Detecting Phishing Emails the Natural Language Way

Rakesh Verma[1], Narasimha Shashidhar[2], and Nabil Hossain[3]

[1] Department of Computer Science, University of Houston
rmverma@cs.uh.edu
[2] Department of Computer Science, Sam Houston State University
karpoor@shsu.edu
[3] Division of Science, Mathematics, and Computing, Bard College
nh1682@bard.edu

Abstract. Phishing causes billions of dollars in damage every year and poses a serious threat to the Internet economy. Email is still the most commonly used medium to launch phishing attacks [1]. In this paper, we present a comprehensive natural language based scheme to detect phishing emails using features that are invariant and fundamentally characterize phishing. Our scheme utilizes all the *information* present in an email, namely, the header, the links and the text in the body. Although it is obvious that a phishing email is designed to elicit an action from the intended victim, none of the existing detection schemes use this fact to identify phishing emails. Our detection protocol is designed specifically to distinguish between "actionable" and "informational" emails. To this end, we incorporate natural language techniques in phishing detection. We also utilize contextual information, when available, to detect phishing: we study the problem of phishing detection within the contextual confines of the user's email box and demonstrate that context plays an important role in detection. To the best of our knowledge, this is the first scheme that utilizes natural language techniques and contextual information to detect phishing. We show that our scheme outperforms existing phishing detection schemes. Finally, our protocol detects phishing at the email level rather than detecting masqueraded websites. This is crucial to prevent the victim from clicking any harmful links in the email. Our implementation called **PhishNet-NLP**, operates between a user's mail transfer agent (MTA) and mail user agent (MUA) and processes each arriving email for phishing attacks even before reaching the inbox.

1 Introduction

Phishing is a social engineering threat aimed at gleaning sensitive information from unsuspecting victims. Attacks are typically carried out via communication channels such as email or instant messaging by attackers masquerading as legitimate and trustworthy entities. In this paper, we focus only on email communication as it is the most popular medium to launch such attacks [1]. As observed

S. Foresti, M. Yung, and F. Martinelli (Eds.): ESORICS 2012, LNCS 7459, pp. 824–841, 2012.

before [2], detecting phishing email messages automatically is a non-trivial task. Our primary contribution in this paper is a comprehensive and effective natural language based phishing detection scheme. Our scheme uses the information present in the email header, text in the email body and the links embedded in the email. We make use of novel techniques to process the header and link information, and deeper natural language techniques to process the text information. To the best of our knowledge, this is the first natural language based scheme for phishing detection.

Natural language processing (NLP) by computers is well-recognized to be a very challenging task because of the inherent ambiguity and rich structure of natural languages. Perhaps this explains why previous researchers have not used NLP techniques for email phishing detection. Despite this difficulty, we show that our scheme outperforms all existing phishing detection strategies in the literature and obtains a phishing detection rate of 97% or better with very low false positives (0.7-0.8%). Our scheme is built on the observation that the fundamental difference between a phishing and a legitimate email lies in its objective. While a legitimate email typically conveys some information to the reader, a phishing email is designed to *elicit* a response. This response often involves making the reader click a link with the intention of obtaining personal sensitive information. None of the detection schemes in the literature make use of this distinction to detect phishing emails. Our scheme is designed specifically to distinguish between "actionable" and "informational" emails. We focus on objectives that are typical of phishing emails - language that intends to create a sense of urgency, threat, worry, concern or offers an incentive to the user to perform an action. Our scheme uses contextual information (when available) to detect phishing. We study the problem of phishing detection within the contextual confines of the user's mail box and show that context plays a significant role in detection. We show that contextual phishing detection outperforms many other non-contextual detection schemes in the literature and is the first contextual scheme to the best of our knowledge. Moreover, the use of context information makes our scheme robust against attacks that are aware of our methods.

Finally, we believe in detecting phishing at the email level rather than detecting fraudulent and masqueraded websites after the website has been visited by the user. Our implementation PhishNet-NLP operates between a user's MTA and MUA and processes each arriving email for phishing attacks. This prevents the user from clicking any harmful link in the email. This approach is in contrast to schemes that analyze the target websites for authenticity. The motivation to operate at the email level is due to the fact that clicking on the link and visiting a phishing website exposes the user to potential malware that could be installed by the website. Furthermore, it is our objective to maximize the *distance* between the user and the phisher - clicking a malicious link puts the user closer to the threat. The added advantage of this approach is that ISPs and email providers

may now be able to prevent such emails from being delivered to the user thereby saving precious bandwidth as well.

2 Prior Work

Phishing is primarily a social engineering attack and has attracted a lot of research interest in this context. Different research groups have studied this problem from various perspectives: server-side and browser-side strategies, education/training, evaluation of anti-phishing tools, detection schemes and finally studies that analyze the reasons behind the success of phishing attacks. We note that phishing has been studied extensively. Here, for lack of space, we briefly outline the prior related work on phishing categorized by research objectives.

Phishing Detection Schemes - Email and Web pages. There are two primary classifications of phishing detection schemes: schemes that detect phishing based on analyzing the content of the target web pages (targets of the embedded email links) and schemes that operate directly on the content of the emails. The schemes for detecting phishing attacks (email and web pages) in the literature can be broadly classified into three categories: 1. Schemes based on information retrieval, 2. Machine learning based techniques and 3. String, pattern and visual matching based detection schemes. Before the advent of such schemes, the most popular (and still a widely-deployed solution) was the integration of blacklist-based anti-phishing techniques into browsers. Ludl et al. [3] tested the effectiveness of the blacklists maintained by Google and Microsoft to understand the viability of this approach, and found that blacklist-based solutions are effective and useful components in the fight against phishing. On the other hand, it has also been shown that blacklists are ineffective for protecting users from phishing attacks *initially* and that their effectiveness increases with time [4].

