

# Enhancing Phishing E-Mail Classifiers: A Lexical URL Analysis Approach

Mahmoud Khonji<sup>1</sup>, Youssef Iraqi<sup>1</sup>, Andrew Jones<sup>2</sup>

Department of Computer Engineering, Khalifa University of Science, Technology and Research<sup>1</sup>

Department of Information Security, Khalifa University of Science, Technology and Research<sup>2</sup>

Edith Cowan University<sup>2</sup>

PO.Box 573, Sharjah, UAE<sup>1</sup>

PO.Box 127788, Abu Dhabi, UAE<sup>2</sup>

{mkhonji, youssef.iraqi, andrew.jones}@ku.ac.ae

## Abstract

*This is a study that focuses on enhancing the mitigation of bulk phishing email messages (i.e. email messages with generic socially engineered content that target a broad range of recipients). This study is based on a phishing website detection technique that we have proposed previously. The previously proposed technique was able to achieve 97% of classification accuracy of phishing websites by lexically analyzing their URLs. The centre claim of this study is that the classification accuracy of anti-phishing email filters enhance when they incorporate the proposed lexical URL analysis technique. To evaluate the claims, a highly accurate anti-phishing email classifier is constructed and tested against publicly available phishing and legitimate email data sets.*

*Index Terms* — phishing attacks; lexical analysis; URLs; emails;

## 1. Motivation

Phishing is defined as a form of crime that unlawfully, and through *Social Engineering*, obtains data from victims for the attacker's benefit, over an electronic communication channel.

User awareness training programs can aid the mitigation process of phishing attacks. However, training programs, alone, are not sufficient as knowledge retention tends to be around 16 days unless the training is made continuous [1], and non-technical end-users have cognition limitations[2]. Thus technical solutions are also beneficial to enhance the mitigation process of phishing attacks.

This study focuses on enhancing the classification accuracy of anti-phishing email classifiers. The approach that is followed is by incorporating a previously proposed website classification technique into an email classifier.

## 2. Background and Related Work

Relevant literature items to this study are techniques that deploy Machine-Learning models to classify the messages as they have shown noticeable advantage over other techniques. Other techniques, such as blacklisting and heuristic techniques, are excluded from this section as they are evaluated to suffer with zero-hour attacks rates or have high false positive rates[3].

Fette et. al. [4] presented the design and evaluation of the first Machine Learning-based email classifier to detect phishing messages, which showed promising results that achieved low false positives while avoiding using blacklists. The proposed work in [4] extracted a number of features from suspect emails, including external scores from SpamAssassin<sup>1</sup> and Whois lookups. The data set used in the evaluation process contained phishing and legitimate samples from publicly available datasets, namely [5] and [6] respectively, and by running 10-fold cross-validation they achieved an  $f_1$ -score of 97.6%.

Bergholz et. al. [7] proposed a Machine Learning classifier with model-based features — that is, features that themselves are classification models and require to be trained first prior to their use by a parent classifier. The proposed classifier used a total of 27 features, two of which were model-based features. The performance evaluation run against a data set obtained from the same sources as [4], namely: [5] and [6], and resulted in an  $f_1$ -score of 99.46%.

Another classifier was proposed by Toolan et. al. [8] that used a novel Machine Learning ensembling technique that is composed of a parent classifier (C5.0 in this case) with an ensemble of 3 learners (SVM,  $k$ -NN with  $k = 3$  and  $k = 5$ ) that are applied on the legitimate branch of the parent classifier. The primary motive behind this ensemble is to re-label false negatives to boost the true positives (or recall) rate. The proposed work achieved

<sup>1</sup><http://spamassassin.apache.org/>

an  $f_1$ -score of 99.31%, which is lower than that of [7], however it used only 5 features.

In a previous study [9], we proposed a novel URL tokenization technique to classify phishing websites. The approach achieved 97% of classification accuracy by lexically analyzing suspect URLs. One of the key objectives of the approach was achieving good classification accuracy while excluding expensive tests that analyze the HTML content of phishing websites, or performing networks tests such as DNS queries or Whois lookup over the network.

A relevant website classification technique that inspired our lexical URL analysis approach is [10], which is also an early prototype of Google's large scale phishing website classification [11]. As presented in [10], URLs were lexically analyzed to extract URL tokens with five or more characters that appeared most frequently in phishing and legitimate URLs (e.g. "signin" was seen more in phishing URLs than legitimate), and then used to construct a collection of words as binary features (e.g. each word is a feature that is 1 if found in a URL, and 0 otherwise). We continue from this point by further analyzing URL tokens to increase classification robustness.

The primary differences between our previously proposed lexical URL analysis and the proposed work in [10] is the following (details will follow in subsequent sections):

- Our lexical URL analysis approach considers the relative position of a token (e.g. *PayPal* as a main domain is different than when it exists as a subdomain), which reduces the overlap between phishing and legitimate URL tokens. This helps in raising the precision of the technique.
- Our lexical URL analysis approach considers all tokens, including indistinctive and small tokens.
- Our lexical URL analysis approach does not consider tokens as binary features. Instead, each token is assigned normalized real number that reflects its *Phishiness*.

