SemanticPhish: A Semantic-based Scanning System for Early Detection of Phishing Attacks

Qian Cui, Guy-Vincent Jourdan, Gregor v. Bochmann
University of Ottawa,
Ottawa, Canada
cuibuaa@gmail.com,{gjourdan, bochmann}@uottawa.ca

Iosif-Viorel Onut
IBM Centre for Advanced Studies
Ottawa, Canada
vioonut@ca.ibm.com

Abstract—**In the fight against phishing attacks, time is of the essence. Each individual attack is usually short-lived, but many people are still victimized during that short timeframe. To curb the problem, one way is to detect the attack shortly after the site is deployed, before victims have a chance to access it. Monitoring every new URL on the internet clearly is not a practical option, but monitoring sites that have a good chance of hosting an attack can be done. One of the ways to spot such a site is to monitor domain names. It is known that a growing number of phishing attacks are hosted by the attacker [1], [2], using their own domain names. Therefore, domain names might help spotting likely attacks. In this paper, we look at the following questions: can we currently tell apart domain names used in phishing attacks from other domains? If so, can we train a system to automatically detect these domains? And can such a system find attacks before they are being reported by victims? We show that the semantic of the words used by many phishing domains is different from the semantic of the words used by benign domain names, and that we can train a classifier to reliably flag these domains. We propose a system, *SemanticPhish*, which efficiently monitors these domains and is able to detect many phishing attacks without requiring the attack to be reported first. *SemanticPhish* can find attacks several days before Google's "safe browsing" starts flagging them.**

1. Introduction

A "phishing" site is a web site impersonating another, legitimate web site, put together by a "phisher" in order to entice end-users to disclose sensitive information meant for the impersonated site. The current anti-phishing solutions mainly focus on "late" detection: the URL of the attack has to be disclosed to the detection system for it work. This means there is a delay between an attacker launching an attack and the attack being detected, and an opportunity for the attack to be effective for some time before being detected. According to a the Anti-Phishing Working Group (APWG) [3], phishing attack instances are blocked after about 10 hours in average. Although this is not long, it still provides a window of time for attackers to collect information.

One way to detect some phishing attacks at an earlier stage is to start to find out about the attack before it is advertised to the victims. One way to achieve this is to monitor web sites "blindly" in order to find new attacks. This is however unpractical, given the large number of domain names being activated every day. To make this approach more practical, we need a way to narrow down the range of domains being monitored, and focus on the ones that are much more likely to eventually host an attack.

In this paper, we explore the possibility to use the semantics of domain names, and the semantic differences between domains used to launch phishing attacks and legitimate domains names to detect sites that are worth monitoring. Note that phishing attacks can be hosted by the attacker on a domain that they own, which is the situation of interest to us here, or they can be hosted on a compromised server, in which case the domain name is not related to the attack. According to [1], [2], a growing number of phishing attacks are hosted by the attacker, and currently represent 38 to 49% of the attacks. To avoid confusion, in the following we refer to a phishing domain owned by the attacker as a malicious domain.

We look at the following research questions:

- **RQ1**: Is there a noticeable difference in the strings used to create malicious domains when compared to legitimate ones?
- **RQ2**: If the answer to RQ1 is true, then can we train a system to distinguish malicious and legitimate domains with a reasonable accuracy?
- **RQ3**: If the answer to RQ2 is true, then can such a system be used to effectively detect some phishing attacks significantly earlier than they are today, in particular before there is any evidence that the attack itself is active?

To answer RQ1, we propose a model which creates vectors based on the semantics of the words used in malicious and legitimate domain, and show that these vectors can be separated by hyperplans in the vector space.

To answer RQ2, we train a machine-learning model using the vectorization of RQ1 and achieve an accuracy of almost 84% despite working with a noisy dataset.

To answer RQ3, we propose *SemanticPhish*, a system that prioritizes the domains that are most likely malicious in order to monitor them. We show that *SemanticPhish* can indeed detect attacks without being provided any URL. We also compare our detection results with Google's live update blacklist service: Safe Browsing. Our results show that around 70% of the attacks detected by *SemanticPhish* are still not reported by Google Safe Browsing 5 days after *SemanticPhish*'s detection.

We stress that the goal of this research is **not** to propose a new, general phishing attack detection system. As already mentioned, not all phishing attacks occur on a domain owned by the attacker. Many attacks are hosted on domains that have been compromised, or on general hosting sites. And attacks using domains owned by the attacker do not always try to lure victims by using semantically loaded words in the URL, and even those who do sometimes use the path and not the domain name to do it. What is more, some legitimate domains also use similar words. The goal of **RQ1** is to study whether we can use word semantics to flag **some** of the phishing domains that do fall into the category of interest to us, not to flag every phishing domain, let alone every phishing attack. **RQ2** and **RQ3** will tell us if it is worth monitoring these particular domains to detect phishing attacks early.

The paper is organized as follows: In Section 2, to answer RQ1, we introduce our domain words model and conduct a series of analyses comparing malicious and legitimate domains. Then in Section 3, we discuss three machine learning models that can be used to identify malicious domains, therefore answering RQ2. The discussion about *SemanticPhish* is presented in Section 4. We apply *SemanticPhish* on a live stream of domain names log to answer RQ3. We provide an overview of the literature in Section 5 before the conclusion in Section 7.

All of the source codes and the data used in this paper can be found at http://ssrg.site.uottawa.ca/SemanticPhish.

2. Domain Words Model (RQ1)

We first answer RQ1: *Is there a noticeable difference in the strings used to create malicious domains when compared to legitimate ones*?

In order to convince a victim to respond to a phishing attack, an attacker not only creates phishing pages that mimic legitimate pages, but often also uses an URL that appears to be legitimate. When the phisher owns the domain name hosting the attack, then that domain can also be used to convey some information about what the site is supposed to be about. When this is the case, attackers usually use one of two strategies when choosing malicious domain names: they can be related to the target of the attack, or they can be related to the content of the attack. The target-related domain uses similar words or homographs of the target domain, e.g., "apple-id.xyz" or "äpple.com". When it comes to the content-related domain name, we can rely on the fact that phishing attacks usually target social network and e-commerce sites. Therefore, words used in phishing domains that are meant to convey the content of the site are biased towards a specific corpus of words which is different from regular websites. This provides an opportunity to identify these phishing domains through the semantics of the words used to construct these domain names.