Phishing Detection Over Web page Content. A typical approach to detect phishing using web page content is analyzing the structure of the URLs and validating the authenticity of the content of these target web pages. Cantina [5,6] is one such scheme: a content-based approach to detecting phishing websites based on information retrieval and text mining algorithms. A research team from Google has presented a machine learning technique to accomplish a large scale automatic classification of phishing web pages [7] by analyzing both the URL and the content of the page and achieves 90% accuracy in classifying web pages.

Phishing Detection Using URL analysis. [8] and [9] proposed schemes that identify phishing URLs by analyzing only the structure of the links and not the content of the target web pages. In [8], the authors describe several features that can be used to distinguish a phishing URL from that of a benign URL. They

use these features to model a logistic regression filter and show that it has high accuracy in detecting phishing emails. The algorithm of [9], LinkGuard, uses the phishing data provided by the APWG to extract generic characteristics of hyperlinks embedded in phishing emails. It successfully detected 195 out of the 203 phishing attacks.

Phishing Detection Over Email Content. Most phishing detection schemes that operate at the email level use machine learning techniques on a feature set designed to highlight user-targeted deception in electronic communication [10,11,12,13,14,15]. A statistical classifier is trained on a set of features extracted from the email content and structure over the training data. After the training, this classifier is used to detect phishing emails from the email stream. These detection schemes differ both in the number and type of features used in the training process. These statistical filters can either be installed on the server or the client side. One of the important maintenance aspects of a machine learning phishing detection scheme is that these filters need to be updated on a regular basis. [16] presents a comparison of machine learning techniques for phishing detection. PhishCatch is a heuristic algorithm (not based on machine learning) proposed by [17] which performs header, link and a cursory text analysis (scanning for the presence of certain text filters) of incoming emails. In [2], the authors study the evolution of phishing email messages and develop a classification of phishing messages into two groups: *flash* and *non-flash* attacks, and classify phishing features into *transitory* and *pervasive*. For more details on phishing and detection schemes, the reader is encouraged to refer the books by [18,19] and [20].

3 Definitions and Tools

3.1 TF-IDF

In information retrieval, TF-IDF (Term Frequency-Inverse Document Frequency) is a weight used to determine the importance of a word to a document in a collection of documents. The importance of a word increases proportionally to the number of times a word appears in the document (term frequency) and is inversely proportional to the document frequency of the word in the collection. The IDF is a measure of the discriminating power of the term. It measures how common a term is across an entire collection of documents. Thus, a term has a high TF-IDF weight by having a high term frequency in a given document and a low document frequency in the whole collection of documents. For more details about TF-IDF, refer to the book by [21].

3.2 Natural Language Preliminaries

Despite the difficulty of natural language processing on computers, due to the inherent ambiguity and rich structure of natural languages, our approach to email text processing employs the following NLP techniques: lexical analysis,

part-of-speech tagging, named entity recognition, normalization of words to lower case, stemming and stopword removal. The goal of lexical analysis is to split the email into sentences and each sentence into words. The part-of-speech tagging phase tags each word with its part-of-speech, viz., noun, verb, etc. Named entity recognition tags the named entities in the email, which are nouns that name either person, location or organization. Words are converted to lower case in a normalization phase. The goal of stemming is to reduce each word form to its root or stem. For example, the verb *acting* is reduced to *act*. A popular program for stemming is the Porter Stemmer [22]. The aim of stopword removal is to remove common words such as *it, a, an, the,* etc. For this purpose a stopword list is used. We also use semantic NLP techniques, viz., word-sense disambiguation and WordNet, as opposed to purely syntactic or statistical ones based on feature counting. The *sense* or meaning of a word depends on its context. For instance the word "plant" could mean a factory in one context and could mean a tree in another context. The goal of word-sense disambiguation is to find the appropriate sense of a word based on the context.

3.3 WordNet

According to Fellbaum [23], WordNet combines features of both a dictionary and a thesaurus. The building block in WordNet is a *synset* (a set of synonyms), which consists of all the words that express a given concept, and the basic semantic relation in WordNet is synonymy. The semantic relation that is the most important in *organizing* nouns into a hierarchy is the hyponymy relation between synsets. Hyponymy is the relation of subordination (or class inclusion or subsumption). For example, the word "poodle" is a hyponym of the word "dog" since a poodle is a kind of dog, and "dog" is the hypernym of "poodle." Miller writes ([23] page 26): "Since a noun usually has a single hypernym, lexicographers include it in the definition." The key point to be noted is that although the hypernymy relation is defined on synsets in WordNet, and hence it could happen that a synset can have more than one hypernym, this situation is not frequent for nouns[1]. However, for verbs the situation is quite different and the hyponymy structure is not even acyclic [24]. The relation between verbs to other verbs is used by PhishNet-NLP.

We use the hyponymy relation between verbs, which is defined as follows: A is a *hypernym* of B if the meaning of A encompasses the meaning of B (B is called the *hyponym*). All nouns in WordNet are stored in a graph (that is close to a tree) that represents the hypernymy hierarchy. The word *entity* is the root of the tree, because it is believed to encompass the meaning of all other nouns. Traversing down the tree manifests more specific nouns as shown in Figure 1 of a small portion of the hypernymy tree. All verbs in WordNet are arranged in a hypernymy graph as well, but for verbs this graph is "forest-like" but not a forest due to the presence of cycles.

[1] We do take care of the situation in which there are multiple hypernyms as explained in the Text Analysis subsection 4.1.

