depth, collecting 1039 good webpages in that category.

To introduce webpages with login forms into our data set, we downloaded 961 login pages, utilizing Google's "inurl" operator and searching for pages with keywords such as "signin", "login", etc. in the URLs. Although it is not necessarily true that each page we downloaded contains an actual login form, it is guaranteed that all of these URLs point to legitimate websites.

3Sharp [22] released a public report on anti-phishing toolbar evaluation two years ago, and we downloaded 101 good English pages out of the 500 provided in the report that still existed at the time of downloading.

Moreover, we went to Yahoo directory's generic bank category [23], crawling the bank homepages for a varying number of steps within the same domains and collecting 988 bank pages.

Likewise, we conducted crawling on other categories [24]–[29] of Yahoo directory including US bank, credit union, online escrow services, travel agencies, real estates and financial services, and gathered 371 webpages. We name this data set "*Yahoo misc pages*" for reference convenience.

To test the robustness of our methods, we chose 83 login pages of the most common phishing target websites, such as ebay, bankofamerica, paypal, etc. We call this data set "*prominent pages*" in the rest of this paper. Note that none of the other five categories has overlap with URLs in prominent pages, rendering this category truly independent.

4.3. Test Methodology

In our experiment, we adopted the standard train-validation-test methodology based on machine learning common practices, in which we held out a portion of the whole data set for final testing, used one portion of the remaining data for model training, and the other portion for model tuning. All the train-validation-test splits were performed randomly.

Specifically, we held out 3472 phishing pages, 519 alexa pages, 480 login pages, 50 3sharp pages, all 988 bank pages, 185 Yahoo misc pages, and all 83 prominent pages for testing. Since our training set was composed entirely of phishing instances, we used all the remaining good webpages for parameter tuning, which had 520 alexa pages, 481 login pages, 51 3sharp pages, and 186 Yahoo misc pages. These numbers are approximately half of each corresponding corpus size. Among the remaining 3472 phishing pages we chose for training purposes, referred to as "*whole training corpus*" for convenience, p percent was selected to train the models, and the rest $1 - p$ percent was used for tuning model parameters. We varied the values of p to examine the size of the training phish set on the detection performance.

To reduce random variation and avoid a lucky train/validation/test split, we used the average statistics of 10 runs in all our experiments.

4.4. Experimental Results

In this section, we will first report the experimental results of model tuning, and then present the final testing statistics of our detection methods based on the optimal parameters. All model tuning was conducted on a training set with 70% phish from the whole training corpus. Note that the hashing baseline has no parameters to tune.

4.4.1. Model Tuning.

k-centroid Clustering Baseline

The parameter in our clustering baseline is the minimum cluster size, and we achieved different detection performance under various values, as seen in Fig.1.

Since good URLs usually differ drastically from phishing URLs (see [6] for phish URL taxonomy) and we imposed strict constraints on the formation of initial clusters, we achieved 0% FP and 100% precision regardless of the minimum cluster size thresholds. It is unsurprising that average TP (Fig.1) decreased as the threshold grew, because we lost clusters due to the stricter constraints. The highest TP is 47.18% at a threshold of 2 and lowest is 41.18% at a threshold

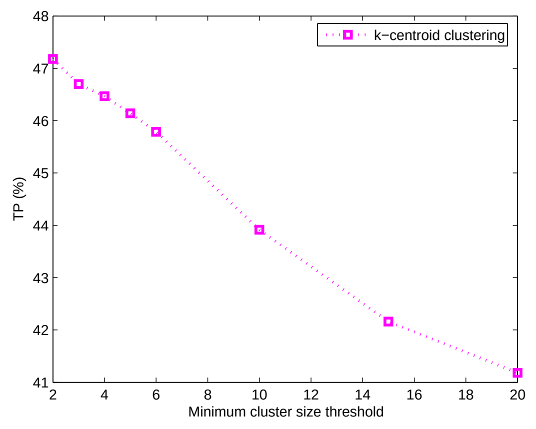


Figure 1. *k-centroid clustering parameter tuning. Unsurprisingly, TP declines as the minimum cluster size grows, with the highest 47.18% occurring at 2 clusters and lowest 41.18% appearing at 20 clusters.*