Our approach is to combine natural language processing and a model created by machine learning to build a word model which is able to "understand" the semantics of domain names. We proceed in three steps: the domain canonicalization, the words parser, and finally the words vectorization. Figure 1 illustrates this with the domain *UpdatϵYourÀccθunt.ga*.

2.1. Domain Canonicalization

The first step is the *domain canonicalization*. The so-called *homograph attacks* [4] use visually confusing text rendering to create domains that can be easily misread by the victim and confused for something else. In our case, we use the Unicode Technical Report #36 from Unicode Technical Standard[1] to create a characters replacement table and obtain a domain name with only digit and English letters. For example, the domain *UpdatϵYourÀccθunt.ga* would be changed to *updateyouraccount.ga*.

We then remove the domain TLD as well as some common sub-domain names, such as www, mail, cpanel, webmail and webdisk, that do not carry meaningful semantics for the analysis. Since some domains are subdomains of common host providers such as myshopify, sharepoint or wordpress, we also remove several of these well-known second level domains [2]. The resulting text is the input to the **Words Parser**. The complete list can be found on our website http://ssrg.site.uottawa.ca/SemanticPhish).

2.2. Words Parser

A straightforward way to extract words from a domain name would be to use a predefined list of separators such as "-" and ".". However, such a method is too restrictive, and would fail if the domain name consists of multiple consecutive words. In order to effectively split a string into a list of words, we apply Zipf's law [5] to infer the words position. Specifically, Zipf's law found that occurrence frequency of a word times its rank in a given frequency table is equal to a constant which is linearly related to # of words in the frequency table. Formally, $freq(w) * r_w = aN$, where r_w is the rank of the word w in the frequency table, and N is the number of words in the dictionary. The frequency table is compiled by counting the frequency of words appearing in a large corpus of text and sorting the words in descending order by frequency. As an example, Table 1 shows a 10-word frequency table for a dictionary of 10 words ($a = 1$). The higher the rank of a word (lower index) in the table, the higher the occurrence frequency of that word. Once the frequency table is compiled, we use dynamic programming

1. http://www.unicode.org/reports/tr36
2. See http://ssrg.site.uottawa.ca/SemanticPhish for the complete list.

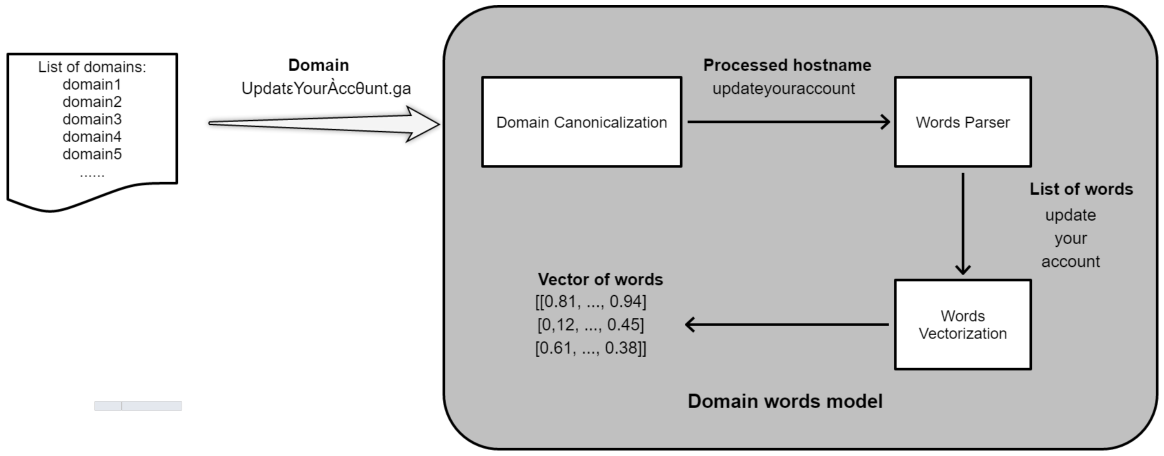


Figure 1. Structure of the domain words model

to infer word position. The resulting word list is the one that maximizes the frequency of each individual word.

Word	Frequency rank	Occurrence probability
apple	1	10.0
security	2	5
paypal	3	3.3
update	4	2.5
account	5	2.0
your	6	1.6
user	7	1.4
service	8	1.3
verify	9	1.1
lock	10	1.0

TABLE 1. EXAMPLE OF A FREQUENCY TABLE

We first split the domain name into multiple text segments by using the separator dot ("."). We then remove non-alphabetic characters in each text segment and apply the above words parser to the extracted words. The implementation of our words parser is based on the Python library [6], which uses a frequency table of over 126,000 "words" created from English Wikipedia. The list of words extracted from each segment is combined as the input to the **Words Vectorization** step.

2.3. Words Vectorization

Word Embedding is a technique that converts a word into a high-dimensional vector. In our context, we are interested in the semantics similarity of words. In general, words with similar meanings will be close to each other in the vector space. For instance, the words "car" and "truck" have two similar vectors in the word vector space because they are instances of the same category. Training such a word vector model requires very powerful computing resources and a large collection of texts. We use a pre-trained vector model provided by the Stanford Natural Language Processing Group[3], which has been trained using a large number of texts from Internet, including 2.2 million words. We use the intersection between the list provided by Stanford and the frequency table of *WordNinja* to convert domain names into a list of word vectors.

Another useful feature of word vectorization in our context is that we can create a vector for a string consisting of multiple words by adding the vectors of each word of that string. This provides us a tool to measure the semantic similarity of domain names by comparing the vectors that we obtain for these domains: vectors that are similar tend to be made from domains that use semantically close words.

2.4. Analysis of the Model

2.4.1. Dataset.

In order to test our model, we need two datasets: a set of malicious domains and a set of legitimate ones. For the list of malicious domains, we use a dataset that was provided by PhishLabs[4] to the authors of [1]. This dataset contains almost 10,000 domain names that have been manually cured and is supposed to contain only malicious domains used in phishing attacks (see [1] for additional details about the construction and cleaning of this dataset). Note that despite the cleaning efforts, this dataset of malicious domains still has some noise and contains a number of legitimate domains.

We created the list of legitimate domain names by randomly selecting domains ranked between 10,000 to 1 million in Alexa Top 1M[5]. We chose to exclude the domains in the top 10K because we needed a source of "normal" domains which are representative of the average newly created legitimate website. Moreover, the domains in the top 10K are most frequently targeted by phishing sites and thus are not a good training set to extract the legitimate keywords that are not often used by phishing sites. After removing duplicates with the same hostname, we ended up with 9,768 malicious domains and 19,976 legitimate ones

3. https://nlp.stanford.edu/projects/glove/

4. https://www.phishlabs.com/
5. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

as our ground truth data. In the following sections, we use this dataset for various experiments.