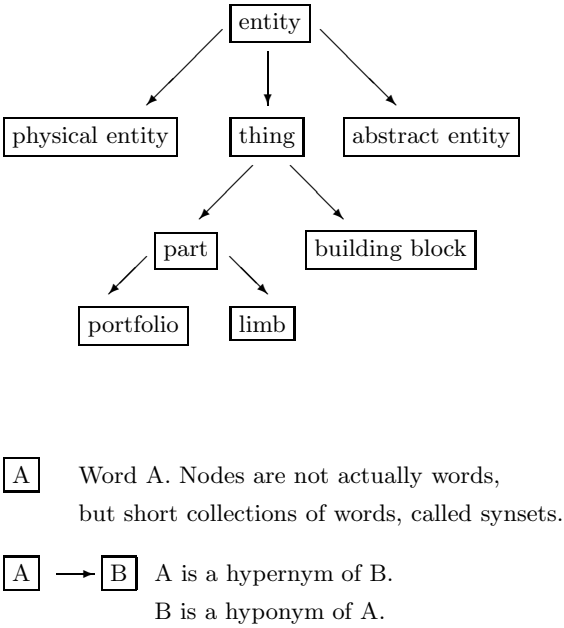

Fig. 1. A tiny WordNet hypernymy tree

We need to invoke our word sense disambiguation software before we can call the WordNet program. The reason is that a synset is designed to refer to a single concept and hence we need to disambiguate words in the document to find the correct synset for a noun. As mentioned above, the word "plant" could mean a factory in one context and could mean a tree in another context. Hence the word plant would be found in two different synsets in this case.

4 Phishing Detection Algorithm: PhishNet-NLP

PhishNet-NLP is a comprehensive scheme that makes use of all the information present in an email, except attachments, to ascertain which class it belongs to: phishing or legitimate. The first step in the protocol is parsing: PhishNet-NLP accepts an incoming email from the MTA and proceeds to parse it into its constituent components: header, links and text. If the email is HTML encoded, as indicated by the header, we further decode the HTML email body to plain text to perform further analysis. Having obtained the header, links and text, we proceed to analyze each component through their respective classifiers as discussed below. PhishNet-NLP then proceeds to perform majority voting on the scores obtained from the header, link and text analysis classifiers to determine whether an email is legitimate or phish. The reason for using majority voting as opposed to considering certain weight factors for each of the individual classifiers

```
   Input: SMTP server name, user name, password
   Output: Label for each email: Phishing or Legitimate
 1 Fetch email from SMTP server
 2 if (new email downloaded) then
 3 |  foreach email e do
 4 |  |  header h = extractHeader();
 5 |  |  if (h indicates that e is HTML encoded) then
 6 |  |  |  decodedEmail dE=HTMLDecode(e);
 7 |  |  end
 8 |  |  parsedEmail pE = emailParser(dE);
 9 |  |  headerScore = headerAnalysis(header);
10 |  |  linkScore = linkAnalysis(links);
11 |  |  textScore = textAnalysis(text);
12 |  |  cs = combineScore(headerScore, linkScore, textScore);
13 |  |  if cs ≥ 2 then
14 |  |  |  Output Label: Phishing
15 |  |  end
16 |  |  else
17 |  |  |  Output Label: Legitimate
18 |  |  end
19 |  end
20 end
```

Algorithm 1. PhishNet-NLP: Phishing Detection Algorithm

is to assign an equal importance to each of the classifiers. The first author has proved that under the assumption of independence, the majority voting approach has better coverage (accuracy) than that of each individual classifier whenever each classifier in the combination has better than a 50% coverage (accuracy). Majority voting also avoids the following two vexing problems: (i) how to compute optimal weights, which requires a training corpus, and (ii) the optimal weight combination is likely to be different for different corpus and users. Algorithm 1 shows an outline of PhishNet-NLP. We begin our discussion of PhishNet-NLP with our novel text analysis classifier and then discuss the header and link analysis classifiers respectively.

4.1 Text Analysis

The goal of email text analysis is to classify the email into two classes: informational and actionable. This is done by analyzing the email text and giving a score to the email called Textscore. The overall approach of PhishNet-NLP is designed for maximum flexibility and efficiency. When the "context" information of an email is available, PhishNet-NLP will use the context to generate a score called Contextscore for the email as well. The context of an email is defined to be the other saved emails of the user, this includes both sent and received emails. For efficiency purposes, the user is given full control over PhishNet-NLP's context analysis option: whether or not to use context analysis, the context size to

use for context analysis, and the date at which the context starts. Context size could be specified in two ways: number of emails or a date range. When the context option is used, then the two scores - the Contextscore and the Textscore are combined logically.[2]

To generate the Textscore of the email, we employ a semantics-based method since phishing emails are typically short and designed to hide their sinister purpose and appear innocuous to the user. Hence, applying syntactic techniques, such as sentence position, or purely statistical approaches, such as word frequencies, to email text analysis are likely to prove suboptimal.

Our semantic approach to email text processing employs the following NLP techniques: lexical analysis, part-of-speech (POS) tagging, named entity recognition, normalization of words to lower case, stemming and stopword removal. Stopword removal will include removal of common suffixes such as Jr., Sr., II, etc., after names (named entities) and prefixes such as titles (Dr., Prof., Mr., Ms., etc.). The novelty of PhishNet-NLP consists in deeper word analysis by extracting important words from the email text, tagging them with their senses based on the surrounding contexts of the words, and using these to query WordNet. These distinguished words are called *keywords*. The sense of the word is used in locating the word in the WordNet hypernymy tree and to generate a score for the word as described below. We employ SenseLearner [25] for word sense disambiguation, and TextRank keyword extraction for identifying the important words of the email text [26]. SenseLearner was trained using the SemCor 2.1 database, which was compiled using WordNet 2.1.

