Learning from the Ones that Got Away: Detecting New Forms of Phishing Attacks

Christopher N. Gutierrez†, Taegyu Kim‡, Raffaele Della Corte†, Jeffrey Avery†§, Saurabh Bagchi†, Dan Goldwasser†, Marcello Cinque‡

†Purdue University ‡Universitá degli Studi di Napoli Federico II §Northrop Grumman Corporation

Abstract—Phishing attacks continue to pose a major headache for defenders of computing systems, often forming the first step in a multi-stage attack. There have been great strides made in phishing detection, however, some insidious kinds of phishing messages appear to pass through filters by making seemingly simple structural and semantic changes to the messages. We tackle this problem in this paper, through the use of a machine learning classifier operating on a large corpus of phishing and legitimate messages. By understanding common phishing features, we design a system to extract features and elevate some to higher level features that are meant to defeat common phishing mail construction strategies. The algorithms are instantiated in a usable system called SAFE-PC (Semi-Automated Feature generation for Phish Classification). To evaluate SAFE-PC, we collect the large corpus of phishing messages from the central IT organization at a tier-1 research university. The execution of SAFE-PC on the dataset exposes some hitherto unknown insights about phishing campaigns directed at university users. SAFE-PC can detect more than 70% of the emails that had eluded our production deployment of Sophos, a state-of-the-art email filtering tool today. It also performs better than SpamAssassin, a commonly used email filter. We also develop an online version of SAFE-PC that can be incrementally retrained with new samples. Its detection performance is found to improve with time as new samples are fed in, while the time to retrain it stays constant.

1 INTRODUCTION

Reports from the Anti-Phishing Working Group (APWG) indicate that incidences of phishing attacks have been on an increasing trend and in 2016, the number of domain names used for phishing reached an all-time high [1].

Phishing continues to be a source of attacks despite significant progress in production-deployed phishing filters, which come standard in email clients and servers. The continued prevalence of phishing happens fundamentally because the defensive measures can be reverse engineered by the originators of phishing campaigns, and bypassed. In addition to this observation, many instantiations of phishing filters tend to be fragile to change, e.g., rigid regular expressions looking for specific patterns in the text, which can be bypassed by trivially modifying the text. Another challenge is the transient nature of the domains hosting phishing content, which makes it difficult for detection techniques that rely on periodic scans of URLs embedded in emails. In APWG’s 4th quarter of 2016 report [2], an increase in the use of URL redirection to render phishing websites exemplifies the challenges associated with detection using URLs and blacklists. This suggests that alternative measures and characteristics within emails must be identified and used to accurately detect phishing messages.

To efficiently handle the barrage of phishing emails, automated solutions have become quite adept at identifying and removing them from a user’s mailbox. Examples of popular products in this space are Webroot-RSA’s Real-Time Anti-Phishing Service, BrandProtect’s Anti-Phishing Email Analysis package, SpamAssassin, and the one deployed in our university, Sophos PureMessage. However, the challenge is identifying previously unseen phishing emails. A concrete example from our phishing dataset as seen in Table 2, is that an email about a Blackboard update for a new semester was caught in February 2014, but in May 2014, an email about a generic Blackboard update was not caught. These unforeseen phishing messages subvert filters that rely on known structures, blacklists of known IP addresses, or lexical analysis. The phishing filters can be easily subverted by reverse engineering a filter’s rules and adjust the phishing message by changing the structure of the message (to subvert regular expressions), use a previously unseen IP address (to avoid blacklists), or replace words within the phishing message (to throw off lexical analysis).

A fatal flaw concerning phishing detection is that they classify emails based on the surface level text, rather than looking for linguistic patterns and subterfuges that are common in deceptive communication. Such examples of subterfuge are the use of synonyms, different sentence construction, or the use of a different organization name from the same domain. One notable exception to this surface level defense is research that uses a variety of Natural Language Processing (NLP) techniques for phishing detection and is a foundation that our work builds upon [3].