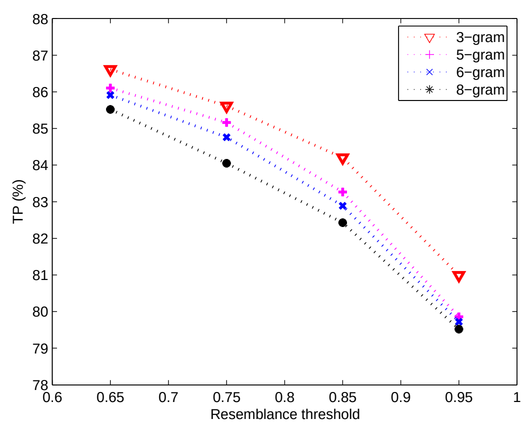


Figure 2. *Shingling parameter tuning. As resemblance threshold is increased, the rate of detection drops as expected. TP tops at 86.61% under 3-gram and a threshold of 0.65.*

Table 1. *Average number of clusters and cluster size. 2430 training phish URLs form less than 70 clusters, the centroids of which are used for query page evaluation, dramatically boosting the testing efficiency.*

	Minimum cluster size threshold							
	2	3	4	5	6	10	15	20
Avg cluster number	68.00	46.60	40.20	36.40	35.20	27.50	20.40	17.60
Avg cluster size	21.08	29.84	34.09	37.34	38.42	47.33	59.97	66.80

of 20. Since no false positives occurred and precision remained 100% for all conditions, we do not plot the graphs for FP and Prec here. The optimal setting according to cross validation is a threshold of 2.

Table 1 shows the average number of clusters and average cluster size under each minimum cluster size (10 runs for each threshold). As expected, the former kept declining and the latter continued ascending. Noticeably, we had a total of 2430 training phish URLs in this experiment, and the clustering method dramatically reduced this figure to below 70 centroids.

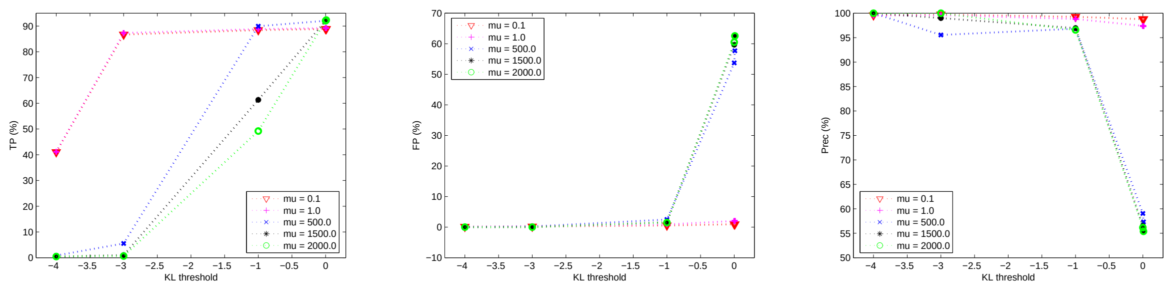
Shingle-based Soft Matching

Fig.2 shows the validation performance of this approach under different shingle lengths and resemblance thresholds. For all n-grams in the evaluation, the true positive rate monotonically decreased as we raised the resemblance bar higher. Fig.2 also suggests that shingling with shorter n-grams tends to capture more phish, which makes perfect sense, in that we have more such fine-grained units in a webpage with smaller n. With a resemblance of 65%, shingling achieved over 85% TP under all shingle lengths, which manifested the prevalent similarity among phishing webpages due to rock phishing. All scenarios had 0% FP and 100% Prec, which are not plotted here.

Language Modeling

When it comes to unigram modeling of the page content, the trend of the three metrics is still predictable, since the smaller the threshold of the inter-distribution distance (KL) is, the stricter the criterion becomes to classify a query page as phish, and therefore the lower the true positive and false positive rate, and the higher the precision.

