APPROACHES TO PHISHING DETECTION USING LINK CLASSIFICATION

Brian Curtis and Jan Seruga
Australian Catholic University - Sydney, Australia

ABSTRACT

Phishing is a growing problem with serious financial and security consequences for those entrapped by it. This paper examines the evolution, over the past seven years, of schemes for classifying Links and URLs as Phishing related. The initial techniques relied upon the use of expertly hand-selected Link/URL features, plausible, but ad-hoc, heuristics, a reliance on the context of the embedding email or web page and often the use of black and white lists. Over time many of these techniques have lost much of their potency due to the increasing sophistication of the Phishers' methods. Consequently, more recent work places a greater reliance upon automating feature selection using text classification techniques. Our work involves building a URL classifier and using to test the various schemes and general feature classes to determine their efficacy and possibility for improvement. We make use of publicly available lists of black and white URLs for purposes of both training and testing the classifier. Our results indicate a simple approach is superior to an over-complicated separation of feature sets.

KEYWORDS

Phishing, URL classification, blacklists, whitelists, Naïve Bayes classification

1. INTRODUCTION

Phishing scams are a growing problem, affecting all users on the Internet, that trap the unwary into betraying private information including identity and passwords to important web sites. While education must play a major role in combating such scams, other techniques involving automatic filtering of emails and warnings of the presence of suspicious links must also be employed. In section 2 we consider the development of automatic techniques used for the classification of links as phishing-related. In section 3, we employ a modular URL classifier, to experimentally investigate the efficacy of various schemes for feature sets selection.

2. THE DEVELOPMENT OF LINK & URL CLASSIFIERS

Current anti-phishing link/URL classification tools can be categorized as either list-based (black and white) or rule-based. Our particular interest is in online/automatic classification but we first consider the role and effectiveness of blacklists in section 2.1.

Section 2.2 considers systems that make significant use of the context (email/webpage) of a link in order to perform classification. These systems tend to depend on expertly informed presumptions about the factors that indicate suspicious links or emails.

Section 2.3 discusses machine-learning techniques, in particular those of Ma et al, which operate largely independently of link context. Rather than make many assumptions about the nature of suspicious links they generally leave such feature selection up to an automated learning process. It is this method that we further investigate in section 3.

2.1 Blacklists

A number of services maintain blacklists of "known" phishing website URLs. PhishTank [1] provides a volunteer based service to manually classify submitted sites to construct a freely available database. The Anti Phishing Working Group [2] is an industry organization that only provides data to (generally) fee-paying members. For blacklist techniques to be successful, a list must be updated in near real time with newfound black sites. Thus there will always be a problem with newly emerged, but as yet unclassified, black sites. Additionally, the delay in the distribution of database updates further adds to this window of vulnerability. For example, the current database of PhishTank, containing around 10,000 to 15,000 currently active black sites, may be downloaded as a ½ megabyte compressed CSV file. Although not very large, it is still something that one would not wish to download at high frequency (perhaps only once or twice an hour). Alternatively, an API request may be made to classify a particular URL, although, again, there is an hourly usage limit.

As phishing emails are an instance of spam, the application of such blacklists may be performed by spam filtering services rather than relaying the email to the client's computer. This of course accentuates the perennial problem of false positives as the client looses any override ability.

2.2 Context Based Classifiers

Tools that reside in the users email client can perform checks on their emails to detect questionable hyperlinks.

A number of methods have been developed to classify such links based on characteristics of the links URL, the links context within an email (or webpage) and an analysis of the content of the target web page.

2.2.1 LinkGuard

Chen and Guo [3] describe "LinkGuard", a client-side algorithm for the detection of phishing attacks that operates by analyzing the hyperlinks present in phishing emails. Instead of simply consulting a blacklist, the hyperlinks are examined by their domain name characteristics.

