E-Mail Classification for Phishing Defense

Wilfried N. Gansterer, David Pölz

University of Vienna, Research Lab Computational Technologies and Applications

Abstract. We discuss a classification-based approach for filtering phishing messages in an e-mail stream. Upon arrival, various features of every e-mail are extracted. This forms the basis of a classification process which detects potentially harmful phishing messages. We introduce various new features for identifying phishing e-mail and rank established as well as newly introduced features according to their significance for this classification problem. Moreover, in contrast to classical binary classification approaches (spam vs. not spam), a more refined ternary classification approach for filtering e-mail data is investigated which automatically distinguishes three message types: ham (solicited e-mail), spam, and phishing. Experiments with representative data sets illustrate that our approach yields significantly better classification results than existing phishing detection methods. Moreover, the direct ternary classification proposed is compared to a sequence of two binary classification processes. Direct one-step ternary classification is not only more efficient, but is also shown to achieve better accuracy than repeated binary classification.

1 Introduction

In recent years, phishing ("password fishing") has become an enormous problem and threat for all big internet based commercial operations. The term covers various criminal activities which try to fraudulently acquire sensitive data or financial account credentials from internet users, such as account user names, passwords or credit card details [1]. Phishing attacks use both social engineering and technical means in order to get access to such data. The damage caused by successful phishing attempts was estimated a six digit Euro amount in Austria in 2006[1] and more than four million Euro in Germany[2]. These figures do *not* yet account for image loss and reduced customer trust.

A central component in social engineering-based schemes is the abuse of *e-mail* communication. Unsolicited e-mail messages are masqueraded in order to pretend that they come from a trustworthy entity and lead users to counterfeit web sites which ask the recipients to divulge sensitive data. The research summarized in this paper investigates new methods for filtering phishing e-mail messages. The focus is on classification-based techniques, in particular, on the definition of properly suited feature sets, on the application of feature selection

[1] `http://www.bmi.gv.at/bmireader/documents/428.pdf`
[2] `http://www.at-mix.de/news/1226.html`

methods, and on machine learning methods for detecting e-mail phishing attempts. More specifically, as an alternative to classical binary spam filtering, we investigate the problem as a *ternary* classification problem, i. e., automatically distinguishing *three* classes of messages in e-mail data: solicted and legitimate messages ("ham"); unsolicited, but otherwise harmless messages ("spam"); and phishing messages, which are also unsolicited, but dangerous and potentially harmful. Obviously, this problem can also be addressed with a sequence of two binary classification problems, for example, first applying a binary filter which separates unsolicited e-mail from solicited e-mail, and then applying a binary filter on the result which separates spam and phishing. Such a two-step approach does not only have disadvantages in terms of overhead and efficiency compared to a direct ternary approach, but also achieves lower accuracy, as we will illustrate.

Related Work. A lot of research has been done on filtering *spam* e-mail (unsolicted commercial or bulk e-mail), an older problem than phishing (see, for example, [11, 5, 7] and the references therein).

Relatively little research has been done so far on specifically detecting phishing e-mail, which usually differs significantly in its properties and characteristics from spam e-mail. The latter is—even for humans—usually much easier to distinguish from legitimate e-mail messages than phishing e-mail, which is deliberately designed to look like a legitimate message from a trusted contact. Existing work ranges from theoretically oriented academic research efforts to tools for practical use which are often based on heuristics and ad-hoc strategies. Among the former, we find concepts based on changing or enhancing the graphical user interfaces and on modifying the workflow of the user, for example, in projects like WebWallet [18], AntiPhish [8] or SPS [13]. Unfortunately, such approaches impose additional work upon the user. Thus, often these additional security features are disabled. The method proposed in [12] tries to detect phishing web sites by sending faked formular data and analyzing the returned answers. It is based on the assumption that many phishing web sites do not check input data, whereas an authentic web site would usually check the input and produce error messages. A complementary approach to thwart social engineering attacks such as phishing attempts is to raise the awareness and education of users [6, 9, 16].