The centre claim of this study is that our proposed lexical URL analysis technique can also enhance the classification accuracy of email classifiers. Although the lexical URL analysis, alone, evaluated to have high classification accuracy, the intention is to supplement existing classifiers as a feature. In this study, we empirically show that the proposed lexical URL analysis technique enhances the classification accuracy of anti-phishing email classifiers with all evaluated feature subsets.

### 3. Website Classification by Lexical URL Analysis

Since the centre claim of this paper is evaluating the effectiveness of our lexical URL analysis approach in enhancing email classification performance, we find it important to briefly introduce the technique. Parts of this section are taken from [9] where LUA was previously published.

In many cases, an educated end-user is able to detect phishing websites by only analyzing the URL of a suspect website. For example, the URLs `http://www.paypal.com/` and `http://www.paypal.com.hostingcompany.com/` have the brand name, *paypal*, positioned differently. This can be an obvious hint to an educated user. While following this logic sounds simple, many end-users fail to pay attention to various meta-data elements such as address bars, for the very same reason why anti-phishing toolbars are evaluated to be ineffective in regulating end-user's behavior[12].

Our proposed Lexical URL Analysis (LUA) in [9] provides a URL tokenization mechanism to export the same logic stated above to the software, so that the software is able to make better decisions than an uneducated end-user. In addition to that, LUA is able to discover relations that even an educated human would not be able to discover easily, such as: if the keyword *signin* appears in a specific location the website is legitimate, but if the keyword *sign-in* appeared in the same location it might be phishing. Discovering such un-obvious relations requires performing a lot of repetitive tedious tasks which humans do not desire to do. However since LUA performs a similar logic in software, performing the repetitive tasks is fairly possible, and can lead into finding relations that an educated human might not find.

#### 3.1. Token distribution in real-life website URLs

To proceed with the lexical analysis of token distribution in phishing and legitimate URLs, we analyzed URLs to see what percentage of tokens re-appeared in subsequent URLs. For example, the first phishing URL we analyzed contained only tokens that had never been seen before (expected, as it was the 1<sup>st</sup> URL), then subsequent URLs appeared to reuse tokens that were seen in previously-analyzed URLs.

Our URLs dataset consists of collected phishing and legitimate URLs from the period July 2010 to December 2010. Our sources for phishing URLs are Phish-Tank<sup>2</sup> and HTTP access logs gathered from Khalifa University's<sup>3</sup> HTTP proxy. Our sources for legitimate URLs are Khalifa University's HTTP access logs as well as volunteers whom selectively used our experimental

<sup>2</sup><http://www.phishtank.com>.

<sup>3</sup><http://ku.ac.ae/>

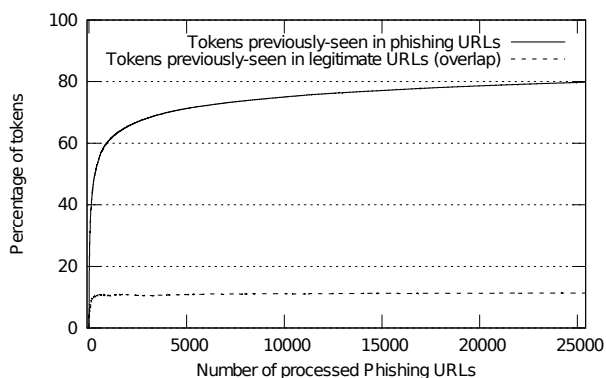


Figure 1: Percentage of previously-seen phishing URL tokens with respect to total phishing URL tokens. Source: [9]

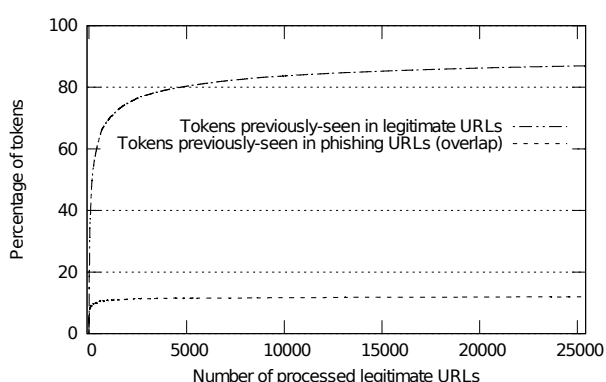


Figure 2: Percentage of previously-seen legitimate URL tokens with respect to total legitimate URL tokens. Source: [9]

HTTP proxy server. This result in 25,428 unique legitimate URLs and 48,305 unique phishing URLs — 73,733 URLs in total. The subsequent URL studies use a variant of the same dataset that maintains a 1:1 ratio between phishing and legitimate URLs.

We wrote a Perl script to automate the process described above, and the results indicated that the percentage of previously-seen tokens were much higher than our expectations. See Figures 1 and 2 for phishing and legitimate URLs respectively.