The influence of various Dirichlet prior parameter values, however, is not that straightforward, and Fig.3 suggests that smaller μ almost unanimously outperformed bigger values. We would give an intuitive interpretation based on the analytical form of the scoring function in Eq (5). Considering the property of logarithm $\log(x)$ that the curve to the left of $x = 1$ is much steeper than that to the right, a small μ will yield a very small negative number (big in terms of absolute value) for the second term $\log \frac{\mu}{\mu+|d|}$ in the parenthesis in Eq (5). The first term in parenthesis, however, tends to be a small positive number regardless of μ due to the multiplicative factor $p(w|q)$ and the logarithm property. The net result is thus a positive score for Eq (5) and for sufficiently small distance thresholds, classification



(a) TP I (b) FP (c) Prec

Figure 3. *Language modeling parameter tuning. With smaller μ, it is harder to classify a page as phish, under which FP is low under all threshold values. Optimal setting is a TP of 86.71%, a FP of 0.19% and a Prec of 99.74% at $\mu = 0.1$ and threshold $= -3.0$*

is in favor of a low false positive rate. Optimal setting for this model is 0.1 for μ and -3.0 for the distance threshold, with a TP of 86.71%, FP of 0.19% and precision of 99.74%.

4.4.2. Model Testing. The optimal parameters obtained via cross validation in the previous section were used for each model, and the performance of our methods and the baselines on the held-out data set is shown in Table 2-3.

The TP on the testing corpus is shown in Table 2. By removing spaces and default input values of the forms in the HTML, our hashing baseline was able to identify over 60% phish using only 20% of our training data, which is an indicator of the widespread use of phishing toolkits. Based on webpage URLs only, the clustering baseline was able to correctly detect over 40% of the testing phish. It missed more than half, however, probably due to the fact that URLs could be easily obfuscated in different ways [6], but this 40% is based on an extremely simple technique that could be improved by trying different constraints. The soft n-gram-based shingling method and probabilistic language modeling approaches avoided the brittle hard matching strategy, and recognized over 80% phish, much more than the baselines. Within each model, the TP increased monotonically as more phishing pages were introduced into the training set.

For the FP (Table 3), hashing and clustering performed perfectly (0%) under training sets of all sizes. This is one of the merits of the strict matching mechanisms, because it is unlikely that a legitimate webpage could match our URL pattern heuristics, or contain the same HTML as a phishing page without also being on a whitelist. The soft shingling method and probabilistic language modeling technique misclassified a few good instances, but still kept the false positive rate as low as 0.0434% and 0.286% respectively.

The TP-FP dilemma is an analog of the bias-variance tradeoff in machine learning, which basically states that bias and variance are two inevitable components of the general error and introducing a certain amount of extra bias in the model may eventually boost its performance. In here, the much higher true positive rate as a result of increasing model complexity in our probabilistic approaches came at the price of non-zero false positives.

Manual examination on the false positives of the language modeling approach reveals that this type of error mostly came from three sources, all partially due to the limited scope of the whitelist. First, the query page is very short and contains only a few words, resulting in a big positive number for the first term in the parenthesis in Eq (5) and accordingly a negative KL score smaller than the threshold. Second, the legitimate query page and some phishing page are very similar or even identical in terms of page content. Third, the query page and some phishing page have similar vocabularies and word distributions, though the content might look quite different. This is the hard case that needs some ingenuity to solve. The shingling method considers the order of the terms in computing similarities and only suffers from the second problem above, thus achieving fewer false positives.

4.4.3. Evaluation against Toolbars. In [5], Zhang et al proposed CANTINA, a content-based method, which performed competitively in their experiment against two state-of-the-art toolbars, SpoofGuard and Netcraft. We implemented an offline version of CANTINA, and evaluated our algorithms with CANTINA on the same 10 randomly generated testing sets as above.