In particular, an HTML hyperlink of the form: *Anchor* is examined and classified as follows (occurrence percentages as per their paper):

Class:

1) The Anchor is itself of a URL form but does not correspond to that of the link (44%)

2) The URL is an IP address rather than a domain (42%)

3) The URL is formed using ASCII alphabet encoding or special characters (e.g. @) are used in the Anchor to obscure a malicious link (17%)

4) The Anchor is neutral (e.g. "Click") but the URL is very similar to some known site. (33%)

5) A vulnerability of a genuine target site (URL) is exploited; e.g. for an injection attack. (2%)

The authors give the example:

*

Interestingly, the proportion of class 2 phishing links (IP addresses) plummeted from 42% quoted at the time of their paper (2006) to under 2% reported by the APWG in its 4th quarter 2012 trend report [4], although this has increased to slightly over 5% in its 2nd quarter 2013 report[5].

The authors note that there is some overlap between these classes; i.e. they do not partition the space of phishing hyperlinks. This is evidenced by the sum of their occurrences exceeding 120%. Also, class 5 links (vulnerability exploits) constitute only 2% and not further considered. (Note there are many mentions of category 5 in their paper where the authors probably mean category 4 or simply a potentially clean site).

The general outline of their algorithm is an ordered check:

- class 1 links indicate phishing
- class 2 links indicate possible phishing
- class 3 links are decoded and analyzed recursively.

- remaining links are compared to blacklists (phishing), whitelists (not phishing), the email sender dns (possible phishing) and if absent are subjected to pattern analysis to determine if they fit into class 4, indicating possible phishing.

The pattern analysis is designed to find URLs that are suspiciously similar to known domain names. This is determined by some, vaguely specified, string difference metric (possibly Levenshtein / edit distance), which, if small but non-zero, indicates possible phishing. The known domains are derived over time from "proactively collecting" domain names that are manually entered by the user into their web-browser, and hence may be assumed to be trustworthy.

The authors argue that although LinkGuard may return false positives it is "very unlikely to cause [sic] harmful false negatives". The basis of this claim is that if a domain name is not closely similar to one of the user's "know" domains, then the user is not likely to have important information on the site. However, a potential flaw in this reasoning is that it could be also be argued that the user has been lured to such a new site with the intention of acquiring their identity and/or credit card information (e.g. by a deal too good to miss).

A second potential flaw in the design is that, as it stands, it can only work where the user restricts themselves to one computer system. If the user gets a phishing email on say their mobile device then it is quite possible that, while their browsing history is quite different from their home/office/laptop (where they do internet banking), they feel compelled to immediately go to the site to correct a fraudulently reported vulnerability. Potentially this could be rectified by cloud sharing of such browsing history (although that can introduce its own problems)

2.2.2 PILFER

Fette, Sadah and Tomasic [6] describe a machine learning approach to filtering phishing attacks called PILFER. They combine information internal to an email with that in the external world to create a "feature vector" which contributes to training a model. This model is used to determine the likelihood of an email being spam.

They note that browser toolbars are unable to leverage the contextual information present in a phishing email. This includes both the "words used to entice the user" and the detailed header information relevant to its provenance. Conversely, the email client does not have access to the web page information present in the browser. This suggests that a greater integration of both email and web clients would be profitable to phishing detection. This could of course be augmented by the use of user browsing history in the manner of LinkGuard.

They use an email URL feature-based classification, in some ways similar to LinkGuard. In particular, their Nonmatching and IP-based URLs are respectively equivalent to LinkGuard's Class 1 and 2. They also use a number of additional heuristics to determine the likelihood of spam:

- It is in HTML; an (almost) necessary rather than sufficient condition.
- The presence of "Fresh" domains; registered within the last 60 days
- The number of links in the email
- The number of distinct "main" domains in the email
- The number of dots in each URL (including the path and query part)
- Whether neutral links have non-modal domains (modal domains are those that appear with the highest frequency in the email)
- The email contains Javascript
- The email has been previously flagged as spam.

2.2.3 Examples

Zhang et al [7] discusses a number of example URLs that highlight various method of obfuscating a phishing domain:

- http://item.taobao.com-eis.tk/member/login.asp
  The domain appears to be "taobao.com" but is in fact "com-eis.tk"
- http://account.member.paypal.adabim.net/paypal/webscr/load.php
  This seems to indicate "member.paypal" rather than "adabim.net"
- http://hitechsense.com/images/www.chase.com/update.php

This is an instance of using a hacked webserver “hitechsense.com” to host a phishing site apparently targeting “chase.com”.