Figures 1 and 2 indicate that:

- Higher percentage of previously-seen legitimate URL tokens than phishing URL tokens suggests that the token dictionary of legitimate URLs is smaller than the token dictionary of phishing URLs. This is due to the fact that users do not browse random sites everyday, but follow certain browsing patterns and anomalies. Our survey indicates that at 20,000 processed legitimate URLs, 86.25% of the tokens were previously seen.

- Although the token dictionary of phishing URLs is larger than the token dictionary of legitimate URLs, phishing URL tokens are still surprisingly predictable as most of the tokens were also previously seen. For example, when 20,000 phishing URLs were processed, 78.62% of the tokens were previously seen.
- Since the tokens dictionaries of both phishing and legitimate URLs are — in practice — limited, it might be possible to construct a robust classification model based on URL tokens, once a sufficient volume of URLs are analyzed.
- The figures also present the percentage of overlapping tokens in relation to phishing and legitimate URLs. For example, if a legitimate URL contained token  $L_1$ , and was previously seen in a phishing URL, then we consider  $L_1$  as an overlapping token. The overlap is around 12% of total seen tokens. This means that the majority of previously seen tokens, in both phishing and legitimate URLs, are not overlapping, which is a promising point to begin with.

The existence of overlapping tokens is expected. For example brand name tokens, such as *paypal*, *ebay*... etc, are commonly seen in both phishing and legitimate URLs. However, what differentiates them is their relative position — if *paypal* is seen as a main domain (e.g. <http://www.paypal.com/>), is different than if it is seen as a subdomain (e.g. <http://www.paypal.com.phish.com/>). To evaluate the effectiveness of the inclusion of a token's relative position, we repeated our analysis above except that tokens positions were also recorded. We propose a novel approach to calculate the relative position based on token hops in relation to a reference point, which we decided to make the TLD — so positive positions indicate tokens residing in the URL's path, while negative positions indicate tokens residing as domains or subdomains. For example, as depicted in Figure 3, the URL "<http://paypal.com.phish.com/>" is lexically broken into the (*tokenValue*, *relativePosition*) pairs: *paypal*, -3, *com*, -2, and *phish*, -1. Note that the token *com*, 0 is ignored as the use of Top Level Domains (TLDs) are too common and it can cause false positives or false negatives.

In other words, a token is considered previously-seen if both elements, the token value and token's relative position, match. Our goal is to reduce the overlap percentage, while keeping the non-overlapping previously-seen token percentages close to what they are. The results are depicted in Figures 4 and 5.

As we present in Figures 4 and 5, the percentage of previously-seen tokens dropped slightly, which is expected (previously-seen tokens dropped to 71.61% in

```

http://paypal.com/
      -1  0
http://phish.com/paypal.com/
      -1  0  1  2
http://paypal.com.phish.com/
      -3 -2 -1  0

```

Figure 3: URL tokenization example followed by the proposed lexical URL analysis approach.

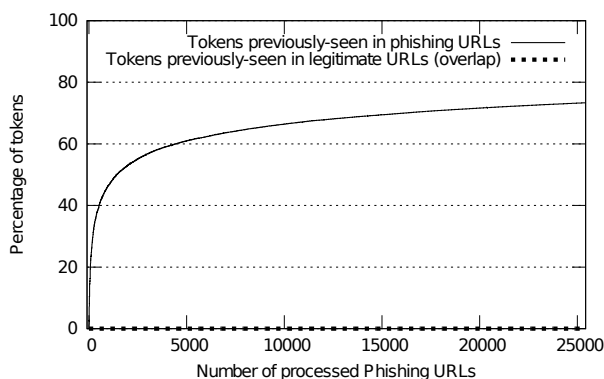


Figure 4: Percentage of previously-seen phishing URL tokens with respect to total phishing URL tokens (after the inclusion of token value's relative position). Source: [9]

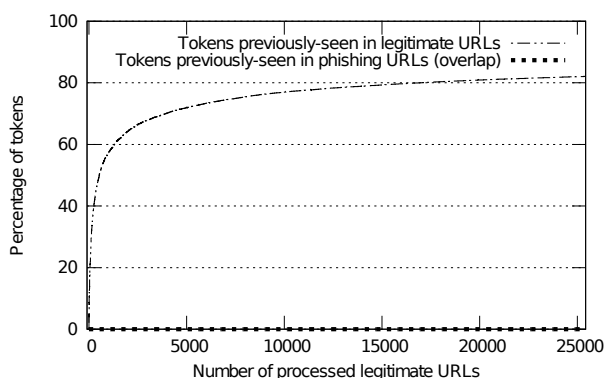


Figure 5: Percentage of previously-seen legitimate URL tokens with respect to total legitimate URL tokens (after the inclusion of token value's relative position). Source: [9]

phishing URLs, and to 80.92% in legitimate URLs at 20,000 analyzed URLs). However, the overlapping tokens dropped noticeably and reached as low as 0.005% in both phishing and legitimate URLs at 20,000 analyzed URLs, which is a significant reduction in overlapping tokens.

This indicates that our proposed lexical analysis of URL tokens provides excellent discriminative features and can be used as the basis of a robust clas-

sification model. Next, in Section 3.2, we proceed with constructing a classification model that only uses  $(tokenPosition, relativePosition)$  pairs as features. The aim is to empirically evaluate the effectiveness of our lexical URL analysis as classification features.