Table 4 shows that the TPs of the shingling

Table 2. *Test TP (%) with optimal model parameters. All techniques improve in performance given a larger training set. The shingling and language modeling algorithms significantly outperform the hashing and clustering baselines, with a maximum TP of 87.42% and 86.51% respectively.*

	Percent of training phish (%)								
Method	20	30	40	50	60	70	80	90	100
Hashing baseline	64.02	66.05	67.82	69.18	70.39	71.26	72.06	72.75	73.36
Clustering baseline	45.12	45.88	46.32	46.72	46.83	46.99	47.18	47.24	47.3
Shingling	82.47	83.86	84.82	85.4	86.07	86.44	86.81	87.19	87.42
Language Modeling	80.16	81.92	83.22	84.05	84.84	85.31	85.83	86.16	86.51

Table 3. *Test FP (%) with optimal model parameters. The shingling and language modeling approaches have low but non-zero false positives, a cost of higher model complexity. The hard-matching hashing and clustering baselines have 0% FP under all conditions.*

	Percent of training phish (%)								
Method	20	30	40	50	60	70	80	90	100
Hashing baseline	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Clustering baseline	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Shingling	0.013	0.0087	0.0174	0.0217	0.0217	0.0304	0.0434	0.0347	0.0434
Language Modeling	0.15	0.2	0.195	0.21	0.25	0.26	0.282	0.27	0.286

Table 4. Baselines and our probabilistic algorithms vs CANTINA. Shingling and language modeling methods perform comparably with CANTINA in terms of TP, while far outperform it on FP ($0.0434\%/0.286\%$ vs 5.4%). Hypothesis tests compare all our methods against CANTINA under TP and FP, with all FP cases giving statistically significant results (marked by $*$).

	Baselines		Probabilistic methods		Existing method
	Hashing	Clustering	Shingling	Language Modeling	CANTINA
TP(%)	73.36	47.3	87.42	86.51	91.34
FP(%)	0.0	0.0	0.0434	0.286	5.4
p-value (TP%)	0.99999	0.99999	0.99999	0.99999	
p-value (FP%)	$\ll$ 0.00001 ($*$)	$\ll$ 0.00001 ($*$)	$\ll$ 0.00001 ($*$)	$\ll$ 0.00001 ($*$)	

and language modeling algorithms are comparable to CANTINA, while their FPs are much better (0.0434%/ 0.286% vs 5.4%). Hypothesis tests were conducted comparing the TP/FP of the baselines and our probabilistic methods with CANTINA, all with the null hypothesis hypothesizing equal performance while alternative hypothesis favoring our methods. Table 4 shows statistical significance in all FP cases (marked by $*$) with strong evidence in favor of our detection algorithms.

Even though phishing signatures constantly evolve, the conclusion from [5] still carries and our experiment results in this section suggest that our proposed algorithms are at least as good as, if not better than, the state-of-the-art anti-phishing toolbars.

5. Discussion

5.1. False Positives

One important issue in our framework is our domain whitelist. For example, all false positives of the shingling method and partial of the language modeling method using 100% training phish (rightmost column of Table 3) are caused by the incompleteness of this whitelist. Enlarging the coverage of the whitelist is a necessary step to a more practical phish detection system. Alternatively, the TF-IDF heuristic in [5], which utilizes search engines to search top TF-IDF terms from the page content and reports phish based on the discrepancy of top result entries and the page domain, can be applied here when the query domain is absent from the whitelist, to further reduce false positives. Due to the use of a whitelist, however, it is difficult for phishers to actively introduce errors that will raise the FP of our algorithms. The problem is also alleviated somewhat, however, by the fact that phishing activities usually target only a small number of well-known websites, and building a compact whitelist still often yields decent performance.

5.2. The Dominance of Rock Phish

In this paper, we proposed two hard matching methods as simple baselines with 73.36% and 47.3% TP respectively in our experiment, strongly illustrating the prevalence of rock phishing and toolkit-based attacks.

Phishing pages have to look somewhat legitimate to the users for the attack to succeed, and there are generally two ways to achieve that. The first involves visual similarity, a common practice of which is to put target brand logos on the fake sites. The second centers around the semantics of the textual content of the phishing pages, which often adopts finance-related phrases and words as analyzed in section 3. Both are unlikely to change in light of the nature of the phishing attacks, which lend strong credibility to the effectiveness and practicality of our proposed methods.