2.4.2. Semantic Similarity Analysis.

In order to analyze the semantic similarity between domains, we build what we call the **domain connection graph**. Specifically, we use the cosine similarity as the metric to assess the similarity between word vectors. If the absolute value of the cosine similarity between two domains is more than the empirical threshold of 0.8, an edge is built in the graph. It is noted that this threshold is only used for illustration purposes, in order to limit the number of edges in the visual rendering of the domain connection graph. Another threshold could have been used. It is not used by our system and thus does not impact our results. We apply this strategy to both of our datasets. If a domain has no connection to any other domain in the dataset, it is dropped from the graph.

The domain connection graphs of both domain datasets are shown in Figure 2. The malicious domain connection graph consists of 5,136 domains (52.58% of the malicious dataset), whereas the legitimate domain connection graph only includes 2,992 domains (14.98% of the legitimate dataset). This indicates that it is much more common for malicious domains to have some level of semantic similarity with other malicious domains than it is for legitimate domains. The figure also shows clearly that there are more clusters in the malicious domain connection graph than in the legitimate domain one, which suggests that these domains can be detected by looking for clusters of semantically related malicious domains.

In Table 2, we provide a random sampling of domain names for the top five largest clusters in both datasets. It can be seen that the domains in the cluster of the malicious graph have similar semantics (e.g. domain names related to security for the first cluster), even though these domains only share few common words. On the other hand, the largest clusters of the legitimate domain graph are only connected because the domains either use exactly the same word (cluster 2 to 5) or, in the case of the top cluster, because they use only one letter between digits (the letter "l" in the case of that cluster), which causes our vectorization algorithm to group them together.

2.4.3. Domain Words Analysis.

In the above analysis, our results indicate that some of the malicious domains are created based on a set of topics, such as "software security" or "account update". But it is still unclear whether the words used by malicious domains are limited to a narrow range. In order to answer this question, we build a **word-domain** graph to analyze the correlation between words and domains. Specifically, we first sort words based on the number of domain name they cover in decreasing order. During this process, we ignore single characters and stop words. We then draw a cumulative graph to show the correlation between # of words and # of domains covered. The result is shown Figure 3. 50% of the malicious domains are covered by only 0.51% of the words in that dataset, while 3.33% of the words are required in the legitimate dataset. The 80% and 90% range is reached after 6.9% and 17.65% of the words in the malicious dataset, while 17.31% and 36.54% of the words are required in the legitimate dataset. If we look at the boxplot between # of words used in phishing domains and legitimate domains, as shown in Figure 4, we can see that phishing domains tend to use more words than legitimate domains, although the median values of both sets are very close

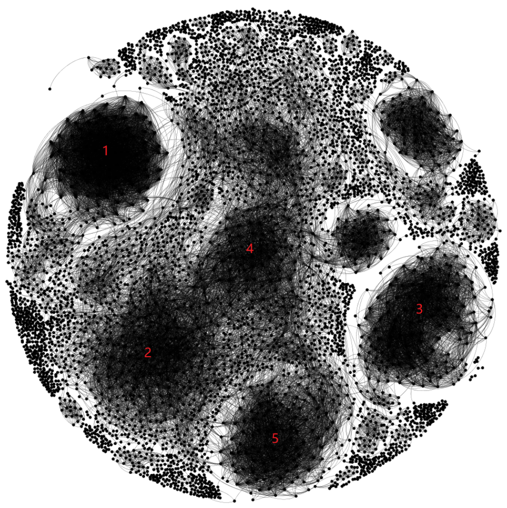
The results indicate malicious domains tend to use fewer but more semantically similar words than legitimate domains, and thus the answer to RQ1 is **yes**. It is however still possible that this model can only capture words that have been seen in the training set, that is, words that happen to appear in our ground truth dataset. It is also not clear that even though the two dataset do not have the same behaviour when it comes to semantic similarity, that difference can be used to separate the two sets apart. The latter is the topic of RQ2, which we address next. We will then be able to address the former, using the classifier to create a completely new dataset of malicious domain and show that the set of words in the newly created malicious dataset is quite different from the set of word in the ground truth malicious dataset.

3. Machine Learning Classifier (RQ2)

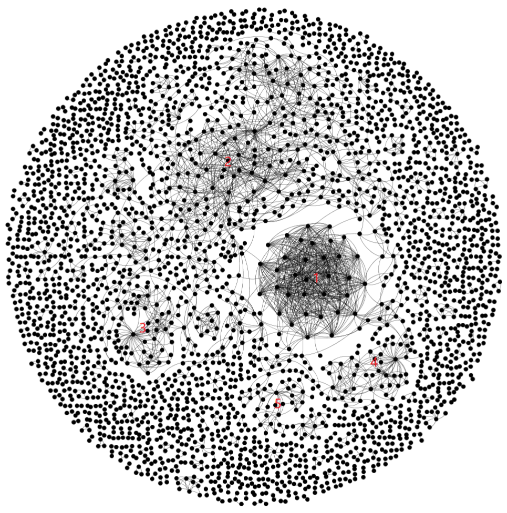
We now focus on RQ2: *If the answer to RQ1 is true, then can we train a system to distinguish malicious and legitimate domains with a reasonable accuracy?*

In order to answer this, we have tested three machine learning algorithms that are often used for binary classification: AdaBoost (ADA), Random Forest (RF), and Support Vector Machine (SVM). AdaBoost is a machine learning framework, which combines the results of multiple weak learners (for example, decision trees) into a weighted output, therefore improving the performance of the weak learners. The Random Forest algorithm uses a similar idea: it constructs multiple decision trees during the training process. Each decision tree uses a random subset of features. The model output is a combination of the results of the individual trees (e.g. majority voting, mean value etc.). The goal of the SVM algorithm is to search for the hyperplane that "best" separates the two classes. Specifically, it first uses a kernel function to project the input data into a high-dimensional space, and then searches for the hyperplane with the maximum distance to the nearest points on each side. Since the output of standard SVMs is a distance to the hyperplane rather than a classification probability, we use **Platt calibration** [7] to transform the distance into a probability over classes. All these machine learning models are implemented by scikit-learn[6] with the default parameters, as shown in Table 3. The input of these classifiers is the embedding vector of domain names discussed in Section 2.3. In other words, these classifiers tend to identify phishing domains