### 3.2. A statistical website classifier

In this section, we build a statistical classifier that takes unclassified URLs as input, and returns a predicted binary class as output (either Phish or Legit). The aim of this section is not to evaluate the classifier itself (i.e. we might get better results using other classification techniques), but rather to evaluate the effectiveness of URL tokens as discriminating features.

The classifier's life begins by a supervised learning phase. During this phase, the classifier is fed with pre-classified URLs, followed by breaking each URL into tokens. Each token is then represented as a two-dimensional vector  $V = (t, l)$ , where  $t$  is the token value and  $l$  is token's relative position in a given URL, see Figure 3 for a tokenization example. During the learning phase, the two-dimensional vectors representing URL tokens are submitted to our classifier along with their pre-defined class. The classifier is then able to infer a classification model.

Once the learning phase is complete, the classifier is given unclassified URLs as input, and a predicted class is returned as output. The classifier achieves this by breaking the suspect URL into tokens, and attempts to find out the number of occurrences of each token along with its relevant class. For example, if a token appeared 10 times in legitimate URLs, and 90 times in phishing URLs, then the *Phishiness* of the token is predicted to be 0.9, using Equation 1.

$$Pr(P|t_i) = \frac{N_{t_i \rightarrow P}}{N_{t_i \rightarrow P} + N_{t_i \rightarrow L}} \quad (1)$$

where  $Pr(P|t_i)$  is the probability estimate of a URL to be a Phish given the token  $t_i$ ,  $N_{t_i \rightarrow P}$  is the number of times token  $t_i$  appeared in phishing URLs, and  $N_{t_i \rightarrow L}$  is the number of times token  $t_i$  appeared in a legitimate URLs.

The process is repeated until the *Phishiness* is calculated for each token in a given suspect URL. Then we represent the URL's phishiness by averaging the *Phishiness* of all its individual tokens, using Equation 2.

$$Phishiness(U) = \frac{\sum_{i=1}^n Pr(P|t_i)}{n} \quad (2)$$

where  $Phishiness(U)$  is the phishiness value of URL  $U$ ,  $t_i$  are the tokens of  $U$  and hence  $n$  is the number of tokens in  $U$ . A given URL  $U$  is considered Phish if  $Phishiness(U)$  exceeds a certain threshold — we will refer to this threshold as the *Phishiness threshold*.

This technique showed high classification accuracy of phishing and legitimate URLs even when trained with few phishing and legitimate URL ratios as presented in Table 1.

#### 4. Classification of E-mail Messages and Feature Subset Selection

This section describes the implementation of a Machine Learning-based email classifier to evaluate the claim that the addition of LUA enhances the classification accuracy of email messages.

A number of features are extracted from email messages, which results into representing each message as a multi-dimensional vector where each element in the vector represents the value of a respective feature.

The LUA technique is added as a feature where it analyzes URLs found in a suspect email message, and considers the highest URL phishiness value as the phishiness value of the email message.

In [13], we found that the *Wrapper* and *Best-first: Forward* were the most effective feature subset selection methods that resulted in maximum  $f_1$  score compared to other methods. Thus, we used the *Wrapper* and *Best-first: Forward* as the feature subset selection method.

To construct the classification model, we used Random Forests (RF). This decision is due to RF's superior accuracy in our preliminary experiments compared to other algorithms (e.g. C4.5, ANN, SVM, LR), which are also confirmed by [4] and [14].

### 5. Features Set

In [15], the authors collected 40 features most of which were previously used in other literature such as [4, 7, 16]. The vast majority of the features used in our study are from [15] in addition to 2 external features from [7] and 5 additional features derived from those in [15]. This results in a total of 47 features that are extracted by analyzing data parts **C**, and **D** as depicted in Figure 6.

A common approach in extracting features found in data parts **A** and **B** is via the use of blacklists, which were intentionally avoided as blacklists perform poorly against zero-day attacks.

#### 5.1. Email Body Features

A total of 11 features were extracted from the body part of email messages that are found in **D**, namely:

**5.1.1. bodydearword.** A binary feature that returns 1 if the word “dear” is found in the body of a message, and 0 otherwise.

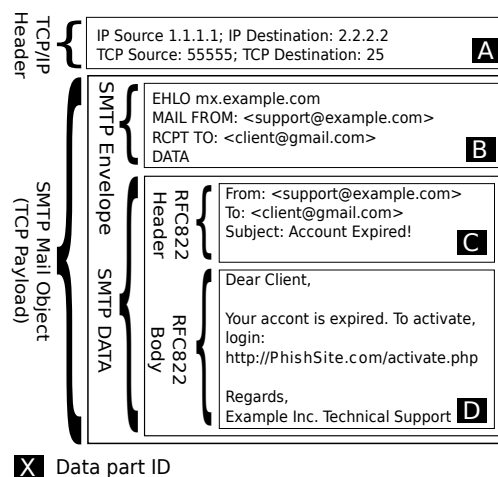


Figure 6: An overview of email data parts.