5.3. Running Time

Time complexity is another issue that deserves our attention. The theoretical complexity of our algorithms for query page evaluation is linear in the number of training phish. Table 5 shows the 10-run average running time of our algorithms and the baselines on each test webpage (a total of 5777 pages) using the whole training corpus with the models in memory on a machine with 1.73GHz CPU and 1G RAM. Test on each case consists of three phases, i.e., loading, processing, classifying, and the value in each cell was the total time of the three stages. Hashing and clustering were comparable, while both were at least an order of magnitude faster than the two soft matching algorithms, mainly in that the latter had to compute distance scores, which was much more computationally expensive. Moreover, the number of unique words on a webpage is usually much smaller than the number of unique n-grams, which partially explains the superiority of language modeling over shingling in terms of running time. Training the models, however, took significantly longer time, but we can do it offline and the online testing stage does not have to relearn the models, thus running much faster. The statistics in the table suggest that with proper caching, all methods are fast and potentially applicable for realtime phish detection. To further optimize our algorithms, we could distribute our models to multiple servers to parallelize the detection process or leverage advanced architecture assigning different weights to blacklist URLs and traversing the phishing URLs in a particular order to achieve faster-than-linear running time.

5.4. The Robustness of Our Algorithms

Unlike invisible DOM objects, such as the href field of anchors and the action field of forms, the semantics of the visible page content are nontrivial to obfuscate, because the phishing sites have to show the users what information is requested via text and there are only a limited number of finance-related representations. Exploiting this linguistic uniformity, our proposed content-based methods achieved 87.42% and 86.51% respectively.

Phishers could try to subvert our algorithms by injecting garbage text to the DOM with colors designed to be hardly perceptible. This is, however, more of an implementation issue than a design one, and we can easily fix this by removing text which makes use of such color manipulations as part of our pre-processing.

A more extreme approach might involve removing all textual components and only using images of text on the phishing sites. This is a hard case. However, we really doubt if any legitimate entity would design their website this way (especially the login page), and would probably directly classify this kind of page as phish.

Raw IP addresses are heavily used in the phishing world, and detection methods based on machine learning techniques usually take the use of a raw IP address as a feature in learning classifiers. Our algorithms work by examining page content via the DOM in detecting phish, rendering URL/IP oriented obfuscation futile.

Another issue is the zero-hour attack, i.e., new phishing patterns, which presents a rather serious problem to the existing blacklist-based methods, because the large number of new unique phishing URLs cannot be identified and added to the blacklists rapidly enough. Though the first a few cases of the new attacks are initially able to evade our detection, we only need to identify a few new phishing instances, update our models via online retraining, and we are subsequently able to block the rest of the attacks while maintaining a nearly zero false positive rate. This is a significant improvement over the blacklist-based methods that are generally unable to cope with a high volume of unique phish URLs.

Although phishers will likely adapt their techniques and defeat our specific algorithms eventually, the general idea of relaxing human-verified blacklists

Table 5. *Average running time on each test webpage (out of* 5777 *pages) of* 10 *runs using the whole training corpus with the models in memory on a machine with 1.73GHz CPU and 1G RAM.*

Algorithms	hashing baseline	k-centroid clustering baseline	shingling	language modeling
Running time (ms)	0.002203	0.001969	0.126642	0.065550

via probabilistic approaches still holds, and further content-based or vision-based methods could be investigated which would again provide strong protection.

6. Conclusions and Future Work

Combining human-verified blacklists with information retrieval and machine learning techniques, we presented in this paper a phish detection framework that is fast, highly reliable, and adaptive to new phishing attacks. Specifically, we designed and evaluated two content-based algorithms for phish detection harnessing the semantic regularity of the phishing webpages, with a focus of achieving the lowest (ideally zero) false positive rate, a concern which is of paramount importance to anti-phishing industry.

Our algorithms fully exploit the high similarity in the phishing page content. One of our methods relies on finer-grained components, n-gram or n-shingle, to measure the inter-webpage similarity, while another builds language models from the webpages and examines the webpage distances probabilistically via the Dirichlet-smoothed KL divergence. In addition, we proposed a hash-based duplicate detection method and a heuristics-constrained k-centroid clustering algorithm as baselines for comparison. Domain whitelists fulfilled the role of controlling false positives. Extensive experiments over phishing data from Phishtank and good pages from six authority sources showed that our method was able to detect 87.42% of the phish with a low false positive rate of 0.0434%. Moreover, our methods are able to adapt quickly to zero-hour attacks by incrementally updating the models with a few new phishing instances out of the huge magnitude of rock phish attacks, thus providing a feasible framework for industry to vastly improve the existing limited blacklists without increasing their false positives.