For a user u, let Basic-Names(u) denote the lower-case versions of u's last name, first name, middle name(s) if any, and their common spelling variants. This set can be initialized by the user. Let $Names(u)$ denote all permutations of words from Basic-Names(u) taken two at a time, three at a time, and so on until |Basic-Names(u)| at a time. For an email text, e, let Named-entity(e), denote the set of named entities in e ignoring only the greeting part of the email, which can be identified easily as a sentence fragment using parsing or heuristics such as missing verb and presence of named-entity from $Names(u)$. If |Named-entity(e)−Names(u)| = 0, then email e receives a Textscore of 0 (a score of 1 represents phishing and 0 stands for legitimate). The reason is that a phishing email is very likely to mention at least one institution in the body of the email. Now, assume that |Named-entity(e)−Names(u)| ≥ 1. Since we are interested in determining the extent to which an email is actionable, we score certain verbs in the body of the email. If the email contains no text, we mark it as phishing, since this means the email has either links or attachments only and legitimate email senders usually write a few words to explain the links or attachments that they are sending out.

[2] The reason for these options is that the user may wish to restrict context analysis when they have a large mail box with lots of emails unprocessed by our context analysis routine, since for a large unprocessed mail box, context analysis will take more time, which the user may not have. In such a case the user can be warned that the context analysis is using limited information and could be less precise.

Let $V = \{$*click, follow, visit, go, update, apply, submit, confirm, cancel, dispute, enroll*$\}$. To each word in the set V, the appropriate verb sense (denoted by #v at the end of the word in WordNet) is attached. For any set X containing words along with a sense for each word, let $Synset(X) = \{synset(x) \mid x \in X\}$, where $synset(x)$ is the WordNet synset of x for the specified sense. For natural number $i \geq 1$, let $Hypo^i(Synset(V))$ denote the union of all the synsets reached by following up to i hyponymy links from the synsets in $Synset(V)$. We let $SV = Hypo^4(Synset(V))$ be the set of special verbs. Note that the WordNet verb hierarchy is not a tree structure and is not even acyclic [24], which means that following hyponymy links must be done together with cycle detection. Let $SA = Synset($*{here, there, herein, therein, hereto, thereto, hither, thither, hitherto, thitherto}*$)$ with each word in this set SA having the adverb sense, $U = \{$*now, nowadays, present, today, instantly, straightaway, straight, directly, once, forthwith, urgently, desperately, immediately, within, inside, soon, shortly, presently, before, ahead, front*$\}$ (words conveying a sense of urgency), and $D = \{$*above, below, under, lower, upper, in, on, into, between, besides, succeeding, trailing, beginning, end, this, that, right, left, east, north, west, south*$\}$, the set of direction words. These words were chosen mainly based on the authors' experience with the phishing emails that they have received in the past, and a scan of about 20 (0.4%) emails in the phishing database.

To motivate the above definitions, consider a phishing email in which the bad link appears in the top right-hand corner of the email and the email (among other things) directs the reader to "click the link above."

The score of verb $v \in SV$,

$$score(v) = \{1 + x(l + a)\}/2^L.$$

The parameter $x = 1$ if the sentence containing v also contains either a word from $SA \cup D$, and, either a link or the word "url," "link," or "links" appears in the same sentence; otherwise, $x = 0$. The parameter $l = 2$ if the email has two or more links, $l = 1$ if the email has one link, and $l = 0$ if there are no links in the email. The parameter $a = 1$ if there is a word from U or a mention of *money* in the sentence containing v, otherwise $a = 0$. We include money since phishers often lure targets by promising them a sum of money if they complete a survey, or by stating that someone tried to withdraw a sum of money from the user's bank account recently, etc. The parameter L is the level of the verb, where *level* of a verb in SV is one more than the least number of hyponymy links followed to reach the verb from a synset in $Synset(V)$.

The reason for weighting the link score of the email (l) and the urgency or incentive score (a) of the sentence with a directive to take action (x) with respect to a link is to reduce the false positives for emails that acknowledge some previous action of the user, or for emails received by user A that are replies to emails sent by A and contain a link in either A's signature included in the reply or in the signature of the sender of the reply. For example, when someone submits a proposal or report to FastLane, an automatic acknowledgment is sent by the website and it usually includes a link. We are aware of several instances in which

emails contain links in the signature fields. The reason for the exponential decay with L is the diversity of verbs and the proliferation of their different senses at greater distances from SV, which leads to an increase in the imprecision of word-sense disambiguation. Even without this complexity, word-sense disambiguation is a challenging problem due to the ambiguity inherent in natural languages. The Textscore of an email e is given by $Textscore(e) = Max\{score(v) \mid v \in e\}$. We also experimented with the average score of the scores of all verbs with a nonzero score, but this function gave inferior results in our experiments.