**5.1.2. bodyform.** A binary feature that returns 1 if the email message contains a HTML form, and 0 otherwise. To simplify parsing, this feature searches for opening and closing HTML form tags. This feature was previously proposed in [7].

**5.1.3. bodyhtml.** A binary feature that returns 1 if the email message has HTML content, and 0 otherwise. To simplify parsing, this feature searches for a match for “Content-type: text/html” string. This feature was previously proposed in [4, 8, 15].

**5.1.4. bodymultipart.** A binary feature that returns 1 if the email message has a multipart MIME type, and 0 otherwise. To simplify parsing, this feature searches for a match for “Content-Type: multipart/alternative”. We have derived this feature from *bodyhtml*.

**5.1.5. bodynumchars.** A feature that returns the total number of characters found in the body of a given email. This feature was previously proposed in [17].

**5.1.6. bodynumwords.** A feature that returns the total number of words found in the body of a given email. This feature was previously proposed in [17].

**5.1.7. bodynumuniqwords.** A feature that returns the total number of unique words found in the body a given email message. This feature was previously proposed in [17].

**5.1.8. bodyrichness.** A feature that returns the result of dividing the total number of words by the total number of characters found in the body of a given email. This

Table 1: Classification accuracy of LUA with various datasets. Source: [9]

Dataset ID	Phishing URL %	Legitimate URL %	Phishiness threshold	Accuracy
1	10%	90%	0.35	95.53%
3	30%	70%	0.45	96.75%
5	50%	50%	0.50	97.31%
7	70%	30%	0.54	97.12%
9	90%	10%	0.60	96.41%

feature was previously proposed in [17]. See Equation 3.

$$Richness = \frac{N_{Words}}{N_{Characters}} \quad (3)$$


where  $N_{Words}$  is the total number of words in the email's body, and  $N_{Characters}$  is the total number of characters in an email's body.

**5.1.9. bodynumfunctionwords.** A feature that returns the total number of function words found in the body of a given email. Function words are: “account”, “access”, “bank”, “credit”, “click”, “identity”, “inconvenience”, “information”, “limited”, “log”, “minutes”, “password”, “recently”, “risk”, “social”, “security”, “service”, and “suspended”. This feature was previously proposed in [17].

**5.1.10. bodysuspensionword.** A binary feature that returns 1 if the word “suspension” is found in the body of an email, and 0 otherwise[18].

**5.1.11. bodyverifyyouraccountphrase.** A binary feature that returns 1 if the phrase “verify your account” is found in the body of an email, and 0 otherwise.

## 5.2. Email Header Features

A total of 11 features were extracted from RFC822 email header fields that are found in  (e.g. sender, reply-to, subject), namely:

**5.2.1. subjectbankword.** A binary feature that returns 1 if the “bank” word is found in the subject field of a given email message, and 0 otherwise[18].

**5.2.2. subjectdebitword.** A binary feature that returns 1 if the word “debit” is found in the subject field of a given email message, and 0 otherwise[18].

**5.2.3. subjectfwdword.** A binary feature that returns 1 if the word “Fwd: ” is found in the subject field of a given email message, and 0 otherwise. This feature was previously proposed in [15].

**5.2.4. subjectreplyword.** A binary feature that returns 1 if the word “Re: ” is found in the subject field of a given email message, and 0 otherwise. This feature was previously proposed in [15].

**5.2.5. subjectverifyword.** A binary feature that returns 1 if the word “verify” is found in the subject field of a given email message, and 0 otherwise[18].

**5.2.6. subjectnumchars.** A feature that returns the total number of characters found in the subject field of a given email. This feature was previously proposed in [15].

**5.2.7. subjectnumwords.** A feature that returns the total number of words found in the subject field of a given email. This feature was previously proposed in [15].

**5.2.8. subjectrichness.** A feature that returns the result of dividing the total number of words by the total number of characters found in the subject field of a given email. This feature was previously proposed in [15]. See Equation 3, where  $N_{Words}$  is the total number of words in the email's subject, and  $N_{Characters}$  is the total number of characters in an email's subject.

**5.2.9. sendnumwords.** A feature that returns the total number of words found in the “sender” field of a given email. This feature was previously proposed in [15].

**5.2.10. senddiffreplyto.** A binary feature that returns 1 in the case where a difference between the sender and reply-to email addresses is found, and 0 otherwise. This feature was previously proposed in [15].

**5.2.11. sendunmodaldomain.** A binary feature that returns 1 in the case where the sender email address uses an unmodal domain name, and 0 otherwise. A modal domain name is defined as the most frequently referred to domain name in the body of a given email. This feature was previously proposed in [15].

### 5.3. URL Features

A total of 18 features were extracted from URLs and anchors that are found in **D**, namely:

**5.3.1. urlatchar.** A binary feature that returns 1 in the case where a given email message contained a URL with an “@” sign, and 0 otherwise. This feature was previously proposed in [16].

**5.3.2. urlbaglink.** A binary feature that returns 1 in the case where any of the following keywords are found, and 0 otherwise: “click”, “here”, “login”, and “update”. This feature was previously proposed in [7].