2.3 Machine Learning Classifiers

Ma et al [9] apply statistical methods from machine learning to perform classification of URLs based on a combination of their lexical and host-based features. By training their system with a “large” dataset (20-30,000 malicious URL’s) they claim that their method automatically selects many of the features that are identified by “[problem] domain experts”, and additionally extracts new and non-obvious features that are highly diagnostic of malicious web sites. In particular, their work seeks to determine the costs and benefits of utilizing a large number of automatically generated features rather than a (necessarily) small number of manually chosen features.

They specifically ignore both the URL’s referenced-page content and it context (e.g. its embedding email). They justify this in an on-line context by noting that:
- “avoiding downloading page content is safer for users” .
- “downloading the page and then using its contents for classification” is a relatively heavyweight operation.
- Obtaining the malicious version of a page or email presents many practical issues.
- Simply “focusing on URL features makes a classifier applicable to any context … Web pages, email”.

Nevertheless, we note that their context-free classification may still be augmented by contextual heuristics such as mismatched URL and anchor (LinkGuard class 1) and similarity to other slightly different but know URLs (LinkGuard class 4).

For lexical analysis they divide the URL into a path and a hostname (the protocol is dispensed with as they are basically only interested in http/https) and calculate some simple metrics (length, number of dots). They then form a token set by splitting the hostname by dots and the path by special characters “/?.=-_”. [Their later 2009 paper apparently splits the entire URL by special characters]. The resulting tokens are distinguished by their membership of hostname, path, top-level-domain, primary-domain (that given to the registrar) and last token of the path (file extension). The token order is ignored, a technique known as bag-of-words. The authors claim that this representation is able to achieve sufficiently accurate results to render more complex (e.g. Markov) models superfluous.

The second mode of analysis utilizes features of the hostname as provided by lookups of the DNS and, optionally, “whois” databases. The specific properties are:
- Blacklist membership: of the full hostname or IP
- IP address: Are those of the A, MX, NS records in the same Autonomous System (AS)? Do they share a common prefix with a disreputable ISP ?
- WHOIS: dates of registration, expiration, update; registrar; registrant; In particular, a set of black domains registered by the same individual would flag that registrant as a malicious feature.
- Domain name: Time-to-live, PTR record for the host, commonality of ISP (& others)
- Others: popularity rank; Google indexing, geographic properties, uplink speed (indicating hosting by compromised residential machine) etc

As a binary feature is assigned for every lexical token encountered as well as every host-based property. This leads to extremely large, but sparse, feature vectors that inevitably grow over time (typically of the order of 1,000,000 elements per vector).

The paper discusses a variety of feature classifiers including Naïve Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR). These classifiers are then evaluated for their effectiveness in identifying “malicious” URLs and with specific regards to answering the following questions:
- Does using more features lead to more accurate classification?
- What is the most appropriate classification model to use?
- How is accuracy affected if the classifier is tuned to reduce false positives?
- Can a model trained on one source accurately classify data from another source?
- What trends can be found in the relevant features?
- What do the misclassified examples have in common?

They conclude that, with appropriate classifiers, it is indeed feasible to automatically sift through exhaustive feature sets and extract those that are most predictive for URL classification. They tentatively chose LR as a basis for further research.

Trials with known malign and benign URLs lead them to some observations of incorrect classification.

In particular, false-positives were almost entirely benign sites hosted at disreputable ISPs (or sites hosted in the same AS or IP prefix as malicious sites); as they point out "guilt by association". They suggest that the maintenance of an ISP reputation register - to be consulted by legitimate site administrators - would be appropriate and encourage ISPs to more diligently monitor and quickly crack down on their hosting of malicious sites.