6. https://scikit-learn.org/stable/index.html



(a) Malicious domain connection graph



(b) Legitimate domain connection graph

Figure 2. Domain connection graphs (generated by Gephi using layout ForceAtlas 2)

Cluster label	Malicious	Legitimate
1	a)service-appteamsupport.support b)macsoftwareinternalstorageappleerrorcodesecurewaringalert.xyz c)securesoftwarestorageinternalwaringalertcode0978.xyz	a)2007l04.com b)l25.ir c)l495b9.com
2	a)verifyaccount-unlockedsid.tk b)manageaccount.ga c)resolvemyaccount-locked.com	a)mygobe.com b)gowesgo.com c)letgo.cz
3	a)wellsfargo-43043l33.com b)wells-fargo-profile-l430l023.com c)wellsfargocards.net	a)online4.love b)sudonline.sn c)onlinevsem.ru
4	a)securitycentre-appleid.com b)applehomesecure.com c)appleid-fraud-operations.com	a)pro.com b)date-pro.com c)tspro.com.br
5	a)service-account-billing-information.com b)recoveryidinformation.com c)security-informationpayment-apple.com	a)itsfree.club b)gofreeapp.net c)freeexe.net

TABLE 2. Example domains in labeled clusters

and legitimate domains by learning semantics meaning from word embedding.

Model	Hyperparameters
AdaBoost	base_estimator=DecisionTree, n_estimators=50, learning_rate=1, algorithm=SAMME.R
Random Forest	n_estimators=10, criterion=gini
SVM	C=1.0, kernel=rbf, degree=3, gamma=auto_deprecated, coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200

TABLE 3. Model Hyperparameters

We compare the performance of these three models, and list our results in the following section.

3.1. Classifier Comparison

We first compare the results between the three classifiers. We apply a 4-fold cross validation, and report the average of the results in Table 4. It can be seen that the SVM classifier yields the best accuracy (83.96%), and the RF classifier yields the best false positives (1.15%).

Model	Accuracy	False Positive
AdaBoost (ADA)	78.01%	12.66%
Random Forest (RF)	81.32%	**1.15%**
Support Vector Machine (SVM)	**83.96%**	3.05%

TABLE 4. Comparison between three classifiers

In order to further compare the performance between these three classifiers, we draw the Receiver Operating Characteristic (ROC) curve, which shows the relation between true positives against the false positives at various threshold, and use the Area Under The Curve (AUC) to evaluate the performance: the greater, the better. The result is shown Figure 5. We can see that the SVM classifier performs

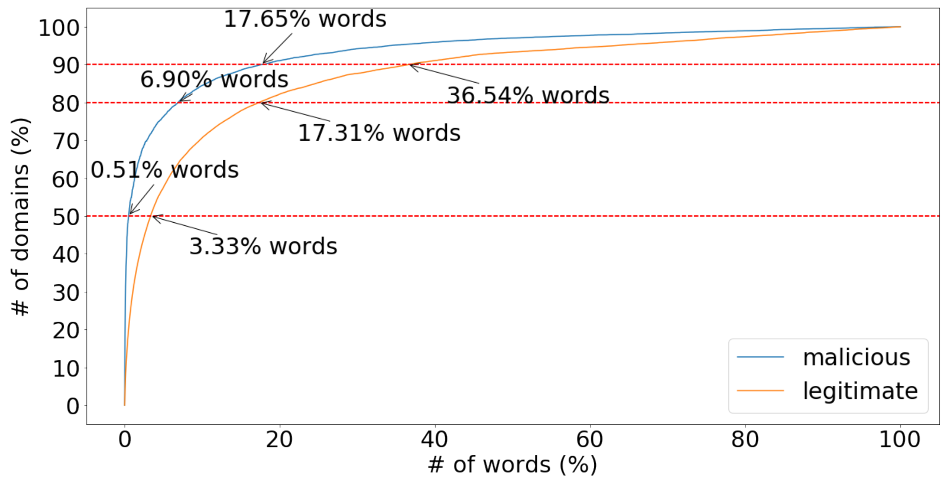


Figure 3. Word-domain Graph

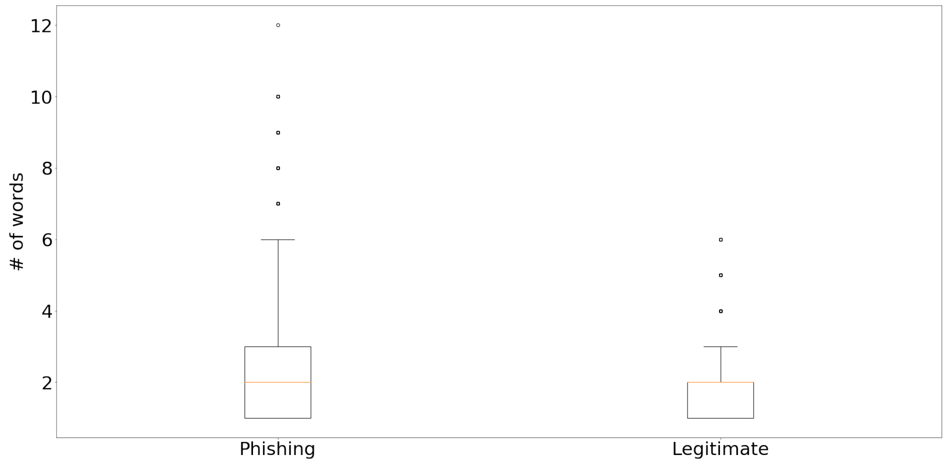


Figure 4. Word Boxplot

better than the other two models, with a relatively high AUC (0.88). In addition to performing better, SVM also provides a more natural interpretation of the score. In our case, for malicious domains, the further away from the hyperlane the sample is, the higher the score. The opposite is true for legitimate domains: the further away from the hyperplane, the lower the score. Therefore, in the following experiments, we use SVM as our classifier.

Note that the results are far from perfect, the best accuracy being only around 84%. This is in fact to be expected: what we are trying to detect is some distinguishing features in the semantics of the words used to create malicious domain names. What we show is that the vast majority of these domains do exhibit such detectable characteristics. Of course, not **all** malicious domains will have these characteristics, and therefore we cannot expect such a classifier to find all malicious domains.