In this paper, we present a system, called SAFE-PC, which improves the state-of-practice for detecting novel
phishing campaigns. First, it is customized to extract features from emails that we have found through long observation to recur in phishing campaigns, such as, the presence of keywords such as “account”, “suspend”, “expire”, etc, a commonly used method feature in phishing detection [3]. Second, it performs feature engineering to thwart phishing strategies, such as, deliberate mis-spellings, use of white space characters, and use of alternate characters that look similar to the original characters (e.g., use of “llnl.gov” in the URL instead of “llnl.gov”). These are mapped to a smaller feature set before classification. Third, it uses NLP techniques to create “higher level” features, such as, through Named Entity Recognition and Freebase (which maps both “Bank of America” and “Wells Fargo” to “financial institution” for example) and through synonym substitution. Finally, SAF-PC builds a classifier customized to handle the unbalanced nature of email datasets—there are likely to be many more legitimate emails than phishing emails. The classifier is an ensemble model that is comprised of multiple weak learners, which additionally does undersampling of the majority class (legitimate emails) to reduce the imbalance. Overall, SAF-PC is a framework that can help practitioners in protecting their systems against sophisticated phishing campaigns that are targeted to their specific context. Our design and evaluation are restricted to our university context. Since phishing emails carry with them significant context (such as, directed to students wanting to access their online grade books), we hypothesize that our system will have to go through a feature engineering process before being used in other contexts.

In this problem domain, it is essential to keep the classifier updated with new phishing emails that show up and are manually flagged on a regular basis, by mail administrators, security administrators, and end users. The phishing campaigns evolve and change, sometimes in concert with external events. Therefore, the obvious conclusion is that a batch mode classifier may not be suitable for the task. Thus, we evaluate an online variant of SAF-PC, which can be periodically and incrementally retrained as new samples become available. The online variant demonstrates the expected properties in that the detection performance improves gradually over time with minimal impact on training time.

We evaluate SAF-PC using two datasets of phishing messages collected from the central IT organization of a tier-1 research university. The first dataset comprises 37,606 email messages that the deployed email message filter software (Sophos PureMessage) did not catch. These messages were collected by a variety of ad-hoc mechanisms with a significant manual effort from the system administrators. The second dataset comprises 388,264 messages that Sophos did catch. Alongside both datasets, we use legitimate emails from some universities that we manually collect through from public newsgroups and publicly available financial emails, thereby keeping the domain and or subject of legitimate emails to be the same as the phishing ones. We train and test on non-overlapping time periods from the datasets, with a time gap between the two, thus evaluating the ability of SAF-PC to catch variants of old phishing campaigns. We also see strong temporal characteristics in the phishing campaigns, such as the end of the semester coinciding with a sharp increase in the number of phishing emails, plus a change in the characteristic with a significant fraction related to Blackboard, an online classroom management tool. We find that SAF-PC can detect 71% of the phishing emails that had been incorrectly classified by Sophos while having a false positive rate of 15%. We compared these results with SpamAssassin, an advanced machine learning based tool for detecting spurious email messages, spam, phishing, and others. SpamAssassin turns out to be non-competitive for phishing detection with a detection rate of less than 10%. We present an in-depth analysis of emails that were not detected, including SAF-PC, and identify strategies that phishers are utilizing to overcome these filters.

While there has been much work on spam detection and phishing detection, our work breaks new ground in the following ways:

1) We provide a method to extract features from free-form emails which we have found to be distinctive in phishing email detection. These features are of various kinds and at various levels of semantic hierarchy from specific to general and help reduce common subterfuges used in crafting phishing emails. We incorporate synonym analysis, Freebase, and Named Entity Recognition into our classification technique, an amalgamation that has not been presented before.

2) Prior NLP works for spam or phishing detection fall short in handling the real-world challenges that our actual data brings out — we discuss these in detail in Section 7. Specifically, our work can be thought of as an improvement of [3] in that we made the feature selection portable and based on empirical observation of evolving datasets.