It is likely that phishers adapt their techniques and defeat our specific algorithms eventually, however, the general idea of combining human-verified blacklists, information retrieval and machine learning techniques for phish detection still holds, and more content-based or vision-based methods could be investigated to detect further phishing attacks.

Currently, we access domain whitelists by simple lookups, and considering the scarcity of easily available white domains, one possible future research direction is the automatic assessment of new domains given the known good ones. Dealing with webpages of different length is another area that could improve the performance of the language modeling approach.

Acknowledgements

Thanks to the members of the Supporting Trust Decisions project for their feedback. This work was supported in part by the National Science Foundation under grant CCF-0524189 ("Supporting Trust Decisions") and by Army Research Office grant DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems"). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the U.S. government.

References

[1] The Internet Crime Complaint Center (IC3). The 2007 Internet Crime Report, http://www.ic3.gov/media/annualreport/2007_IC3Report.pdf

[2] Phishtank statistics. http://www.phishtank.com/stats.php

[3] Zhang, Y., Egelman, S., Cranor, L. and Hong, J. Phinding Phish: An Evaluation of Anti- Phishing Toolbars. In Proceedings of the 14th Annual Network & Distributed System Security Symposium (NDSS 2007), 2007

[4] Cova, M., Kruegel, C. and Vigna, G. There Is No Free Phish: An Analysis of "Free" and Live Phishing Kits. In Proceedings of the 2nd USENIX Workshop on Offensive Technologies (WOOT'08), 2008

[5] Zhang, Y., Hong, J. and Cranor, L. CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. In Proceedings of the 16th International Conference on World Wide Web (WWW'07), pages 639–648, 2007

[6] Garera, S., Provos, N., Chew, M., and Rubin, A. A Framework for Detection and Measurement of Phishing Attacks. In Proceedings of the 2007 ACM Workshop on Recurring Malcode, pages 1–8, 2007

[7] Ludl, C., McAllister, S., Kirda, E. and and Kruegel, C. On the Effectiveness of Techniques to Detect Phishing Sites. In Lecture Notes in Computer Science (LNCS), volume 4579, pages 20–39, 2007

[8] Pan, Y. and Ding, X. Anomaly Based Web Phishing Page Detection. In Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06), pages 381–392, 2006

[9] EarthLink. http://www.earthlink.net/

[10] Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D. and Mitchell, J.C. Client-side Defense against Web-based Identity Theft. In Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS'04), 2004

[11] Safe Browsing Toolbar. https://wiki.mozilla.org/Phishing_Protection

[12] NetCraft Anti-phishing Toolbar. http://toolbar.netcraft.com/

[13] Moore, T. and Clayton, R. Examining the Impact of Website Take-down on Phishing. In Proceedings of the Anti-phishing Working Groups (APWG) 2nd annual eCrime Researchers Summit. Pages 1–13, 2007

[14] RSA FraudAction Research Lab. What's Going on Between Asprox and Rock Phish? http://www.rsa.com/blog/blog_entry.aspx?id=1338

[15] National Institute of Standards and Technology (NIST). Secure Hash Standard. In Federal Information Processing Standards Publication 180-1, 1995

[16] Broder, A.Z., Glassman, S.C., Manasse, M.S. and Zweig, G. Syntactic Clustering of the Web. In Proceedings of WWW6'1997, pages 391–404, 1997

[17] Google white domain list. http://sb.google.com/safebrowsing/update?version=goog-white-domain:1:1

[18] Millersmiles phishing scam archive. http://www.millersmiles.co.uk/scams.php

[19] URIBL. http://www.uribl.com

[20] Phishtank: http://data.phishtank.com/data/online-valid/

[21] Alexa top 100 English sites. http://www.alexa.com/site/ds/top_sites?ts_mode=lang&lang=en

[22] 3sharp report. Gone Phishing: Evaluating Anti-Phishing Tools for Windows. http://www.3sharp.com/projects/antiphishing/gone-phishing.pdf