The most promising methods utilize the general concept of feature-based phishing detection. In [10], key words are extracted from every e-mail, and then the web pages linked within the e-mail are compared with web sites which are close to these key words based on their *visual layout*. In a related approach, a browser plugin which analyzes the content of a website referred to from an e-mail has been described in [19]. First, keywords are extracted from the website and queried in a search engine. Then, domain names linked from the e-mail are compared to the search results. Suggestions for six e-mail features which are indicators for a phishing message mail have been made in [2]. Another feature-based approach called *PILFER* has been described in [4], where ten features are used for deciding whether an e-mail is considered a phishing message. For a binary classification of ham vs. phishing messages, an overall accuracy of 99,5%, 0,2% false positive rate, and 4% false negative rate is reported.

Some of the concepts developed have been integrated into tools. Apache's Spamassassin system (`http://spamassassin.apache.org/`) is a widely used server side solution which has become a de-facto standard, primarily for filtering spam messages. It is based on a set of more than 700 static rules. Over time, some rules have been integrated which specifically target phishing messages. Other tools enhance or alter the information provided about websites visited in the graphical user interface of the web browser. The idea is to visually warn the user about the web site referred to. Based on this information provided the user has to decide whether he trusts the web site or not. Examples are toolbars such as Siteadvisor (`http://www.siteadvisor.com/`), Netcraft (`http://toolbar.netcraft.com/`), or the Google toolbar (`http://www.google.com/tools/firefox/safebrowsing/`). A drawback of such toolbars is that they act *after* the user has already decided to follow a link provided in an e-mail. It is preferable to provide an earlier line of defense in the e-mail client. Mozilla's Thunderbird or Microsoft's Outlook (as well as the respective browsers) use combinations of (URL) black- and whitelists and various heuristics for identifying a phishing message before the user has followed any link in the message.

Approach Taken in this Paper. Concepts based on graphical user interfaces have various potential problems: The toolbars may fail to provide the proper judgement of a web site, and many users do not pay enough attention to or misinterpret the information provided. This leads to rather poor efficiency of toolbars. Investigations have shown that only 40-50 percent of the phishing attempts were spotted successfully by the users based on the information provided by toolbars [15, 17].

The main objective of the work summarized here was to improve on existing anti-phishing methods. Conceptually, feature-based e-mail classification is the most promising approach. The method discussed in this paper further extends and refines feature-based phishing detection, on the one hand by adding new features which turn out to be crucial for the classification accuracy, on the other hand by addressing the ternary classification problem formulated above instead of the classical binary one. This allows for specifically targeting phishing messages in the enormous volume of regular unsolicited e-mail. Moreover, in contrast to some of the related work, the approach pursued here can be implemented server-based, which has several advantages: (*i*) efficiency can be improved compared to repeatedly performing the same tests on identical messages at the client-side; (*ii*) it is not required to support different e-mail clients; and (*iii*) a lot of the responsibility and overhead (setup, administration, maintenance) caused by phishing filtering can be taken away from the end user.

2 Methodology

We adopt the basic structure of a classification process: Based on a fixed set of features which results from a feature selection and feature ranking process, the values of these features are extracted from each e-mail message to be classified

(feature extraction). These extracted feature values are the basis for the actual classification process.

Although there are analogies between e-mail phishing and e-mail spamming (phishers profit from the fact that sending out large numbers of e-mail is very cheap and the revenue from one phished account is potentially rather big), there are also important differences. In contrast to spam attacks an individual phishing attack tends to be carried out over a much shorter period of time [1] (because of the more aggressive prosecution of phishing web sites). Moreover, the structure of phishing messages tends to differ significantly from the structure of spam messages, but it may be quite close to the structure of regular ham messages (because for a phishing message it is particularly important to look like a regular message from a trustworthy source). Consequently, it is pivotal to select good features as the basis for the classification process.

Some of the e-mail features used in our approach are based on results reported in the literature [2–4]. They are summarized in Section 2.1. Some other features are newly introduced in Section 2.2. In Section 2.3 we discuss the method used for ranking and selecting the features according to their significance, and we summarize the classification methods applied.

2.1 E-Mail Features Already Used in the Literature

The following fifteen e-mail features have been proposed earlier in [2–4]. The abbreviations introduced here and in the following section are used throughout the rest of the paper.