3) We show that it is possible to apply online learning to the problem, thereby enabling a practical deployment in which new manually flagged emails can be brought into the system to improve it continuously without paying the cost of complete retraining on the entire corpus.

4) We evaluate our solution with real data sets of phishing campaigns at a large tier-1 research university’s central IT organization, comprising more than 400,000 phishing emails and 150,000 legitimate emails and show SAF-PC performs better than SpamAssassin. We bring out insights about the kinds of deceptive techniques that are difficult for automated techniques, including SAF-PC to detect.
2 CURRENT APPROACHES FOR PHISHING DETECTION

Current phishing defense mechanisms can be divided into three general categories: rules-based, classifiers and manual effort. These categories can be applied individually or in combinations. In practice, phishing detection solutions utilize a mix of rules-based algorithms to filter messages, compiled blacklists to detect known malicious senders, and classifiers to identify potential new phishing messages. We provide a brief overview of these categories below. A more comprehensive review is provided by Khonji et al. [4].

2.1 Rules Based

Filters use a combination of keywords, syntax, sender address, URLs, etc. to generate rules that detect phishing emails. These rules are developed based on prior phishing emails or domain knowledge. Rule-based tools are consistently updated as new phishing campaigns are identified.

Prior work explores using machine learning techniques to identify patterns of keywords in phishing emails [5]. This approach shows that keywords can be identified using automation. The main limitation of this approach is changing the order of keywords or using different keywords compared to prior campaigns would bypass this defense.

A simple method of rules based detection is blacklisting IP and email addresses. Compiled blacklists aggregate data from subscriptions of other blacklists and use the aggregated blacklist as a reference monitor. An analyst creates each blacklist, reviewing potential spam and phishing data from spam and phishing abuse accounts that are reported by human users. There is some automation in expanding the blacklists by tracking relations of confirmed blacklisted entities and the new suspect entity. However, this primarily manual process of finding potential phishing emails is taxing for administrators.

2.2 Classifiers

Classifiers detect phishing emails using statistics and pattern matching. Based on prior phishing emails used to train the classifier, new phishing emails can be detected that have previously been unseen. Prior work has looked at using machine learning to classify emails and websites as phishing or legitimate [6], [7]. [8].

2.3 Manual Effort

Detecting phishing emails using manual effort requires end users, including analysts, to identify phishing emails. This technique relies on the human ability to detect deceptive content within emails. Because humans are considered the weakest link in the security chain, this category is undesirable.

2.4 Combining Techniques

An example of combining techniques is employed by the tier-1 research university whose data we are using. The Sophos PureMessage filter [9], PureMessage utilizes a suite of methods to combat malicious emails — it does not make a distinction between phishing and spam emails. PureMessage employs a malware scanner to detect viruses, worms, and Trojan horses and it also utilizes blacklist to stop messages from untrustworthy IP addresses. PureMessage allows for IT administrators to define their own rules to block messages that are a particular threat to their systems. The university utilizes roughly 6,200 rules to combat malicious emails. These rules include a check against a blacklist of known phishing or spam IP addresses.

For the Sophos filtering approach, rules manually created by developers based on phishing emails seen in the past must constantly be updated as new attack techniques are discovered. The process requires a security expert (at Sophos) to analyze thousands of emails to identify new techniques, craft a signature for detecting these techniques within emails, and update the rule set. There is some automation through the use of regular expression generators [10] which, given multiple appropriately formatted strings (which must be extracted from the email messages) can come up with a regular expression.

The shortcoming of all these techniques in practice, whether used individually or in combination, is they have been implemented based on identifying specific features that are static and inflexible. This leads to phishing defenses that are bypassed by novel phishing campaigns with minimal differences compared to previously seen campaigns. Our approach, SAFE-PC, is an example of a combined technique, using some manual effort and a classifier, that adds greater flexibility to the feature set generation.