Contextscore. For the Contextscore, we treat the email as a vector of TF-IDF [21] values *in the semantics space as opposed to traditional syntactic techniques* after stopword elimination and stemming. Note that the TF-IDF scheme converts a vector of words to a vector of real values using the product of term frequency and inverse document (for our purposes this is the email) frequency as mentioned above in Section 3. WordNet is again employed for this purpose after POS tagging and word sense disambiguation. Words belonging to the same synset are represented by a common word in the vector. For instance, different forms of the same verb "is", "was" , etc. are represented by the common verb "to be" and also different verbs with the same sense and meaning such as "is" and "exists", etc., are also represented by the verb "to be." Then, we perform similarity computation between the email vector ev and the corresponding vector for each email in the context, say ec. For the similarity computation we adopt the cosine measure, $Similarity(ev, ec) = cosine\ \theta$, where θ is the angle between the two vectors. The smaller the θ, the bigger is the similarity between two emails. Finally, $Contextscore(ev) = max_{ec \in C} Similarity(ev, ec)$. We also compute the size of the intersection |Named-entity(ev) $\cap$ Named-entity(ec)| for each email ec with similarity of over high-threshold and if this intersection is null, then we lower the $Contextscore$ down to 0. If Contextscore is below low-threshold it is rounded down to 0. If it is above high-threshold and the size of the intersection is at least one, then it is rounded up to 1. Low-threshold and high-threshold are initially set to 0.5 (an angle of 60 degrees or higher) and $\sqrt{3}/2$ (an angle of 30 degrees or lower) respectively and can be fine-tuned further, if necessary, based on experiments. No rounding is performed if Contextscore is between low-threshold and high-threshold. For efficiency purposes, PhishNet-NLP saves the vocabulary and named-entity information for the context examined, and the corresponding vectors for the emails examined in a database for subsequent reuse. Multiple indices can be constructed on this information for efficient retrieval based on the context options provided in PhishNet-NLP.

4.2 Combining Textscore and Contextscore

The combination of Textscore(e) and Contextscore(e) is done logically to yield Final-text-score(e). It does not make sense to combine them algebraically. If no context information is available, Final-text-score(e) = 1 if Textscore(e) ≥ 1 and 0 otherwise. When context information is available, we proceed as follows.

If Contextscore(e) = 1 and any one of the emails that yield the maximum similarity score is marked as dangerous by the user, the Final-text-score(e) = 1. If Contextscore(e) = 1 and all of the emails that yield the maximum similarity score are marked safe by the user, then Final-text-score(e) = 0. If Contextscore(e) = 0, then the email is not very similar to any email in the context. In this case, Final-text-score(e) = 0 if Textscore(e) < 1 and Final-text-score(e) = 1 otherwise. If low-threshold < Contextscore(e) < high-threshold, then the email has moderate similarity to some email in the context. In this case, if Textscore(e) < 1, then Final-text-score(e) = 0, else Final-text-score(e) = 1.

If user input is an acceptable response, then the user could be queried to determine whether the email has arisen from some past action of the user. This would be useful in two "gray" areas: Contextscore is between low and high threshold and Textscore is less than 0.5 and Contextscore is zero and Textscore is between 0.5 and 1. If $0.5 \leq$ Textscore(e) < 1, the user could be prompted to determine if the email has arisen from some past action of the user. If yes, Final-text-score(e) = 0, otherwise Final-text-score(e) = 1. In our experiments, we simplify the logical combination: rounding down the context score to 0 if it is between 0 to 0.866 (angle greater than 30°) and rounding up to 1 otherwise. These thresholds were not finetuned using the data.

To maintain user's privacy, context analysis can be a separate application that works under user control without downloading user emails into its space.

4.3 Header Analysis

Our header analysis classifier is significantly more advanced than the classifier presented by PhishCatch [17] in several aspects: (i) we deal with email forwarding issues, (ii) we make use of DKIM and SPF information whenever it is available, and (iii) we account for the differences in the headers based on whether the email is sent from a mobile device or relayed by multiple servers in the user's domain. In this classifier, we perform analysis on the data from the extracted headers to determine whether the email is phish. First, the user is asked to input his/her other email addresses that forward emails to this current email address and this information is stored. We assume that these forwarding email accounts and the Local Host also have PhishNet-NLP installed. An in-depth discussion of DKIM/SDID is beyond the scope of this paper and the interested reader is referred to RFC 5585 [27] for an overview of the DKIM service and SDID and to the IETF publication RFC 4408 [28] for more information on SPF. In [29], we present a more detailed treatment of headerAnalysis() and present an interesting discussion on the significance of DKIM signatures and SPF through examples.

Phase 1 - Extracting the data:
We extract the FROM and DELIVERED-TO fields from the header. Then, we extract the RECEIVED FROM field(s) as follows. We look at the received from fields in order, starting with the first such field and then the next such field if present and so on.

- If the Received From section of the email contains a DKIM signature, we store the Signing Domain Identifier [SDID].
- Otherwise, if there is a Received-SPF field below a Received From field, then first we store the Received From field. Additionally, if the SPF query returns "pass," and if the domain in the From Field accepts an IP address as a permitted sender in the Received-SPF field, we perform an NSLOOKUP on this IP address, and store the domain name corresponding to this IP address in the variable SPFQuery.
- Otherwise, we store the RECEIVED FROM field.

Phase 2 - Verifying the data:

- If the first Received From field has the same domain name as the FROM FIELD or LOCALHOST or ANY FORWARDING EMAIL ACCOUNT, or if the NSLOOKUP on the IP address of the permitted sender in the Received-SPF field yields the same domain name stored in the variable SPFQuery, then this email is legitimate.
- Otherwise, if the first Received From field has the same domain name as the user's current email account's domain name, then we look at the next received from field. The justification for this is provided in the security analysis of our scheme.
- Otherwise, we mark the email as phishing.