[23] Banks, Yahoo Directory. http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Financial_Services/Banking/Banks/

[24] US banks, Yahoo Directory. http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Financial_Services/Banking/Banks/By_Region/U_S__States/

[25] Credit Unions, Yahoo Directory. http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Financial_Services/Banking/Credit_Unions/

[26] Online Escrow Services, Yahoo Directory. http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Financial_Services/Online_Escrow_Services/

[27] Travel Agencies, Yahoo Directory. http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Business_Opportunities/Travel_Agencies/

[28] Real Estates, Yahoo Directory. http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Business_Opportunities/Investment_Opportunities/Real_Estate/

[29] Financial Services, Yahoo Directory. http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Business_Opportunities/Financial_Services/

[30] Zhai, C. and Lafferty, J. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In Proceedings of the 24th ACM Conference on Research and Development in Information Retrieval (SIGIR'02), pages 334–342, 2001

[31] Yang, H. and Callan, J. Near-Duplicate Detection by Instance-level Constrained Clustering. In Proceedings of the 29th ACM Conference on Research and Development in Information Retrieval (SIGIR'06), pages 412–428, 2006

Appendix
Derivation of the KL Scoring Function

KL divergence can be rewritten as

$$\begin{aligned} KL(q \parallel d) &= \sum_w p(w|q) \log p(w|q) - \sum_w p(w|q) \log p(w|d) \\ &\propto -\sum_w p(w|q) \log p(w|d) \end{aligned} \tag{9}$$

in which the first term is dropped out because it does not involve phishing webpage d, and can be regarded as a constant here.

Maximum likelihood estimator is often employed to learn $p(w|q)$ and $p(w|d)$. However, it suffers from data sparseness, leading to a zero estimated probability when a word is absent in the training corpus, especially for phish detection where the vocabulary on phishing webpages is rather limited. As a remedy, we applied smoothing, a core concept in language modeling to reallocate probabilities to avoid zero counts, and assigned to an unseen word a probability proportional to its overall frequency in a background language collection.

$$p(w|d) = \begin{cases} p_s(w|d) & \text{if word } w \text{ is seen} \\ \alpha_d p(w|\mathcal{C}) & \text{otherwise} \end{cases} \tag{10}$$

where α_d is a document-dependent constant, and $p(w|\mathcal{C})$ is a background language model.

With some derivation, the KL divergence in Eq (9) reduces to

$$KL(q \parallel d) \propto -\Big(\sum_{w} p(w|q) \log \frac{p_s(w|d)}{\alpha_d p(w|\mathcal{C})} + \log \alpha_d\Big) \tag{11}$$

Smoothed by a Dirichlet prior [30], [31], the model for seen words $p_s(w|d)$ becomes $p_s(w|d) = \frac{tf(w,d)+\mu p(w|\mathcal{C})}{|d|+\mu}$ and the document-dependent constant translates to $\alpha_d = \frac{\mu}{\mu+|d|}$, which guarantees that $p(w|d)$ is still a valid distribution. $tf(w,d)$ denotes the number of occurrences of w in d.

Plugging $p_s(w|d)$ and α_d into Eq (11), we obtain the KL divergence scoring function as in Eq (5)

$$KL(q \parallel d) \propto -\Big(\sum_{\substack{w: tf(w,q)>0 \\ tf(w,d)>0}} p(w|q) \log \big(1 + \frac{tf(w,d)}{\mu p(w|\mathcal{C})}\big) + \log \frac{\mu}{\mu + |d|}\Big)$$

$p(w|q)$ and $p(w|\mathcal{C})$ are learned by maximum likelihood estimation given by

$$p(w_i|q) = \frac{tf(w_i,q)}{\sum_{w_j \in q} tf(w_j,q)} \tag{12}$$

$$p(w_i|\mathcal{C}) = \frac{\sum_{d_j \in \mathcal{C}} tf(w_i,d_j)}{\sum_{d_j \in \mathcal{C}} \sum_{w_k \in d_j} tf(w_k,d_j)} \tag{13}$$

where μ is a parameter in the Dirichlet prior and is tuned via cross validation in our experiment.