Number of links (NoLinks) [4] counts the number of links which are included in the body of an e-mail. *Number of different domains (NoDifDom)* [4] counts the number of different domains that are linked from within the e-mail.

HTML-mail (HTML) [4], a binary feature, checks if the e-mail is an HTML mail by inspecting the Content-Type parameter in the e-mail header. *HTML-form (HTMLForm)* [3], a binary feature, checks if the e-mail is an HTML e-mail which contains an HTML form element. Thunderbird's anti-phishing tool uses this feature as one of three decision variables.

Link-target differs from link-text (DifLinktar) [3, 4] counts the number of links in the e-mail for which the link text does not contain the domain name of the link target. *Link-domain differs from sender-domain (LinkDifSender)* [2] counts how many links point to a different domain than the domain from where the e-mail was sent.

Number of dots in a domain (NoDots) [4] counts the maximum number of dots in any linked domain in the e-mail. *URL contains IP address (UrlIP)* [3, 4] counts how many links in a message contain an IP address. *URL contains @ (UrlAt)* [3] counts how many links in a message contain the "@" character. *URL contains hexadecimal characters (UrlHex)* [3] counts how many links in a message contain hexadecimal characters or URL-escaped characters, which make links unreadable to a human reader. *URL contains a non-standard port (UrlPort)* [3] counts how many links in a message contain a non-standard port (other than 80 or 443).

Use of JavaScript pop-ups (JSPopup) [3] counts the number of pop-ups that are created by the links in the e-mail message. *SSL SelfSigned (SSLSS)* [3] counts how many links in the e-mail message point to a website that encrypts the connection with a self signed certificate. *Compare DNS and rDNS of links (DNSrevDNS)* [2] counts how many domain names within the links of an e-mail message do not have a corresponding reverse DNS entry.

Spamassassin (SaXX) [4] denotes a set of forty boolean features taken out of the Spamassassin rule set. It contains those Spamassassin rules which yielded the highest information gain on our training set. A spam threshold of five was chosen, and only static local rules were considered. Blacklist lookups as well as Bayesian filters were omitted.

2.2 New E-Mail Features Introduced

Sixteen new features summarized in the following. Their relevance for the ternary classification problem stated in Section 1 is investigated in this paper for the first time. These newly introduced features belong to three different groups: The first group contains six "off-line" features, and the second group contains eight "online" features. The third group is a control group of presumably class-independent features containing two features: *Subject length (SubjectLen)* counts the number of characters in the subject field, and *Sender length (SenderLen)* counts the number of characters in the sender field of a message.

Off-line Features. These features can all be extracted locally and quite efficiently. Consequently, they are well suited for a high-load context as it has to be handled on large mailservers.

Number of pictures used as link (NoPicLink) counts the number of pictures which are linked to web sites. *Image maps used as link (NoMapLink)* counts the number of pictures with image maps that are linked to web sites.

URL contains non ASCII characters (UrlChar) counts how many links contain standard ASCII character look-a-likes. Some of these characters (for example, some cyrillic letters) look almost identical to some "normal" ASCII characters so that a user will not see the difference between a legitimate and a spoofed website. *Message size (MesSize)* denotes the size of the e-mail message in bytes.

Countries of links (ColXX) determines the countries of servers which are linked from inside an e-mail based on their IP addresses. Since statistics show that almost 60% of all phishing messages link to only two countries [1], this information can give a hint on the reliability and security of linked servers. The feature is not a single value but consists of 51 numeric values that represent the number of links to 50 different countries and the link targets which cannot be assigned to a certain country. *Signed Mail (Signed)*, a binary feature, checks whether the e-mail has been signed.

Online Features. The extraction of the "online" features imposes a significantly higher cost. Since it is based on internet connections, the time required for the associated queries may vary widely (depending on the status of the internet connection). Consequently, the extraction of these online features may cause

serious performance bottlenecks in practical high load situations (large-scale business e-mail servers, etc.) and severely restrict performance and scalability of the e-mail filtering system. In order to overcome these problems, it is crucial to restrict the number of messages for which such features need to be extracted. One possibility is to apply a layered approach, for example, using *multilevel greylisting* as proposed in [7], where it has been shown that currently much less than 1% of the incoming e-mail messages reach the highest levels in this framework. Due to this enormous reduction in the number of messages and the decoupling of SMTP connections from feature extraction and classification processes in the context of multilevel greylisting, the extraction of costly online features is restricted to very few messages (those which could not be classified unambiguously at the previous levels). This solves the scalabilty problems and eliminates the performance bottleneck of online features. In the following, the newly introduced online features are summarized.