The false negatives fell into two groups based on either lexical or host-based features. The lexical feature of concern was the occurrence of benign tokens in the URL. They suggest these be subject to careful vetting to prevent them becoming [positive] features, but do not suggest how. The following problematic host based features were noted:

- malicious sites using free hosting services
- the use of compromised sites
- redirection services
- sites hosted in reputable geographic areas
- sites with TLDs hosted in the US

Understandably, they make few substantive recommendations as to how these problems may be ameliorated (though they do suggest analysis of the site content). A follow up paper [10] provides further results with an analysis of feature sets showing the types split 62% for lexical versus 38% for host-based.

In [10, 11] Ma et al extended the above work with a switch from "batch" to "on-line" learning. In batch learning, a large training set of URLs, each flagged as malign or benign, is analyzed with URL lexical and host-based features extracted and weighted. In on-line learning, the training examples are provided to the analyzer incrementally, with the prior feature sets being extended with new features or existing features' weights updated. This adapts the resulting classification to changes in the nature of malicious URLs over extended time frames.

Our analysis of this blacklist indicates at least a 50% turnover of entries per month, with approximately 10,000 to 15,000 entries at any one time. If performed monthly (rather than the recommended daily), an online analysis will update the existing feature sets with around 5000 new entries while preserving those features and weights already calculated for the last few years. A batch method would not be able to take the blacklist directly as it only represents the current state of play. Instead it would need to maintain an ever-growing list (5000 per month), which would consist mostly of historical data. It would then need to run the entire list to form a new analysis.

2.3.1 False Positives: (Legitimate Sites Classified as Malicious)

They observe that almost all the false positives were as a result of legitimate sites hosting by disreputable ISPs, or sharing the same AS or IP prefix as malicious sites.

This is a situation that occurs regularly with spam blacklists where an ISP's mail-server hosts a client that either directly sends out large spam mail-outs or the account is hacked and misused for such a mail-out. Often the mail server is placed on one or more blacklists and all the legitimate clients suffer until the ISP removes the offending client. The server's reputation can take some days to recover after its removal from the blacklists, frequently causing serious problems for the affected clients. It is privately reported [12] that at any given time at least one of Telstra's servers is blacklisted.

It is not apparent how this problem can be rectified but we note that is compounded when, rather than a simple blacklist, an "on-line learning" algorithm is used. In particular it is unclear how a host can be removed from the malicious feature set in a timely manner when that host's problem(s) have been rectified.

2.3.2 False Negatives: (Malicious Sites Classified as Legitimate)

The lexical feature that caused most false negatives was the "occurrence of benign tokens in the URL". Their suggestion that such tokens should be "subject to careful vetting" leaves open the question as to how. We suggest that the "bag of words" method, while relatively simple, does lend itself to a potential exploit based on this flaw. For instance the domains "X.com.au" and "X.au.com" are split into tokens by "."s and classified identically. Perhaps some modification to this method could remove the sensitivity of the classifier to these similarities, perhaps by including compound strings in the feature sets.

Their list of false negatives caused by host-based features is fairly predictable. The use of free services, compromised sites and redirection services are intrinsically difficult to deal with automatically without

performing a site content analysis, which certainly lies outside the scope of straightforward URL classification. Also, they specifically do not wish to do this in the client's browser.

3. FACTORS AFFECTING URL FEATURE SELECTION

3.1 Methodology

In order to compare methods of URL feature selection we have constructed a URL classifier, based primarily on Ma's work. Our focus is on the investigation of lexical rather than host-based feature selection so that element of Ma's system is not directly explored. Further, our results are determined from batch training, so we are not directly comparing batch to on-line methods.

We use a Naïve Bayes classification technique as it is commonly found effective in spam filters and, we suggest, that as long as we maintain a consistent technique we are able to adequately compare the variation of feature selection methods.

For reasons of availability, the system is trained and tested using the PhishTank black lists and various sub-sets of the extensive Open Directory Project's [13] (ODP) white lists. In order to perform a comparative analysis, we basically wish to determine rates of false positive and false negative classifications. While our data sets are limited to those we would normally fully utilize for classifier training, the fact that our interest is in the comparison of techniques, rather than achieving maximum accuracy in a real world role, means that we can sequester some of this data as test samples.