4.4 Link Analysis

In this classifier, our objective is to determine whether the URLs present in the email point to the legitimate website that the text in the body of the email claims. We extract all domains from the links in the email in an array (let this array be called DOMAINS). The linkAnalysis() classifier assigns an email a score of 1 for phishing and 0 for legitimate as follows:

- If the length of DOMAINS is 0 (no links), the email is legitimate.
- If the email has more than 10 distinct words, we calculate the top four terms in the email using the TF-IDF scores. The IDF value of a word can be obtained by either doing a Google search for the word and obtaining the number of web pages in which it appears, or by using a standard NLP corpus. If the Google search approach is adopted, then the search information, together with the total number of web pages in Google's database, can be used to calculate the IDF value for each word. However, we note that Google returns only a somewhat loose upper bound on the number of web pages containing the word for efficiency purposes, which is progressively refined as the user examines the search results list. For this reason and the fact that Google discourages frequent automated searching (see the Implementation details Section 5.1), we used the email database itself to estimate the IDF value. We Google search each domain together with the top four terms.
- Otherwise, if the total number of distinct words in the email is less than 10, then we Google search each domain. If all domains appear in the top 30

results returned by the Google search, then we mark the email as legitimate, otherwise phishing. The reason for insisting on 10 words as a threshold is the very small likelihood of obtaining at least four content words in a text fragment that is shorter. Cantina's [5] experiments with varying result sizes of Google search justify our choice.

Combining Scores of the Three Classifiers: Recall that a score of 1 represents phishing and 0 stands for legitimate. If the combined score of the three classifiers (header, link and text) is ≥ 2, PhishNet-NLP labels the email phishing, otherwise it labels it legitimate.

5 Analysis and Results

In this section, we present an overview of our results. On a database of 2000 phishing emails (using the same phishing corpus as PhishCatch [30]), the percentage of emails that are marked by PhishNet-NLP as phishing is over 98% compared to PhishCatch's result in low 80%. On 1000 legitimate emails, PhishNet-NLP marked 99.3% of the emails as legitimate compared to 99% for PhishCatch [17]. However, note that the databases are different in this case since the authors of PhishCatch do not mention how they collected their legitimate emails. In this sense, we were able to increase coverage by about 18% for the phishing emails while obtaining higher accuracy. Furthermore, our header analysis classifier is more advanced than PhishCatch [17] in the sense that we also deal with email forwarding issues and also account for the differences in the headers based on whether the email is sent from a mobile device or relayed by multiple servers in the user's domain. Our header analysis goes beyond that of PhishCatch and examines DKIM (DomainKeys Identified Mail) signatures and SPF (Sender Policy Framework) fields when available. Although the phishing corpus emails were collected five to eight years ago, we still feel it is a good test since phishing sites are so short-lived [6] that the link analysis results should not change significantly when run on more recent phishing emails.

Cantina's experiments [5] were on the detection of masqueraded web pages rather than on phishing emails, and they experimented with only 100 websites. Still, they have a much higher false positive rate for legitimate web pages and lower coverage of masqueraded sites. Moreover, their algorithms exhibit a trade-off between coverage and accuracy. In contrast, our first run coverage (without context information) is never lower than 97.7% for the largest 4550 phishing database and simultaneously achieves high accuracy with high coverage.

[10] apply machine learning techniques on a set of 860 phishing emails, and 6950 non-phishing emails, and are able to correctly identify 92% of the phishing emails with 0.1% false positive rate. Using structural properties of emails, [11] were able to detect 95% of phishing emails but did not explicitly state their false positive percentages. Finally, it is important to note that the above mentioned machine learning approaches require a training corpus of emails whereas our approach does not.

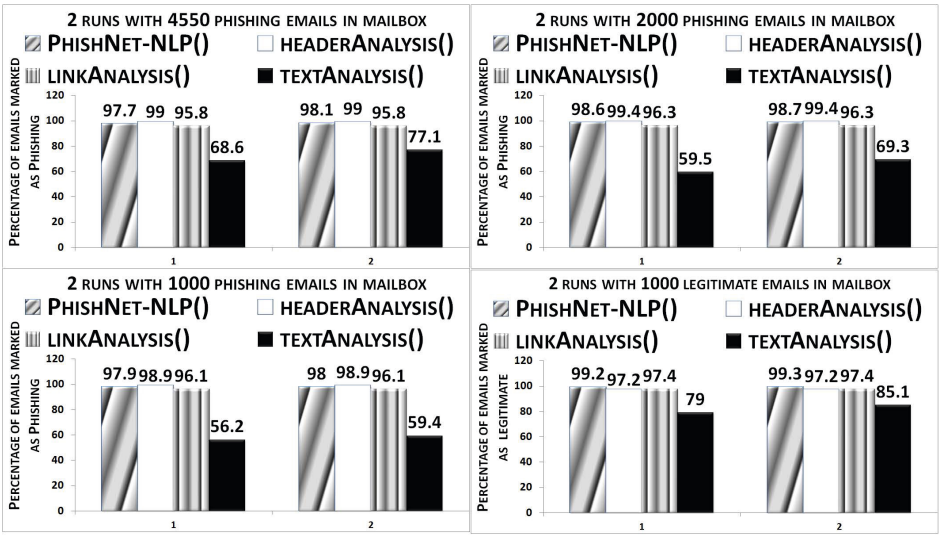

Fig. 2. Results

As explained at the beginning of Section IV, our results show that all three classifiers satisfy the minimum threshold needed for helping to improve the combined classifier since they are all above 50% in coverage and accuracy. However, there is some dependence between the text analysis and link analysis classifiers since one analyzes links and the other uses the presence of links in its scoring. We carefully considered whether to remove even this dependence, but decided against it since links are central to phishing via emails and since text analysis only considers the presence or absence of links and not on analyzing them.

The relatively lower percentage of phishing emails detected by textAnalysis() in the two big mail boxes is explained by the imprecision of NLP tools and the three types of emails: foreign language, emails with unusable text, and emails with tables and pictures and insufficient text that we encountered. Also, in each individual mailbox, the 2nd run produced an increased phishing detection by the textAnalysis() classifier and a small increase in the overall phishing detection. This is a direct consequence of the effect of the Context Score, which was not available in the first runs, but available in the 2nd runs after the first runs assigned scores to each email in the database. We could have obtained a higher detection rate on the first run of textAnalysis() by using the previous context of the first N emails when processing email $N + 1$. However, we preferred to keep a fixed context for analysis of each email rather than a growing context, since in this case our results are insensitive to the order in which emails are processed.