Number of OnClickEvents in the e-mail (NoOCE) counts the use of OnClick-Events in embedded and linked JavaScript code. *HTML-form SSL protected (HTMLSSL)*, a binary feature, determines whether an HTML formular contained in a website is SSL protected.

JavaScript statusbar manipulation (JSStatus) counts the number of statusbar alteration attempts within all linked websites. *Link domain differs from JavaScript domain (DifLinkJS)* counts how many JavaScript parts are loaded from a domain which differs from any other domain linked within the message.

The last four online features are based on queries to search engines. *Result quantity of sender domain (NoSendRes)* counts the hits of a search for the domain name of the sender. *Result quantity of link domain (NoLinkRes)* denotes the *lowest* number of hits when searching for the domain names of the links contained in an e-mail message. *Link domains differ from search results (DifLinkRes)* counts the number of domains linked from the e-mail message which do not match any of the first ten hits from a search for the domain of the FROM-field (sender) of the message. *Distance of message text and linked domain (DifTextLink)* tries to evaluate the "distance" of the message text from the domain names which it links to. For this purpose, five keywords are extracted from the message text using the automatic keyword creating algorithm of the classifier4J software (`http://classifier4j.sourceforge.net/`). These keywords are sent to a search engine individually and also in a combined query. For each of these six queries, the domains of the ten highest ranked hits are compared to the domains linked from the e-mail. The feature value is defined as the number of links in the message which are not found in *any* of the top ten hits of the queries.

2.3 Feature Ranking and Classification

One option for ranking the message features listed above according to how well they differentiate the three classes ham, spam, and phishing would be to use the *information gain*, defined as $\text{gain}(X, C) := \text{info}(C) - \text{info}_x(C)$ for a set of class labels C and a feature set X. The function info(C) is Shannon's entropy function

and info_x is the conditional entropy function defined as $\text{info}_x(C) := \sum_{v \in X} P(v) * P(C|v)$, where $P(v)$ is the probability of v and $P(C|v)$ the conditional probability of C given v.

However, the information gain favors features which assume many different values. In order to address this, we instead rank features according to their *information gain ratio*:

$$\text{gainratio}(X, C) := \frac{\text{gain}(X, C)}{\text{splitinfo}(X)}$$

with $\text{splitinfo}(X) := -\sum_{v \in X} P(v) * \log_2 P(v)$.

For the classification process, we compared several different methods (see Section 3.2) from the open source software Weka (`http://www.cs.waikato.ac.nz/ml/weka/`).

Implementation. We implemented the feature extraction and classification functionality as a plug-in for the Apache James server (`http://james.apache.org/`). The feature ranking is based on the ratio-gain algorithm [14] in Weka.

The extraction of the online features introduced in Section 2.2 has been implemented by randomly querying one of the three search engines Google, Yahoo and MSN. Since each of them limits the number of daily search requests from the same source, it can be required to alternate between them, which also adds some variance to these features. Even if the same search engine is used all the time, the online features based on web search results will change over time. However, experience shows that these variations do not have a negative influence on the relevance of the features for classification.

3 Experimental Evaluation

Our prototype system was tested on an Intel Duo Core E6600 system with 2 GB RAM and a Linux operating system. Our plug-in implemented in Java was integrated into the Apache James server. Test data were sent from an e-mail collection in mbox format using Google Mail Loader (`http://marklyon.org/gmail/`).

3.1 Test Data

The data used for the evaluation of our method has been obtained from two sources: A sample set of 11 000 phishing messages from May 2007 was kindly made available to us by the *Phishery* (`http://phishery.internetdefence.net/`); and from the 2007 TREC corpus (`http://trec.nist.gov/data/spam.html`), which consists of roughly 25 000 ham and 52 000 spam messages.