The specific feature groups used for both training and classification are coded in a modular manner and are easily variable in order to determine an optimal set. A classifier must be trained using the same method of feature selection as is latter applied to the classification of sample URLs. Therefore, for each feature grouping method we wish to explore, we must perform a batch training of the classifier and then trial a large set of URL's to determine its effect. These grouping methods may then be compared quantitatively.

3.2 Data Set Selection for Training and Testing

Our training and testing data comes from two sources. By amalgamating blacklists from PhishTank over successive months we have assembled a list of 37,045 unique black URLs. By extracting URLs from ODP's growing directory listings we have gathered 3,600,030 unique white URLs. (In subsequent work we will continue to add further data to the PhishTank amalgamation and use the latest ODP data).

For any run of the classifier, data selection is performed by traversing the entire source data set (Black or White) and randomly assigning its URLs to one of a Training set, a Testing set or a Discard set. We choose a desired size for the Training and Testing sets, which then determines the weights for the assignment. As a result we never get exactly the desired number but rather a probabilistic approximation. However, this method guarantees that the Training and Testing sets are disjoint and that the entire Source set has been sampled.

Furthermore we explicitly specify a seed to the random number generator for each run so, assuming that the Source sets remain static, any run can be replicated. In future work we will perform multiple runs and average their results; the data selection will for each such run would be made by varying this seed.

3.2.1 Sensitivity to the Relative Size of the Training Sets

Our first analysis seeks to determine the sensitivity of URL classification to the relative sample sizes used for training. Since the available White set is two orders of magnitude larger than the Black (and will likely remain so over time) we hold the Black training set size static at 30,000 (with a 7,000 element testing set, thus utilizing almost the entire set) while varying the White training set size from half that of the black set up to an order of magnitude larger.

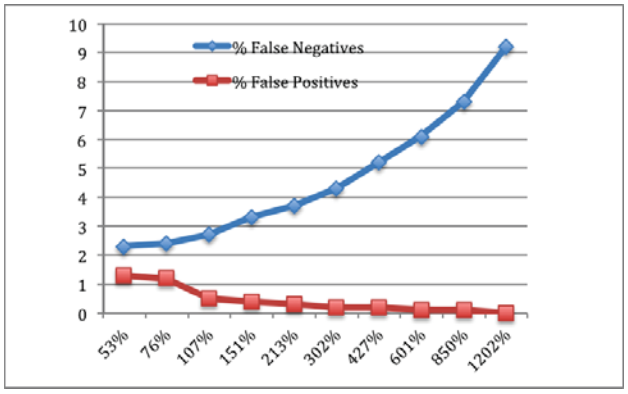

Figure 1. Failure rate vs ratio of White to Black training set size: tested using training set

As a check we first performed a set of tests using, not the prepared testing data, but the entire training data set itself. This is presented in Fig. 1. We would expect it to show "unrealistically good" [14, p9] results since the classifier has already been trained with the data it is testing. As expected, the graph shows a generally low rate of both false positives and false negatives. It also shows, roughly, a linear growth of false negatives and a linear decay of false positives as the White trainings set size increases exponentially.

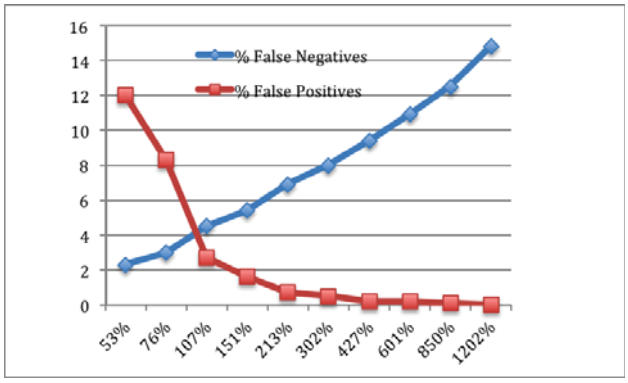

Figure 2. Failure rate vs ratio of White to Black training set size: tested with prepared test set

The true test case is presented in Fig. 2 using the prepared test sets of 7000 URLs from each of the Black and White sources. As the White training set size increases exponentially, a roughly linear growth in the false negatives may be observed. At the same time we observe decay in the false positive reports, which is more exponential than linear.