5.1 Implementation Details

We implemented PhishNet-NLP using Perl v5.12.4, WordNet version 2.1 and SenseLearner 2.0. We used the Stanford POS tagger 2006-05-21 and Stanford

Named Entity Recognizer 1.0. Our implementation platform was a core 2 Duo 2.66 GHz processor, 4 GB RAM machine running 32 bit Windows 7. We used Cygwin for the POS tagger, NER, SenseLearner and WordNet. Some of the challenges we faced during implementation were: 1) The Google Search API would not allow us to perform frequent automated searches. We were forced to use a random delay of 10 to 20 seconds after every search to circumvent this issue. 2) Parsing an email into the constituent header and body and then extracting the text and links from it was challenging since most emails are HTML encoded and the headers do not always end with the same line format.

Our method of extracting data from emails relies on the use of regular expressions. From analyzing thousands of emails, we observed that the message headers were formatted differently among them. So we had to study a large number of email formats to design the decoder (which decodes html if present, extracts info from the header and body and removes any attachments). To summarize, our decoder is reliable, but not 100% efficient in extracting the maximum possible data from all the emails. This is an area of improvement that we are working on at this moment. If an attachment is present in an email, then the last portion of the message header contains one of the following: `Content-Disposition: attachment` or `Content-Disposition: inline`. This is followed by the encoded attachment file. We used this information to ignore all attachments. Given that we had to employ a random sleep time between subsequent Google searches, in our future work, we would like to make use of different search engines for consecutive searches to eliminate this problem and possibly obtain better results.

5.2 Security Analysis and Discussion

We now analyze the security of our scheme against several scenarios and discuss some interesting aspects of our approach.

Is textAnalysis() or linkAnalysis() Redundant? Observe that while the headerAnalysis() classifier alone shows very high coverage and high accuracy, the importance of link and text analysis stems from the fact that a sophisticated phisher can manipulate the originating "Received From", "From" and the "Delivered To" information completely (e.g., see Chapter 3 of [18]). To this end, link and text analysis are very important and provide robustness to our scheme. The case of insider attacker discussed below further justifies their inclusion.

Attacks Based on Knowledge of Our Scheme. The reader might think that a phisher can analyze how our detection algorithm works and then design a phishing email to fool PhishNet-NLP. But our results from the LinkAnalysis show that it is very difficult to create a fraudulent link to bypass LinkAnalysis. Moreover, unless the phishers have hacked into the mail server or the user's account, they would not have access to the context of the user's mailbox. Hence, it is likely that Context Analysis will also play a part in detecting such an email.

Insider Attacker. When someone hacks into an account in some domain and uses a friend list to attack any user in the same domain, headerAnalysis() will

fail to detect this. But even in such a case, PhishNet-NLP can use the linkAnalysis() and textAnalysis() to mark the email as phishing since the intent of the email is still to steal sensitive information by asking the user to click on a link for a malicious website. This even works for the scenario when user A's account is hacked and user A receives a phishing email, for example, if A's sensitive information is stored in an encrypted form.

Is textAnalysis() Flawed? Observe that as of this implementation, our textAnalysis() classifier will score the following email as phishing: "I found this video to be funny! Click on this link <legitimate link here>". This email will be scored as phishing even when coming from a genuine sender and a legitimate link. We would like to clarify that this is not a limitation of our approach - this is actually a design feature of PhishNet-NLP. The reason is that both header and link analysis will have a high likelihood of returning a score of 0 (indicating legitimate) on such emails and therefore, the majority vote will be legitimate. We also argue that while it may seem counterintuitive, such emails MUST be scored as phishing, since otherwise a sophisticated phisher who could fool headerAnalysis() would escape detection as the majority vote would be legitimate if the header and text score say legitimate and only linkAnalysis() indicates phishing.

Foreign Language Email or Emails with Insufficient Text. As of the present design, emails in foreign languages or emails with insufficient text (only links or attachments) present a challenge to the textAnalysis() classifier which leads to a low phishing detection rate by the textAnalysis() classifier. However, we were able to offset this to a certain extent by using context analysis to correctly identify the email as phishing.

Efficiency Considerations. For efficiency, PhishNet-NLP is designed to first execute headerAnalysis() and linkAnalysis() on the email that is being analyzed. If the sum of the scores of these two classifiers is equal to 1, only then will PhishNet-NLP execute textAnalysis() (because if the combined score is either 0 or 2 from the first two classifiers, then the score from textAnalysis() cannot change the final output label of PhishNet-NLP). But we disabled this feature during our testing phase to obtain the results from each classifier.

Justification of Strategy for Examining Received from Fields. As DKIM becomes widely deployed, sending domains will develop reputations as sources of spam or useful messages. We believe that senders are not able to create covert sub-domains under their main domain (unless an authorized insider attacker is involved which we believe may be unlikely) and cannot manipulate the "Received From" fields of legal intermediate MTAs. We note that it is not very easy to identify whether a "Received From" field is from a genuine intermediate MTA or just added by the phisher to confuse the header analysis. The highest probability for a "Received From" field of truly originating from a genuine intermediate MTA is the one closest to the recipient's domain, justifying our use of the closest MTA in our scheme.