As training set we selected the *oldest* 4000 e-mails of each class. As test set we used the *newest* 1000 e-mail messages of each class. This chronological ordering of the test data and training on historical data allows for simulating the changes and adaptations in spam and phishing messages which occur in practice.

3.2 Results

In order to account for the widely differing values of different features, the data was normalized and discretized into equal frequency bins before the ratio-gain algorithm was run for feature ranking. This leads to different bin sizes, but splits the data more evenly into the bins and improves the results achieved with the ratio-gain algorithm.

Feature Ranking and Feature Selection. Figure 1 shows the ranking of the best 30 features for the ternary classification problem and the corresponding ratio-gain values for the ternary and the binary classification scenario. For the binary case, spam and phishing were merged into a single class. The features newly introduced in this paper are marked with double brackets ("<< ... >>"). Note that the first two positions are taken by features already used before, but position three to nine are occupied by new features. Seven out of the top ten features are new. Figure 1 also shows that a feature ranking for the binary classification problem looks differently. In this case the number of links where the source country is Austria, Switzerland or unidentifiable (ColAT, ColCH, Col-) turn out to be the most relevant features.

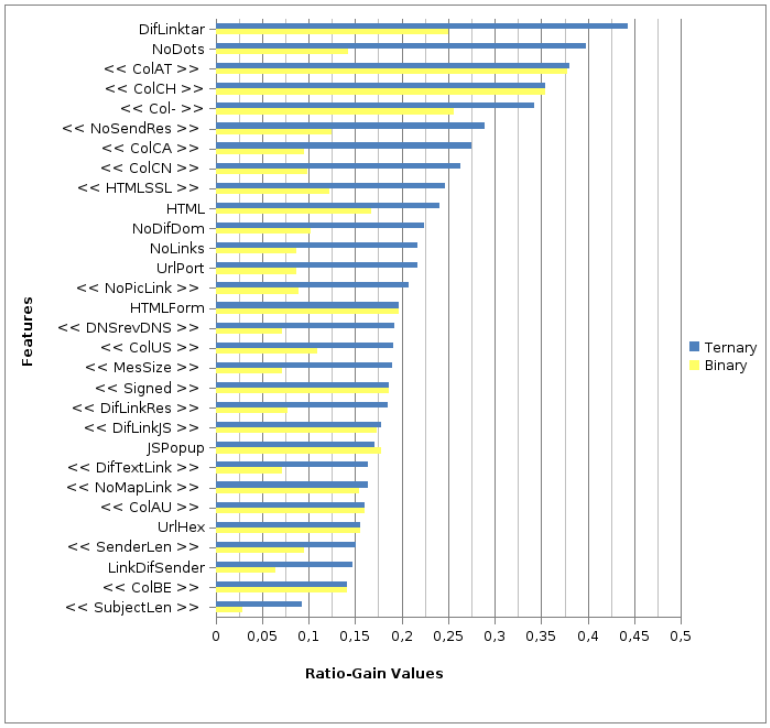

Fig. 1. Ratio gain values for the top 30 features of feature set F_2

Comparison of Feature Sets. In the following, we compare the classification results for three different feature sets. Feature set F_1 constains all features that were explained in Sections 2.1 and 2.2, feature set F_2 contains the features F_1 *without* the SaXX features, and feature set F_3 contains only the features used in the literature as listed in Section 2.1. Table 1 summarizes the classification results of a decision tree generated with the J48 algorithm for the feature sets F_1 and F_2. Overall, 94,1% of all messages were classified correctly based on the feature set F_1, but only 91,5% based on the feature set F_2, which illustrates the improvement from integrating the SaXX features. When using only a subset of F_2, the classification accuracy decreases rapidly: When using the top 20 features from Figure 1, it goes down to 87%, when using only the top 10 features, it goes down to 84,6%.

feature set F_1						feature set F_2						
as Ham		as Spam		as Phish		as Ham		as Spam		as Phish		
979	97,9%	21	2,1%	0	0%	981	98,1%	19	1,9%	0	0%	Ham
35	3,5%	913	91,3%	52	5,2%	36	3,6%	931	93,1%	33	3,3%	Spam
3	0,3%	65	6,5%	932	93,2%	5	0,5%	161	16,1%	834	83,4%	Phish

Table 1. Classification with a J48 decision tree based on feature sets F_1 and F_2

To examine the improvement achieved with the new features introduced we trained an SVM classifier based on feature set F_3. This resulted in an overall accuracy of only 73,3% and a false positive rate for ham e-mails of 13,8%, which is much worse than the accuracy achieved with either F_1 or F_2 or subsets of F_2, which illustrates the importance of the new features we introduced.