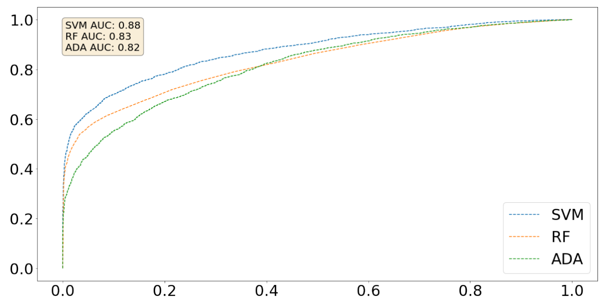


Figure 5. ROC of three classifiers

3.2. Flexibility of the Domain Words Model

As mentioned in Section 2.4.3, it is possible that our model can detect domains that use words appearing in our training set, but fail to recognize words with similar semantics which it has not seen before. In order to find out, we compile a dataset of 47,876 unique phishing domains that have been collected from the community-driven portal PhishTank[7], the cloud-based threat intelligence sharing platform IBM X-Force[8], and the phishing intelligence analysis platform OpenPhish[9] between January 1, 2016 and January 1, 2018. These phishing attacks contain a mix of malicious and compromised domains. We applied the SVM classifier obtained from our initial dataset as described in Section 3.1 on these 47,876 domains. 9,730 of these domains obtained a score greater than 0.5 and are flagged malicious.

To avoid confusion, we call M_0 the set of 9,768 malicious domains and L_0 the set of 19,976 legitimate domains used for training (Section 3.1) and M_1 the set of 9,730 flagged malicious domain obtained from the SVM classifier.

We want to see the similarity between M_0 and M_1. We will also compare M_0 and L_0. We use two similarity metrics for this: the Jaccard index and the histogram intersection. The Jaccard index is defined as the number of items in the intersection between two sets divided by number of items in the union of the same two sets. The histogram intersection calculates the similarity between two histograms. In our case, it is used for the comparison between the word score distribution graphs of M_0 and M_1, and M_0 and L_0. The word score distribution graph is a histogram where the bar in each bin represents the percentage of words in a range of scores. Our results are shown in Table 5 and Figure 6.

Phishing sets M_0 and M_1 end up with very comparable number of domains as well as number of words. The phishing domains in these sets are very different, with a Jaccard index of only 0.000011, but they do share a fair amount of similar words (the Jaccard index is 0.32). When comparing M_0 with L_0, we can see that legitimate domains and phishing domains share fewer common words. If we look at the average score of the words the sets do share, we can see that shared words between the phishing sets score on average much higher than between phishing and legitimate sets (0.34 vs 0.22). Figure 6 (a) shows the details of the Jaccard index comparison. We can see that the two word sets have higher similarity for the words in the high score bin. Specifically, the sets of words with scores greater than 0.8 have a Jaccard index greater than 0.7. This indicates that our semantic-based word model can detect malicious domains that use different words but have similar semantics. This would not be possible with a keyword-based word model using a predefined list of keywords. Figure 6 (b) compares the word score distribution between M_0, M_1 and L_0. We can see that both phishing domain sets have a similar distribution, with the majority of the words falling into the low score bin, [0-0.2]. This is why they have a high histogram intersection at 0.95. Despite this, our model can still detect phishing domains, by learning the few keywords that are specific to them. Looking at the word score distribution between M_0 and L_0, it can be seen that legitimate domains tend to use words with lower scores, while phishing domains uses more “malicous” words with high scores, which yields a lower histogram intersection.

7. https://www.phishtank.com
8. https://www.ibm.com/security/xforce
9. https://openphish.com/

4. Semantic-based Scanning System: *SemanticPhish* (RQ3)

We now turn our attention to RQ3: *If the answer to RQ2 is true, then can such a system be used to effectively detect some phishing attacks significantly earlier than they are today, in particular before there is any evidence that the attack itself is active?*

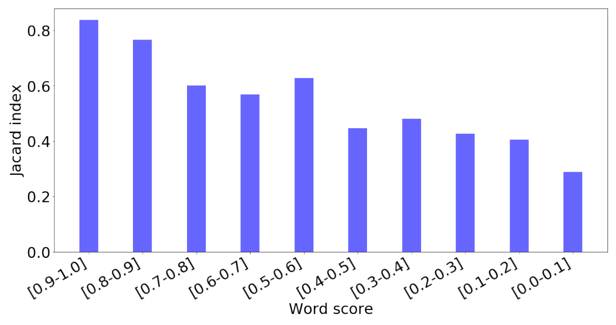
4.1. System Design of *SemanticPhish*

To answer RQ3, we propose *SemanticPhish*, a system for monitoring and detecting phishing attacks at an early stage. The architecture of *SemanticPhish* is shown in Figure 7. In addition to the domain words model and machine learning classifier discussed in the previous sections, there are three other components, responsible for crawling, detecting and verifying.

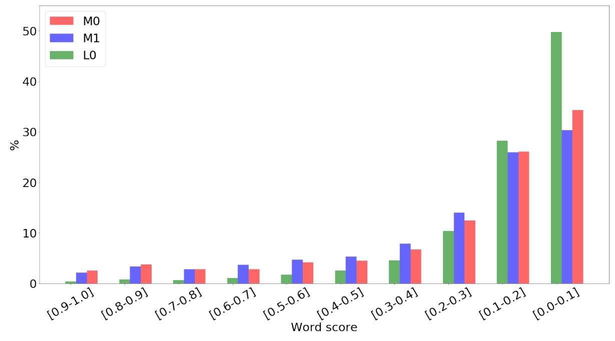
- **Crawling Scheduler**. As discussed in Section 3, the output of the classifier is a score between 0 and 1. In order to efficiently scan a large number of domains, we divide the score range into 10 intervals of length 0.1, corresponding to 10 buckets. We distribute the domains into these buckets based on their score. The priority of the bucket is based on the range of scores it covers. The crawling scheduler assigns crawlers to scan each bucket, starting from the bucket with the highest priority. The crawlers simply scan all the domains in the buckets, and move to the next bucket down when finished. Each time a higher priority bucket is updated (new domains are added), the crawling scheduler sends the crawlers back to that buckets, and the new domains are scanned. The crawlers then resume scanning the lower priority bucket.
  For each domain scanned, the crawler attempts to reach the domain’s root path using http and https protocols, and stores the DOM as well as the page screenshot when one is returned. If there are redirects between the root path and the final URL, the crawler logs the redirect path. Finally, the stored page information is then passed to the **Phishing Detection Model**.
- **Phishing Detection Model**. The goal of the phishing detection model is to identify phishing attacks using any existing phishing attack detection approach. In our experiments, we used [8] as the detection model. Specifically, we first extract the so-called “tag vector”. The tag vector simply counts the number of occurrences of each possible HTML tag in the DOM. We then compare the similarity between that vector and a set of similarly constructed vectors obtained from known phishing attacks. If the distance is less than a specific threshold, the page