6 Conclusion

In this paper, we presented a phishing detection scheme called PhishNet-NLP. To the best of our knowledge, this is the first scheme to utilize natural language based techniques and context information when available to detect phishing. PhishNet-NLP operates by inferring the "intention" of the email - whether it is informational or actionable. Our phishing detection rate is at least 97% with very low false positives. Another novel feature in PhishNet-NLP is that we utilize all of the information available in an email, namely, the header, links and text of an email. Our scheme operates in the default mode and does phishing detection in the absence of any history. The novelty lies in the fact that when prior history is available, our scheme takes advantage and improves the detection capability. Finally, our scheme is designed to detect phishing at the email level rather than to detect fraudulent, masqueraded websites thereby protecting the user from the start. As future work, we plan to implement PhishNet-NLP to permit the user to be interactive enabling us to understand if a particular email resulted by an action of the user. Processing attachments in emails is also an interesting direction for the future. We have reduced our reliance on Google for link analysis [29].

Acknowledgments. Research of the first author was partially supported by NSF grants DUE 0737404 and CNS 1062954.

References

1. Parno, B., Kuo, C., Perrig, A.: Phoolproof Phishing Prevention. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 1–19. Springer, Heidelberg (2006)
2. Irani, D., Webb, S., Giffin, J., Pu, C.: Evolutionary study of phishing. In: 3rd Anti-Phishing Working Group eCrime Researchers Summit (2008)
3. Ludl, C., McAllister, S., Kirda, E., Kruegel, C.: On the Effectiveness of Techniques to Detect Phishing Sites. In: Hämmerli, B.M., Sommer, R. (eds.) DIMVA 2007. LNCS, vol. 4579, pp. 20–39. Springer, Heidelberg (2007)
4. Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., Zhang, C.: An empirical analysis of phishing blacklists. In: Proc. 6th Conf. on Email and Anti-Spam (2009)
5. Zhang, Y., Hong, J., Cranor, L.: Cantina: a content-based approach to detecting phishing web sites. In: Proc. 16th Int'l Conf. on World Wide Web, pp. 639–648. ACM (2007)
6. Xiang, G., Hong, J., Rose, C.P., Cranor, L.: Cantina+: A feature-rich machine learning framework for detecting phishing web sites. CM Trans. Inf. Syst. Secur. 14, 21:1–21:28 (2011)
7. Whittaker, C., Ryner, B., Nazif, M.: Large-scale automatic classification of phishing pages. In: Proc. of 17th NDSS (2010)
8. Garera, S., Provos, N., Chew, M., Rubin, A.: A framework for detection and measurement of phishing attacks. In: Proc. 2007 ACM Workshop on Recurring Malcode, pp. 1–8 (2007)

9. Chen, J., Guo, C.: Online detection and prevention of phishing attacks. In: First Int'l Conf. on Communications and Networking in China, ChinaCom 2006, pp. 1–7. IEEE (2006)
10. Fette, I., Sadeh, N., Tomasic, A.: Learning to detect phishing emails. In: Proc. 16th Int'l Conf. on World Wide Web, pp. 649–656. ACM (2007)
11. Chandrasekaran, M., Narayanan, K., Upadhyaya, S.: Phishing email detection based on structural properties. In: NYS CyberSecurity Conf. (2006)
12. Bergholz, A., Chang, J., Paaß, G., Reichartz, F., Strobel, S.: Improved phishing detection using model-based features. In: Proc. Conf. on Email and Anti-Spam, CEAS (2008)
13. Basnet, R., Mukkamala, S., Sung, A.: Detection of phishing attacks: A machine learning approach. In: Soft Computing Applications in Industry, pp. 373–383 (2008)
14. Bergholz, A., Beer, J.D., Glahn, S., Moens, M.F., Paaß, G., Strobel, S.: New filtering approaches for phishing email. Journal of Computer Security 18(1), 7–35 (2010)
15. Gansterer, W.N., Pölz, D.: E-Mail Classification for Phishing Defense. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) ECIR 2009. LNCS, vol. 5478, pp. 449–460. Springer, Heidelberg (2009)
16. Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S.: A comparison of machine learning techniques for phishing detection. In: Proc. Anti-phishing Working Group's 2nd Annual eCrime Researchers Summit, pp. 60–69. ACM (2007)
17. Yu, W., Nargundkar, S., Tiruthani, N.: Phishcatch-a phishing detection tool. In: 33rd IEEE Int'l Computer Software and Applications Conf., pp. 451–456 (2009)
18. Jakobsson, M., Myers, S.: Phishing and countermeasures: understanding the increasing problem of electronic identity theft. Wiley-Interscience (2006)
19. James, L.: Phishing exposed. Syngress Publishing (2005)
20. Ollmann, G.: The phishing guide. Next Generation Security Software Ltd. (2004)
21. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc. (1986)
22. Porter, M.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
23. Fellbaum, C. (ed.): WordNet An Electronic Lexical Database. MIT Press (1998)
24. Richens, T.: Anomalies in the wordnet verb hierarchy. In: COLING, pp. 729–736 (2008)
25. Mihalcea, R., Csomai, A.: Senselearner: Word sense disambiguation for all words in unrestricted text. In: ACL (2005)
26. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: EMNLP, pp. 404–411 (2004)
27. Hansen, T., Crocker, D., Hallam-Baker, P.: Domainkeys identified mail (dkim) service overview (2009), `http://www.dkim.org/specs/rfc5585.html`
28. Wong, M., Schlitt, W.: Sender policy framework (spf) for authorizing use of domains in e-mail (2006), `http://tools.ietf.org/html/rfc4408`
29. Verma, R., Shashidhar, N., Hossain, N.: Two-pronged phish snagging. In: Seventh International Conference on Availability, Reliability and Security, Availability, Reliability and Security (ARES). IEEE (2012)
30. Nazario, J.: The online phishing corpus (2004), `http://monkey.org/~jose/wiki/doku.php`