It would seem that there is indeed a predictable sensitivity of the classifier to the relative sizes of the training sets. This has a potential to be used for tuning the relationship between the false positive and false negative rates. This can be quite useful as false positives can be particularly annoying to users making them lose faith in a system, with the possible consequence of ignoring its suggestions or dispensing with it completely.

3.2.2 Sensitivity to the Absolute Size of the Training Sets

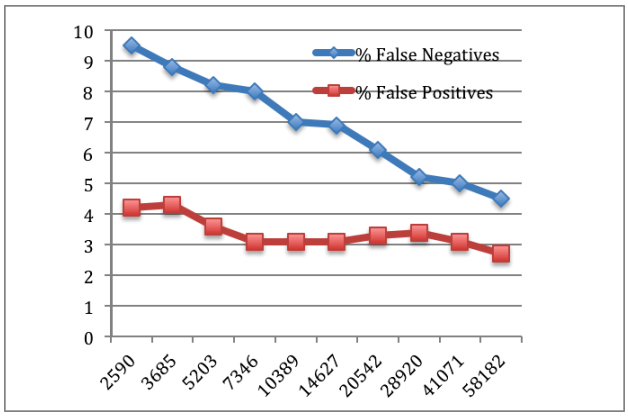

Figure 3. Failure rate vs training set size

Fig. 3 presents the results of varying the size of both the White and Black training sets while maintaining their approximate size equality. As with Figs 1 & 2 we double the size of the training sets every two steps. We observe a roughly linear decrease in both the false positive and false negative rates as the training set size increases exponentially. (Actually the false negative result is relatively flat). In each test we have used of test set of about 7,000 Black and 7,000 White URLs, which exceeds the size of the training set for the first three samples.

While a predictable improvement in performance is apparent from these data, it is not massive; particularly so for false positives. It is quite possible that further improvements would be seen if we could increase the training set size significantly. However, although we have available over 3,000,000 White URLs we are currently limited by the availability of Black URLs (37,000) and so cannot perform further tests at this stage.

3.3 Effects of General Feature Set Selection

We now consider the effects of discriminating specific URL feature sets. In all cases similarly sized Black (32,910) and White (34,782) training sets are used with testing sets of 1,897 black and 1,951 white URLs.

Table 1. Failure rates using various components of URL

URL Parts	%False Negatives	%False Positives	%Failure
Host Only	16.3	7.9	24.2
Host & Path	4.1	3.6	7.7
Host & Path Distinguished	4.2	3.1	7.3

The first row (Table 1) shows very poor performance when only the Host component is considered. The performance improves dramatically in the second row where the Path component is considered as well. Finally the third row shows a marginal improvement when the Path elements are distinguished from the Host elements.

Here, there is definite advantage in considering more, rather than fewer, features. While distinguishing these feature sets lead to a slight (0.1%) increase in false negatives, the total failure rate dropped by 0.4% so such a distinction seems worthwhile in this case.

Unless otherwise noted the following tests all rely on using and distinguishing the Host and Path feature sets. Indeed the above tests for sensitivity used this method exclusively (i.e. no query component was considered).

Table 2. Failure rates with various Query elements distinguished

Query Features	%False Negatives	%False Positives	%Failure
Attr=Value	4.3	2.9	7.2
Attr & Value	4.6	2.5	7.1
Attr & Value Distinguished	4.5	2.4	6.9

Our second test considers the utility of the query string (present in only a relatively small minority of the URLs in our data sets) in concert with distinguished Host and Path feature sets. Each element of a query is of the form attribute=value.

The first row of Table 2 shows the result of treating such composite elements as a single feature. The second row breaks the element into attribute and value features but does not distinguish them further while the third row distinguishes an attribute feature set from a value set.

At each step we see a steady decline in the total failure rate from the best in Table 1. This is driven by a decline in the false positive rate while the false negative rate fluctuates but is always larger than the last two rows of Table 1.