**5.3.3. urlip.** A binary feature that returns 1 in the case where a given email message contained a URL with an IP address in its *authority* portion, and 0 otherwise. This feature was previously proposed in [4, 7, 8].

**5.3.4. urlnumdomains.** A feature that returns the total number of domains found in URLs in a given email. This feature was previously proposed in [4].

**5.3.5. urlnumexternallink.** A feature that returns the total number of external links found in a given email. An external link is a link that points to a resource that is accessible out of the email (e.g. a website). This feature was previously proposed in [7].

**5.3.6. urlnuminternallink.** A feature that returns the total number of internal links found in a given email. An internal link is a link that points to a resource that is accessible in the email (e.g. anchors within the email — fragments). This feature was previously proposed in [7].

**5.3.7. urlnumimagelink.** A feature that returns the total number of image links found in a given email. This feature was previously proposed in [7].

**5.3.8. urlnumip.** A feature that returns the total number of URLs that contain an IP address in their authority section as opposed to a domain name. This feature was previously proposed in [7].

**5.3.9. urlnumlink.** A feature that returns the total number of links found in the body of a given email. This feature was previously proposed in [4, 7, 8].

**5.3.10. urlnumperiods.** A feature that returns the total number of periods in the body of a given email. This feature was previously proposed in [4, 7, 8].

**5.3.11. urlnumport.** A feature that returns the total number of URLs with port numbers in their authority section in a given email. This feature was previously proposed in [15].

**5.3.12. urlport.** A binary feature that returns 1 if a URL with a port number is found in the body of a given email, and 0 otherwise[18].

**5.3.13. urltwodomains.** A binary feature that returns 1 if a URL that has two domain names is found, and 0 otherwise. Technically, URLs can have only a single domain name. However, the second domain name is heuristically detected. For example, `http://paypal.com.phish.com` is technically considered to contain a single domain name, which is “paypal.com.phish.com”, while this feature will treat “paypal.com” and “phish.com” as two different domains. The feature uses a heuristic test that considers any string of characters, including the hyphen, that precedes a TLD as a domain name. This feature is a simplified version of the one proposed in [19].

**5.3.14. urlunmodalbaglink.** A binary feature that returns 1 if an unmodal link is found with certain words in its link text, and 0 otherwise. The particular words are: “click”, “here”, and “link”. A modal domain is the most frequently linked domain name in a given email. This feature was previously proposed in [7].

**5.3.15. urlwordclicklink.** A binary feature that returns 1 if a link with “click” word in its link text is found, and 0 otherwise.

**5.3.16. urlwordherelink.** A binary feature that returns 1 if a link with “here” word in its link text is found, and 0 otherwise.

**5.3.17. urlwordloginlink.** A binary feature that returns 1 if a link with “login” word in its link text is found, and 0 otherwise.

**5.3.18. urlwordupdatelink.** A binary feature that returns 1 if a link with “update” word in its link text is found, and 0 otherwise.

### 5.4. JavaScript Features

A total of 5 JavaScript features were extracted from email bodies that are found in **D**, namely:

**5.4.1. scriptjavascript.** A binary feature that returns 1 if the body of a given email message contains JavaScript, and 0 otherwise. To simplify parsing, the feature looks for a match for “javascript” string. This feature was previously proposed in [4, 7].

**5.4.2. scriptonclick.** A binary feature that returns 1 if an “onClick” JavaScript event was found in the body of a given email, and 0 otherwise.

**5.4.3. scriptpopup.** A binary feature that returns 1 if a given email message contains JavaScript code to open pop-up windows, and 0 otherwise.

**5.4.4. scriptstatuschange.** A binary feature that returns 1 if a given email message contains JavaScript code to modify the status bar, and 0 otherwise. This feature was previously proposed in [16].

**5.4.5. scriptunmodalload.** A binary feature that returns 1 if a given email message contains JavaScript that is loaded from an external website which is not a modal domain name, and 0 otherwise. A modal domain name is defined as the most frequently linked domain name.

## 5.5. External Features

Similar to [7], SpamAssassin<sup>4</sup> (SA) was used to construct two external features, which were both evaluated by running *spamc* against emails with all network tests being disabled at the *spamd* process.

**5.5.1. externalsabinary.** A binary feature that returns 1 if a given email is labeled as a spam message by SA, and 0 if otherwise (i.e. a binary feature).

**5.5.2. externalsascore.** A feature that returns the score of a given email as returned by SA (i.e. a contentious feature).

## 6. Performance Evaluation

### 6.1. Datasets

Publicly available sources of phishing and legitimate emails were used, namely: [5] and [6] respectively. The same email sources were also used in a number of recent studies [7, 14, 15].

Our phishing dataset is composed of 4,116 emails from the following files as distributed by [5]: phishing0.mbox, phishing2.mbox, and

<sup>4</sup>The apache SpamAssassin project <http://spamassassin.apache.org/>.