Comparison entry	M_0	M_1	L_0
# of domains	9,768	9,730	19,976
# of words	5,274	5,717	10,559
Jaccard index of domain sets	-	0.00011	0
Jaccard index of word sets	-	0.32	0.14
Average score of common words	-	0.34	0.22
Histogram intersection of word score distribution graph	-	0.95	0.82

TABLE 5. COMPARING M_0 WITH M_1, AND M_0 WITH L_0



(a) Jaccard index across word score between M_0 and M_1



(b) Word score distribution of M_0, M_1 and L_0

Figure 6. Similarity comparison

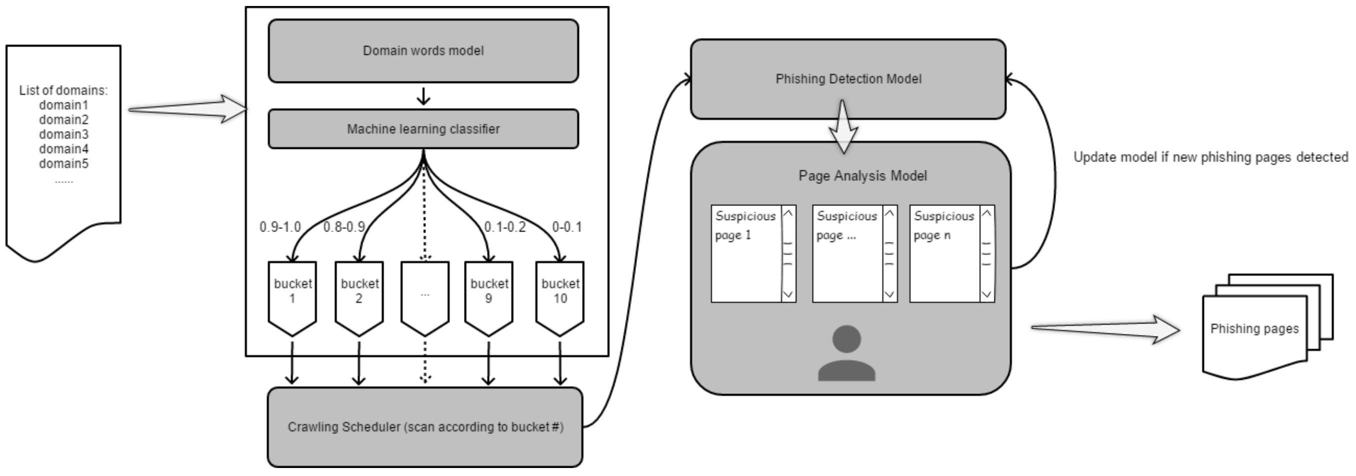


Figure 7. Architecture of *SemanticPhish*

is flagged as phishing. It should be noted that many different detection models can be used instead. The one presented here is chosen because it is fast to compute and easy to deploy.

- **Page Analysis Model** Since the phishing detection model may miss zero-day attacks or yield false positives, we include a manual verification step to our model. In the page analysis model, we list the detailed information of the page that have been scanned, for manual analysis and validation. In order to reduce the duplicates, we group up the pages that have the same tag vector. The phishing page that are validated are output as marked as confirmed phishing attack. The phishing detection model is updated if necessary. This manual verification step is optional and can be adapted to the level of trust one has in the phishing detection model. We have used it systematically here to ensure that the number reported in this paper are accurate.

In *SemanticPhish*, there are two models that need training before being used for detection: the machine learning classifier and the phishing detection model. The classifier is trained as explained in Section 2.4.1. The training data for the phishing detection model is an incremental list of phishing attacks that are updated daily from various phishing blacklist feeds, including https://www.phishtank.com and https://openphish.com/.

4.2. Evaluation of *SemanticPhish*

The goal of *SemanticPhish* is to detect phishing attacks early. In order to evaluate the detection efficiency of *SemanticPhish* in practice, we apply *SemanticPhish* on a real-time stream of domain logs. It is worth noting that the ideal input for *SemanticPhish* is a list of the domains that have recently become active, which allows *SemanticPhish* to start monitoring and detecting as early as possible. However, we do not have access to such a source. We adopt an alternative and use the certificate transparency logs (CT logs) provided by CertStream[10]. Certificate transparency is an open framework that monitors and audits TLS/SSL certificates in real time[11], and is used by many domain providers. The idea of certificate transparency is that each time a domain updates its certificate, a log about the transaction will be submitted to the certificate transparency network. That network is accessible to domain owners, CAs and domain users. Therefore, the domains that are in CT logs are domains for which a certificate has been issued, which includes domains that have had their certificate renewed, and well as new domains that have an TLS/SSL certificate issued for the first time.

4.2.1. Efficiency of Filtering Malicious Domains.

One advantage of *SemanticPhish* is that it is able to initially filter out potential malicious domains only using domains names rather than collecting other information which would be much slower. To assess the efficiency of filtering malicious domain names, we used *SemanticPhish* to track CT logs between January 29, 2019 and February 6, 2019. Since the size of stream CT logs, roughly 1.8 million domains per day at the time, was too large for our server to handle, we randomly sampled around 1% of the logs per hour.

Our results are shown Table 6. In total, we have scanned 109,694 domains, and only a few domains fall in high score bins, e.g., 0.23% in 0.9-1.0 range and 0.78% in 0.8-0.9 range. This is completely expected since the vast majority of the domains in CT logs are regular, benign sites and should not have a high score in our domain words model. 66,295 of the 109,694 domains were reachable by our crawlers at that time. Again, that is not really surprising, since site owners might acquire certificates for their domains before launching their site. Of these, we have 11 verified malicious domains detected, 4 of which fall in the score range 0.8-1.0, which is 2.51% of the domains in that range. The phishing domains with low scores either do not use English words (e.g., scure.anhaengerkalberer.ch or teb-cepte.com), or use misspelled words (e.g., welocmweincre.com or deveryservices.gb.net).

This indicates that our *SemanticPhish* is able to effectively filter out malicious domains from large-scale domain logs. What is more, it only takes on average 8 milliseconds on our machine to compute the score of a domain. A system such as *SemanticPhish* can thus easily be integrated into existing detection systems.