Comparison of Classifiers. Table 2 compares the overall percentages of correctly classified messages of a J48 decision tree (also in bagged and boosted variants), a random forest (RF), a BayesMultinomial classifier (BM), a support vector machine (SVM), and a k-nearest neighbor algorithm (kNN) based on the feature sets F_1 and F_2. Clear differences between various classifiers can be observed. Overall, the SVM achieves the highest accuracy. The differences between feature sets F_1 and F_2 again illustrate the effect of the SaXX features.

feature set	J48	bagged J48	boosted J48	RF	BM	SVM	kNN
F_1	94,1%	93,5%	95,2%	93,7%	65,0%	97,0%	92,7%
F_2	91,5%	91,4%	92,1%	92,1%	62,0%	92,1%	90,6%

Table 2. Accuracies of various classification methods (ternary classification problem)

Weka supports the construction of *cost-sensitive* classifiers by associating misclassification costs with each class. For the SVM, we increased the costs for misclassified ham emails (false positives) to five times the costs of other types of misclassifications. Table 3 shows the results. Compared to Table 1, the number of misclassified ham e-mails was reduced by more than a third, but compared to Table 2, the overall accuracy of the SVM classifier was reduced to 95,6%.

as Ham		as Spam		as Phish		
987	98,7%	13	1,3%	0	0%	Ham
79	7,9%	905	90,5%	16	1,6%	Spam
4	0,4%	19	1,9%	977	97,7%	Phish

Table 3. Cost-sensitive SVM classifier based on feature set F_1

Ternary vs. Binary Classification. So far, only the accuracies achieved for the ternary classification problem were summarized. To put them into perspective, we first compared them with a *binary* ham vs. spam+phishing classification performed by full Spamassassin. We considered phishing e-mails correctly classified if they reached a score of 5 points or more and thus were put into the spam category by Spamassassin. On our test set Spamassassin assigned 93,7% of the messages correctly to the two classes. The false positive rate was 0,3% and the false negative rate 4,7%. Except for the false positive rate this is comparable to the results shown in Table 1. We also compared our results with the scam e-mail detection implemented in the Thunderbird e-mail client. On our test data set, this system performed much worse. It classified only 75,8% of the messages correctly with a false positive rate of 5,7% and a false negative rate of 18,9%.

In order to compare our feature set with the one Spamassassin uses, we ran some more tests with the SVM classifier based on the F_1 feature set (see Table 4). In the first test, we performed a binary classification ham vs. spam+phishing. 98,2% of the messages were correctly classified with a false positive rate of 1,9%. Note that this is better than the binary Spamassassin classification, and also better than the ternary SVM classification. In the second test, we performed a binary classification ham+spam vs. phishing. In this scenario the SVM classified 98,7% of the messages correctly with a false positive rate of 0,8%. In the third test, we performed a binary classification ham+phishing vs. spam. This resulted in an overall accuracy of 95,7% and a false positive rate of 1,7%. However, after completely removing the spam class from the data set and classifying ham vs. phishing, our feature set achieved an accuracy of 99,7% with a false positive rate of 0,2%. This is slightly better than the accuracy reported in [4]. Finally, we emulated a ternary classification process as a sequence of two binary classification processes. First separating spam from phishing+ham and in a second step separating ham and phishing e-mails in the result yields an accuracy of 95%, which is worse than the ternary classification results (cf. Table 2).