Table 3. Failure rates with various Host elements distinguished

Host Breakup	%False Negatives	%False Positives	%Failure
2	4.4	3.3	7.7
3	3.3	7.1	10.4
4	3.7	7.7	11.4

Our third test extends the results of Table 1 (i.e. no Queries are considered) by examining the benefits of further distinguishing the host elements. In Table 3 the first line shows the effect of distinguishing the top-level-domain from the remaining Host feature set. Both the false negatives and false positives are worse than the results from the third row of Table 1. In subsequent rows further Host elements are distinguished (proceeding from the right) all resulting in an increased total failure rate, primarily driven by an increased false positive rate, which is certainly undesirable.

4. CONCLUSION AND FUTURE WORK

We have examined the performance of simple binary Naïve Bayesian text classifier with particular regard to both relative and absolute training set size and the effects of various feature set selection and distinction strategies.

The results seem to indicate that it is generally beneficial to increase the number of substantive feature sets considered, but that there is a limit to the degree that these feature sets may be subdivided and further distinguished.

Our results contain an asymmetry between rates of false positives and false negatives. In tests where the total failure rate drops, it is driven by a drop in the false positive rate often with a marginal rise in the false negative rate. While false positives can lead to problems with the utility of a classifier (particularly that a user may cease to trust it), its purpose is served only if it can credibly predict a large percentage of black sites. We suspect that this asymmetry is a direct result of the sizes of the black and white sets being separated by two orders of magnitude. It seems likely that the very large white set simply better samples the space of (white) URLs and that when we extract a random selection from it that diversity of information is carried into the classifier.

Our further work will necessitate the acquisition of a significantly larger and more diverse Black data set so that both the effects of large training sets (black and white) may be observed and also that the above asymmetry may be removed.

ACKNOWLEDGEMENT

The authors would like to thank Alpha Dot Net Australia for their valuable "in-kind" support and facilities provision.

REFERENCES

[1] PhishTank, databases available at: www.phishtank.com/developer_info.php

[2] The Antiphishing Working Group (APWG), www.apwg.org or www.antiphishing.org

[3] J. Chen, C. Guo, "Online Detection and Prevention of Phishing Attacks", in *Communications and Networking in China*, (ChinaCom '06), Oct. 2006, pp.1-7

[4] "Phishing Activity Trends Report 4th Quarter 2012", APWG, April 2013, available: http://docs.apwg.org/reports/apwg_trends_report_q2_2012.pdf

[5] "Phishing Activity Trends Report 2nd Quarter 2013", APWG, Nov. 2013, available: http://docs.apwg.org/reports/apwg_trends_report_q2_2013.pdf

[6] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails", in *Proceedings of the 16th international conference on World Wide Web* (WWW '07), ACM, New York, 2007, pp649-656.

[7] Y. Zhang, S. Egelman, L. Cranor and J. Hong, "Phinding Phish: Evaluating Anti-Phishing Tools", Human-Computer Interaction Institute, Paper 76. 2006, available: http://repository.cmu.edu/hcii/76

[8] R. Dhamija, J. D. Tygar, and M. Hearst. "Why phishing works", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '06), ACM, New York, 2006, pp.581-590.

[9] J. Ma, L. K. Saul, S. Savage and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious URLs", in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '09), ACM, New York, 2009, pp.1245-1254.

[10] J. Ma, L. K. Saul, S. Savage and G. M. Voelker, "Identifying suspicious URLs: an application of large-scale online learning", in *Proceedings of the 26th Annual International Conference on Machine Learning* (ICML '09). ACM, New York, 2009, pp.681-688.

[11] J. Ma, L. K. Saul, S. Savage and G. M. Voelker, "Learning to detect malicious URLs", *ACM Trans. Intell. Syst. Technol.*, 2, 3, Article 30 (May 2011), pp.24

[12] Private Communication with Teslstra systems administration staff.

[13] Open Directory Project. Database available: www.dmoz.org/rdf.html

[14] F. Sebastiani, "Machine learning in automated text categorization", *ACM Comput. Surv.* Vol 34,1, March 2002, pp.1-47