10. https://certstream.calidog.io
11. http://www.certificate-transparency.org/

4.2.2. Efficiency of Detecting Early Phishing Attacks.

Our previous experiment shows that *SemanticPhish* is efficient at filtering malicious domains. In this section, we assess the performance of *SemanticPhish* in detecting phishing attacks early.

We again tracked the stream of CT logs between May 30, 2019 and June 5, 2019. We only kept the domains with a score of 0.9 or higher, that is, highly suspicious domains. We re-scanned these domains every 6 hours for 5 days in order to detect any change in their status. When changes were detected (e.g. new DOM and new landing URL), we updated our database.

We choose Google Safe Browsing (GSB)[12] as the baseline for our comparison because it is widely used by major browsers, such as Chrome, Firefox and Safari. Since the list of phishing sites provided by GSB is updated dynamically, we verified the list of our domains daily with GSB and collected the domains that were flagged, as well as when they were flagged.

Our results are shown Table 7. Overall, we have scanned 21,131 domains, and 14,546 (68.84%) of them are reachable. Of all the scanned domains, 766 domains are detected as malicious by GSB and/or by *SemanticPhish*. *SemanticPhish* detects 361 of these domains, **250 of which are still not flagged by GSB five days after *SemanticPhish*'s detection**.

When looking at these 250 domains, we find that many of them belong to series of related phishing domains, and GSB only flags a few of the domains in the series. For instance, we had 72 domains following the pattern webmail-client*xx*.dns05.com, where *xx* is a number between 0 and 100. Only 4 of these 72 domains were flagged by GSB.

We also found series of malicious domains that use the same hostname but different TLD. Again, only a few of them were flagged by GSB. These results indicate that the attacker tends to deploy phishing attacks on multiple domains using similar names. *SemanticPhish* is able to flag such phishing attacks at an early stage: whenever the phishing domain appears in the monitoring log, the attack starts being monitored and can be flagged as soon as the phishing website is activated.

For the 516 domains reported to be malicious by GSB, 182 domains were never reachable by our crawlers throughout the experiment and we were thus unable to confirm or refute GSB's verdict. Another 223 domains were not confirmed as malicious by *SemanticPhish*: these domains either host blank pages or are parked with advertisement pages. We cannot tell if these are incorrect verdicts from GSB, or if these domains were indeed phishing attacks that were already taken down by the time we reached them (e.g. some malicious domains may have already been detected or blocked before the certificate was issued for example). The

12. https://safebrowsing.google.com/

Score range	**# of scanned domains (%)**	**# of reachable domains**	**# of domains verified as phishing (% of reachable domains)**
0.9 - 1	257(0.23%)	135	3 (2.22%)
0.8 - 0.9	855(0.78%)	339	1 (0.29%)
0.7 - 0.8	2,150(1.96%)	749	0
0.6 - 0.7	3,822(3.48%)	1,389	2(0.14%)
0.5 - 0.6	5,828(5.31%)	2,673	1(0.04%)
0.4 - 0.5	8,297(7.56%)	4,050	2(0.05%)
0.3 - 0.4	12,525(11.42%)	7,034	0
0.2 - 0.3	18,675(17.02%)	11,106	2(0.01%)
0.1 - 0.2	26,276(23.95%)	17,094	0
0 - 0.1	31,009(28.27%)	21,726	0

TABLE 6. DOMAIN SCORE DISTRIBUTION OF SAMPLED CT LOGS

remaining 111 domains are identified as malicious domains by both GSB and *SemanticPhish*. 31 of them are flagged by GSB one day or more after being first detected by *SemanticPhish*. Our results show that *SemanticPhish* can indeed detect many phishing attacks days before they are detected by GSB, at a very early stage. Adding a system such as *SemanticPhish* to the series of detection tools being used can greatly reduce the lifespan of phishing attacks.

5. Related Work

There exists a significant body of academic work focusing on phishing attacks detection. There are three main approaches that have been suggested.

The first one is to detect phishing attacks by comparing it with its target. Rosiello et al. [9] present a browser extension using the similarity of a DOM tree to detect phishing attacks. The idea of this work is to store a mapping between the user's sensitive information and the legitimate sites that use the information. When the same sensitive information is reused on a different site, the extension checks whether the DOM of a new site is similar to the associated one.

Several studies apply visual similarity comparison to detect phishing attacks. Chen et al. [10] applied the Gestalt Theory to perform a comparison of visual similarity by using normalized compression distance (NCD) as the similarity metric. Sites logo [11] and favicon [12] comparison have also been suggested. Some authors have suggested to use search engines to acquire this knowledge automatically, for example Cantina [13] which attempts to find the current page on Google and warns if it is not found. Similarly, Huh et al. [14] suggested to search the site's URL in different search engines and use the number of returned pages as an indicator of phishing.

The second approach is to attempt to use machine learning to discover the intrinsic characteristics of phishing attacks. The goal of these approaches is to train a binary classifier by learning the relations between data features and the ground truth (phishing or legitimate). Cantina+ [15] proposes a system using Bayesian Network mixing 15 features. Gowtham et al. [16] proposed a detection system using a Support Vector Machines (SVM) classifier and similar features to Cantina+. Their system achieved 99.65% true positive and 0.42% false positive. Daisuke et al. [17] conducted an evaluation of nine machine learning-based methods; in their study, AdaBoost provided the best performance.

Finally, there are a number of new approaches and directions that have been proposed to detect phishing attacks. In [8], it is shown that most phishing attacks are duplicates or variations of previously reported attacks. Thus, new attack instances can be detected using these similarities. Corona et al. [18] proposed a method to detect attacks hosted on compromised servers, which compares the page of the attack with the homepage that hosts it and the pages linked by it.

The most similar work is the studies [19]–[21]. Garera et al. [19] find that phishing attacks tend to build URLs using a limited set of words. They compile a list of sensitive words that typically appear in phishing URLs, such as login and signin. Kkhonji et al. [20] present a approach to detect phishing attacks by using lexical feature of phishing URLs. Marchal et al. [21] applies NLP technique to generate phishing blacklist.

However, all of the above approaches have to know the entry point of the phishing site before detecting it. Our solution fills this gap, and is able to blindly scan phishing attacks at an early stage.

6. Limitation and Future Work

Although our results show that *SemanticPhish* has good performance at detecting phishing attacks at early stage, there are several limitations that could be improved in future work.