Table 2: Classification confusion matrix

	Labeled as Phish	Labeled as Legit
Is Phish	$N_{P \rightarrow P}$	$N_{P \rightarrow L}$
Is Legit	$N_{L \rightarrow P}$	$N_{L \rightarrow L}$

phishing3.mbox. The legitimate dataset is composed of 4,150 emails from the following files as distributed by [6]: 20030228\_easy\_ham.tar.bz2, 20030228\_hard\_ham.tar.bz2, 20030228\_easy\_ham\_2.tar.bz2. This assembles a dataset of phishing and legitimate emails similar to “Dataset 2” in [15].

### 6.2. Evaluation Metrics

In any binary classification problem, only four classification possibilities exist. See the confusion matrix presented in Table 2 for details, where  $N_{P \rightarrow P}$  is number of *phishing* instances that are correctly classified as *phishing*,  $N_{L \rightarrow P}$  is number of *legitimate* instances that are incorrectly classified as *phishing*,  $N_{P \rightarrow L}$  is number of *phishing* instances that are incorrectly classified as *legitimate*, and  $N_{L \rightarrow L}$  is number of *legitimate* instances that are correctly classified as *legitimate*.

Based on our review of the literature, the following are the most commonly used evaluation metrics:

- False Positives Rate (*fpr*) — measures the rate of legitimate instances that are incorrectly detected as phishing attacks in relation to all existing legitimate instances. See (4) for details.
- Precision (*p*) — measures the rate of correctly detected phishing attacks in relation to all instances that were detected as phishing. See (5) for details.
- Recall (*r*) — measures the rate of correctly detected phishing attacks in relation to all existing phishing attacks. See (6) for details.
- $f_1$  score — is the harmonic mean of P and R. See (7) for details.

$$fpr = \frac{N_{L \rightarrow P}}{N_{L \rightarrow L} + N_{L \rightarrow P}} \quad (4)$$

$$p = \frac{N_{P \rightarrow P}}{N_{P \rightarrow P} + N_{L \rightarrow P}} \quad (5)$$

$$r = \frac{N_{P \rightarrow P}}{N_{P \rightarrow P} + N_{P \rightarrow L}} \quad (6)$$

$$f_1 = \frac{2p \cdot r}{p + r} \quad (7)$$



### 6.3. Evaluation Approach

The following primary feature sets are constructed, namely:

- Features set 1: contains 47 feature as described in Section 5.
- Features set 2: incorporates our proposed LUA as a feature, which results in a total of 48 features.

The Wrapper subset evaluation criterion and Best-first: Forward subset searching method are run against both of the feature sets. This process results into the following effective feature subsets:

- Features subset **1-A**: a chosen features subset by Wrapper and Best-first: Forward, that includes LUA feature.
- Features subset **1-B**: identical to the features subset 1-A except that LUA is manually removed.
- Features subset **2-A**: a chosen features subset by Wrapper and Best-first: Forward, that excludes LUA feature.
- Features subset **2-B**: identical to the features subset 2-A except that LUA is manually added.
- Features subset **3-A**: the full features set without features subset selection, with LUA (48 features in total).
- Features subset **3-B**: the full features set without features subset selection, without LUA (47 features in total).

The motive behind running feature subset selection methods on the features sets (as opposed to using all of them) is that unnecessary features can increase the time and space complexity of the classifiers, as well as the possibility of degrading classifier's accuracy [20].

To evaluate the feature subsets above, we constructed 6 classification models through a training phase using the data set described in Section 6.1, with each of the models representing one of the six features subsets above.

The classification models are then evaluated using *10-fold cross-validation* [20], which is the same evaluation approach followed in the literature such as [4, 7].

Each round of the 10-fold cross-validation assigns 90% of the data set as training, and 10% as testing. However, since LUA is a model-based feature, it requires to be trained prior its use in the training phase of the parent classifier (i.e. RF), and thus part of the 90% needs to be assigned to LUA's training phase. In other words, each round in the 10-fold cross-validation is organized as follows:

Table 3: Performance evaluation results of RF classification models with varying feature sets.

Features	$fpr$	$p$	$r$	$f1$
<b>1-A</b>	<b>0.59%</b>	<b>99.41%</b>	<b>99.37%</b>	<b>99.37%</b>
1-B	1.02%	98.98%	99.12%	99.12%
2-A	0.72%	99.27%	98.81%	98.81%
<b>2-B</b>	<b>0.72%</b>	<b>99.27%</b>	<b>99.14%</b>	<b>99.14%</b>
<b>3-A</b>	<b>0.52%</b>	<b>99.47%</b>	<b>99.31%</b>	<b>99.31%</b>
3-B	0.60%	99.40%	99.17%	99.17%

- 30% of the data set is used as a training phase for the LUA technique. Our choice of 30% is due to our preliminary studies that showed good classification results even when trained with only 30% training samples. Once this training phase is complete, the constructed LUA model adds email phishiness values into the remaining 70% of the data set in the same round of 10-fold cross-validation.
- 60% of the data set is used as a training set for the parent classifier (i.e RF).
- 10% of the data set is used as a testing set for the parent classifier.