Binary classes		Accuracy	False positive rate
Ham	Spam+Phish	98,2%	1,9%
Ham+Spam	Phish	98,7%	0,8%
Ham+Phish	Spam	95,7%	1,7%
Ham	Phish	99,7%	0,2%

Table 4. Binary classification results of SVM with feature set F_1

4 Conclusion

A ternary classification approach for distinguishing three groups of e-mail messages in an incoming stream (ham, spam, and phishing) has been investigated. The classification is based on a partly new designed set of features to be extracted from each incoming message. Various classifiers have been compared to assign them into one of the three groups. Over all three groups, a classification accuracy of 97% was achieved, which is better than solving the ternary classification problem by a sequence of two binary classifiers. In particular, it was illustrated that the methodology proposed achieved a significantly better accuracy than the widespread Spamassasin system with a binary classification.

In the future, we will focus on further reducing the false positive rates (ham wrongly classified as spam or phishing). Moreover, we will investigate improvements of the subset selection strategies in order to further reduce the number of features without loss of classification accuracy.

Acknowledgments. This research was partly supported by Internet Privatstiftung Austria.

References

1. Anti Phishing Work Group. Phishing attacks trends report. `http://www.antiphishing.org`, Dec. 2007.
2. A. Inomata, Sk. Md. M. Rahman, T. Okamoto, and E. Okamoto. A novel mail filtering method against phishing. Japan Science and Technology agency, Research Institute of Science and Technology for Society, 2005.
3. C.E. Drake, J.J. Oliver, and E.J. Koontz. Anatomy of a Phishing Email. In *Conference on E-mail and Anti-Spam*, 1841 Page Mill Road, Palo Alto, CA 94304, USA, 2004. MailFrontier, Inc.
4. I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 649–656, New York, NY, USA, 2007. ACM.
5. W. N. Gansterer, A. G. K. Janecek, and R. Neumayer. Spam filtering based on latent semantic indexing. In M. W. Berry and M. Castellanos, editors, *Survey of Text Mining II: Clustering, Classification, and Retrieval*, pages 165–183. Springer-Verlag, 2008.
6. M. Jakobsson and J. Ratkiewicz. Designing ethical phishing experiments: a study of (rot13) ronl query features. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *World Wide Web Conference*, pages 513–522. ACM, 2006.

7. A. G. K. Janecek, W. N. Gansterer, and K. A. Kumar. Multi-level reputation-based greylisting. In *Proceedings of ARES 2008 – International Conference on Availability, Reliability and Security*, pages 10–17. IEEE Computer Society, 2008.
8. E. Kirda and C. Kruegel. Protecting users against phishing attacks with antiphish. *29th Annual International Computer Software and Applications Conference Volume 1*, 01:517–524, 2005.
9. P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge. Protecting people from phishing: the design and evaluation of an embedded training email system. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 905–914, New York, NY, USA, 2007. ACM.
10. W. Liu, X. Deng, G. Huang, and A. Y. Fu. An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing*, 10(2):58–65, 2006.
11. T. R. Lynam, G. V. Cormack, and D. R. Cheriton. On-line spam filter fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pages 123–130, New York, NY, USA, 2006. ACM Press.
12. M. Chandrasekaran, R. Chinchani, and S. Upadhyaya. Phoney: Mimicking user response to detect phishing attacks. *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 00:668–672, 2006.
13. D. Miyamoto, H. Hazeyama, and Y. Kadobayashi. SPS: A simple filtering algorithm to thwart phishing attacks. In *Technologies for Advanced Heterogeneous Networks*, volume Volume 3837/2005, pages 195–209. Springer Verlag, 2005.
14. J. R. Quinlan. *C4.5: Programms for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.
15. S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators. In *IEEE Symposium on Security and Privacy*, pages 51–65. IEEE Computer Society, 2007.
16. S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong, and E. Nunge. Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish. In *SOUPS '07: Proceedings of the 3rd symposium on Usable privacy and security*, pages 88–99, New York, NY, USA, 2007. ACM.
17. M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, New York, NY, USA, 2006. ACM.
18. M. Wu, R. C. Miller, and G. Little. Web wallet: preventing phishing attacks by revealing user intentions. In *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, pages 102–113, New York, NY, USA, 2006. ACM Press.
19. Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 639–648, New York, NY, USA, 2007. ACM.