- As we have shown in Section 4.2.1, *SemanticPhish* cannot effectively detect the domain names created using non-English words. This problem can be solved by using a word corpus in other languages.
- *SemanticPhish* currently only explores the root path of each suspicious domains. This prevents our model from detecting attacks that are hosted somewhere else. In future work, we plan to attempt scanning other paths, using known common locations for other attacks.
- A better domain source can help further improve system performance. Instead of using CT logs, we could use real-time domain logs from DNS servers, or even from registrars.

# scanned domains	21,131
...# reachable domains	14,546
# total malicious domains	766
Malicious domains detected by *SemanticPhish*	
# detected domains (manually confirmed)	361
...# detected malicious domains not reported by GSB	250
Malicious domains detected by GSB	
# detected malicious domains	516
...# unreachable domains	182
...# domains not confirmed by *SemanticPhish*	223
...# domains flagged by both *SemanticPhish* and GSB	111
...# domains detected at least 24h before GSB by *SemanticPhish*	31

TABLE 7. DETECTION RESULT COMPARISON BETWEEN *SemanticPhish* AND GSB

- In our experiments, we choose a 5-day observation period, and find that some suspicious domains have a longer incubation period, and become active after this observation period. Therefore, a longer monitoring period may help capture more attacks.
- One weakness of our proposed word vector-based approach is that it cannot differentiate strings that consist of same words but in different orders. For instance, the strings "business-security-company" and "security-business-company" end up with same vectors. In future work, we plan to use n-grams instead of single words to train the Word2Vec model.

7. Conclusion

In this paper, we have proposed a semantic-based scanning system, *SemanticPhish*, to detect some phishing attacks at an early stage. This system gives us an opportunity to track and monitor phishing activities before an attacker launches an attack. In order to effectively identify suspicious hosts that likely host phishing attacks, we have proposed a domain ranking model by analyzing the semantics of the domain name. Since this ranking model uses only a list of domain names as input, and does not require some time-consuming processing, such as crawling pages and collecting domain information, it allows our system to be easily integrated into existing systems.

Our results show that when compared to legitimate domains, some of the domains created for phishing attacks typically use smaller but more semantically similar word sets. This can be explained by the fact that the goal of these phishing domains is to deceive victims by imitating the domain name of a well-known website or using words that express something of interest to the attacker. What is more, we have observed that attackers often use multiple similar domain names to launch the same attack. Our model can be used to flag and monitor these domains, and ultimately block them immediately when a attack ends up being deployed there.

Finally, we compared our approach with Google Safe Browsing (GSB). We have also demonstrated that *SemanticPhish* can monitor and detect phishing attacks at an early stage. Specifically, around 70% of attacks identified by *SemanticPhish* are not reported by GSB within 5 days. Even for the remaining phishing domains that are detected by GSB and *SemanticPhish*, 28% are reported by GSB more than one day after being detected by *SemanticPhish*. All of the source codes and the data used in this paper can be found at http://ssrg.site.uottawa.ca/blindphish.

Acknowledgments

This research was supported in part by the IBM Center for Advanced Studies and by the Natural Sciences and Engineering Research Council of Canada.

References

[1] S. Le Page, G.-V. Jourdan, G. V. Bochmann, I.-V. Onut, and J. Flood, "Domain classifier: Compromised machines versus malicious registrations," in *International Conference on Web Engineering*. Springer, 2019, pp. 265–279.

[2] Anti-Phishing Working Group, "Global Phishing Survey: Trends and Domain Name Use in 2016," http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf, 2017.

[3] ——, "Global Phishing Report 2H 2014," http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2H_2014.pdf.

[4] E. Gabrilovich and A. Gontmakher, "The homograph attack," *Communications of the ACM*, vol. 45, no. 2, p. 128, 2002.

[5] S. T. Piantadosi, "Zipf's word frequency law in natural language: A critical review and future directions," *Psychonomic bulletin & review*, vol. 21, no. 5, pp. 1112–1130, 2014.

[6] D. Anderson, "Wordninja," https://github.com/keredson/wordninja.

[7] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[8] Q. Cui, G.-V. Jourdan, G. V. Bochmann, R. Couturier, and I.-V. Onut, "Tracking phishing attacks over time," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 667–676.

[9] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi, "A layout-similarity-based approach for detecting phishing pages," in *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks, SecureComm*, Nice, 2007, pp. 454–463.

[10] T.-C. Chen, S. Dick, and J. Miller, "Detecting visually similar web pages: Application to phishing detection," *ACM Trans. Internet Technol.*, vol. 10, no. 2, pp. 5:1–5:38, Jun. 2010.

[11] E. H. Chang, K. L. Chiew, S. N. Sze, and W. K. Tiong, “Phishing detection via identification of website identity,” in *2013 International Conference on IT Convergence and Security, ICITCS 2013*. IEEE, 2013, pp. 1–4.

[12] G.-G. Geng, X.-D. Lee, W. Wang, and S.-S. Tseng, “Favicon - a clue to phishing sites detection,” in *eCrime Researchers Summit (eCRS), 2013*, Sept 2013, pp. 1–10.

[13] Y. Zhang, J. Hong, and C. Lorrie, “Cantina: a content-based approach to detecting phishing web sites,” in *Proceedings of the 16th International Conference on World Wide Web*, Banff, AB, 2007, pp. 639–648.

[14] J. H. Huh and H. Kim, “Phishing detection with popular search engines: Simple and effective,” in *International Symposium on Foundations and Practice of Security*. Springer, 2011, pp. 194–207.

[15] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, “Cantina+: A feature-rich machine learning framework for detecting phishing web sites,” *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, pp. 21:1–21:28, Sep. 2011.

[16] R. Gowtham and I. Krishnamurthi, “A comprehensive and efficacious architecture for detecting phishing webpages,” *Computers & Security*, vol. 40, pp. 23–37, 2014.

[17] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, “An evaluation of machine learning-based methods for detection of phishing sites,” in *International Conference on Neural Information Processing*. Springer, 2008, pp. 539–546.

[18] I. Corona, B. Biggio, M. Contini, L. Piras, R. Corda, M. Mereu, G. Mureddu, D. Ariu, and F. Roli, “Deltaphish: Detecting phishing webpages in compromised websites,” in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 370–388.

[19] 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 - WORM ’07*. New York, NY: ACM, 2007, pp. 1–8.

[20] M. Khonji, Y. Iraqi, and A. Jones, “Lexical url analysis for discriminating phishing and legitimate websites,” in *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*. ACM, 2011, pp. 109–115.

[21] S. Marchal, J. François, T. Engel *et al.*, “Proactive discovery of phishing related domain names,” in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2012, pp. 190–209.