10-fold cross-validation tests are repeated for multiple times, each with the initial data set randomly shuffled using varying random seeds, and the evaluation metric rates are then averaged.

### 6.4. Evaluation Results

Performance evaluation results, as described in Section 6.3 are presented in Table 3.

As presented in Table 3, all feature sets that include the LUA feature have an increased detection of phishing emails, while reducing their false positives rate.

Features set 1-B has an  $fpr$  of 1.02%, which we believe is too high for most production environments. However, the addition of the LUA feature reduced it down to 0.59%.

The only exception is features subset **2-B** where no  $fpr$  improvement is made compared to that of features set **2-A**. We believe this is due to the fact that the Wrapper and Best-first: Forward was run to select feature sets specifically to exclude the LUA feature. However, the addition of LUA still showed noticeable improvements in rates  $r$  and  $f1$ .

Interestingly, when RF was run with AdaBoostM1 [20] and using features set **3-A** (all of the 48 features including the LUA), its classification model resulted in an  $f1$  score of 99.45%. Only one classifier is known to have a higher  $f1$  score of 99.46%

[7], however it uses additional model-based features and image processing techniques.

## 7. Conclusion

This study presents a modified variant of a website classification technique that we have previously proposed that proved to be effective in enhancing the classification accuracy of anti-phishing email filters as well. As evaluated empirically on publicly available phishing and legitimate email data sets, the addition of the LUA technique proved to be effective in enhancing the classification performance of the evaluated email classifier with virtually all evaluated features subsets.

We believe that the positive results are due to the fact that most phishing email messages contain URLs, and enhancing website detection techniques can directly benefit the performance of anti-phishing email classifiers. One of the advantages of the implemented LUA technique is its ability to accurately classify phishing websites by only analyzing their URLs lexically. Other website classification techniques also process website contents (e.g. HTML) or send queries over the network.

## Acknowledgment

The authors would like to thank Buhooth<sup>5</sup> for funding this work.

## References

- [1] A. Alnajim and M. Munro. An evaluation of users' anti-phishing knowledge retention. In *Information Management and Engineering, 2009. ICIME '09. International Conference on*, pages 210–214, April 2009.
- [2] Gerry Gaffney. The myth of the stupid user. <http://www.infodesign.com.au/articles/themythofthestupiduser>. Accessed March 2011.
- [3] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. <http://ceas.cc/2009/papers/ceas2009-paper-32.pdf>, 2009.
- [4] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web, WWW '07*, pages 649–656, New York, NY, USA, 2007. ACM.
- [5] Jose Nazario. Phishing corpus. <http://monkey.org/~jose/wiki/doku.php?id=phishingcorpus>. Accessed July 2010.
- [6] SpamAssassin. Public corpus. <http://spamassassin.apache.org/publiccorpus/>. Accessed January 2011.
- [7] André Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paaß, and Siehyun Strobel. New filtering approaches for phishing email. *J. Comput. Secur.*, 18:7–35, January 2010.
- [8] F. Toolan and J. Carthy. Phishing detection using classifier ensembles. In *eCrime Researchers Summit, 2009. eCRIME '09.*, 20 2009.
- [9] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Lexical url analysis for discriminating phishing and legitimate websites. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS '11*, pages 109–115, New York, NY, USA, 2011. ACM.
- [10] Sujata Garera, Niels Provos, Monica Chew, and Aviel D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malware, WORM '07*, pages 1–8, New York, NY, USA, 2007. ACM.
- [11] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. <http://research.google.com/pubs/pub35580.html>. Accessed July 2010.
- [12] Lorrie Cranor, Serge Egelman, Jason Hong, and Yue Zhang. Phishing phish: An evaluation of anti-phishing toolbars. [www.cylab.cmu.edu/files/cmucylab06018.pdf](http://www.cylab.cmu.edu/files/cmucylab06018.pdf), 2006.
- [13] Mahmoud Khonji, Andrew Jones, and Youssef Iraqi. A study of feature subset evaluators and feature subset searching methods for phishing classification. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS '11*, pages 135–144, New York, NY, USA, 2011. ACM.
- [14] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit, eCrime '07*, pages 60–69, New York, NY, USA, 2007. ACM.
- [15] F. Toolan and J. Carthy. Feature selection for spam and phishing detection. In *eCrime Researchers Summit (eCrime), 2010, eCrime '10*, Dallas, TX, 2010.
- [16] Wilfried N. Gansterer and David Pölz. E-mail classification for phishing defense. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval, ECIR '09*, pages 449–460, Berlin, Heidelberg, 2009. Springer-Verlag.
- [17] M. Chandrasekaran, K. Narayanan, and S. Upadhyaya. Phishing email detection based on structural properties. In *NYS Cyber Security Conference*, 2006.
- [18] SonicWall. Bayesian spam classification applied to phishing e-mail. In *Whitepaper*, 2008.
- [19] Mahmoud Khonji, Andrew Jones, and Youssef Iraqi. A novel phishing classification based on url features. In *GCC Conference and Exhibition (GCC), 2011 IEEE*, 2011.
- [20] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

<sup>5</sup><http://www.buhooth.ae/>