3 DESIGN

SAFE-PC is comprised of several semi-automated steps that process a corpus of phishing emails in preparation for training a classifier. The first step is to analyze the corpus of emails to create a rich set of features. The next step transforms each phishing email by projecting the raw email into a feature vector, thus seeking to remove noise that is inherent in any free-form media, like email. Next, a classifier is trained on all unique feature vectors. Finally, the classifier is used on production emails one-at-a-time to predict if a given email is legitimate or phish. SAFE-PC is capable of updating the classifier (without complete retraining) as new phishing emails are reported by users and collected by system administrators. Figure 1 shows the general workflow of SAFE-PC for the training phase.
The feature set consists of five kinds of features: (i) commonly known phishing words from domain knowledge and their synonyms, (ii) words associated with the tier-one research institution, (iii) commonly occurring words from our phishing corpus and their synonyms, (iv) proper noun organization names and their types, and (v) structural features in the email. The features are heavily based on prior work [3], [11] on NLP techniques for phishing detection. Our work demonstrates the effectiveness of these features on a much larger and more current data set presented in prior work.

SAFE-PC’s feature generation is design to capture deceptive semantics found in phishing emails. Certain words such as “urgent”, “update”, and “account” are commonly found in phishing campaigns [3]. These are obtained from postings on security mailing lists, security advisories from vendors, and prior papers on phishing detection. The set of common phishing words comprises feature set (i).

SAFE-PC determines the largest set of its features based on a word frequency analysis. English words are parsed and counted from each phishing email in the training set. Stop words, such as “they”, “the”, “I”, etc., are ignored. The selection criteria are words that appear in at least 1% of all emails within the training set. Feature set (ii) consists of words collected in the frequency analysis.

Feature set (iii) consists of words commonly associated with the tier-one research university. The words in this set include the name of the mascot, the name of the city where the university is located, the name of the IT department, the names of management software such as Blackboard, Banner, or Peoplesoft, and other university services and departments that manage resources.

Similar to the approached found in [3], for each word in the feature sets (i), (ii), (iii), all its synonyms, if any, are found via the Natural Language Toolkit’s synset functionality, which was configured to use the WordNet database [12]. The idea of using synonyms is motivated by the fact that phishing messages are often altered slightly to avoid signature-based detection by replacing certain words with synonyms. Further, we make the distinction if the word (or synonym of a given word) occurs in the body of the email or within the subject line.

Attackers commonly use trusted organizations to hide the deceptive nature of their communication, using the organization’s reputation to gain the recipient’s trust. We apply this knowledge by creating a set of features derived from organizations and entities found within phishing emails in our training set. We use the Stanford Named Entity Recognizer (NER) [13] to recognize different entity types, such as persons, locations, organizations, etc. within text. NER has found some use in phishing related work such as [3], [11]. Then, for a given entity, we query the FreeBase knowledge graph [14] to determine the domain that the entity is associated with. Freebase returns potential categories the entity belongs to and provides a score of how relevant each category is. We created “super-categories” based on phishing in the university context, by aggregating multiple categories from FreeBase. We consider three super-categories: bank/financial institution, university, and technology provider. Each super-category is a feature used in SAFE-PC’s classifier. This technique is particularly valuable in detecting phishing because a common technique is to create new emails by replacing one organization name with another from the same field, e.g., “Citibank” is replaced with “Chase Bank”. Feature set (iii) consists of one binary feature for each of the three super categories. It should be noted that super categories have been created manually by inspecting the categories returned by FreeBase. Similar categories have been grouped together in order to create the super categories, e.g., both “university” and “college” categories returned by FreeBase are represented by the “university” super category. That manual process can be easily transformed in an automatic process; however, it is beyond the scope of this paper. These features address a key limitation of current approaches. The rigid rules/pattern matching in current approaches could lead to inaccurate classifications. Super categories collect and amalgamate specific features into a broader feature representation. This provides greater resiliency to fluctuating deceptive language in phishing emails.

Feature set (iv) considers key structural components found within an email. These features were derived through observing that some phishing emails use special encoding or formatting. Specifically, the features consider the number of embedded links (e.g. href tags), the total number HTML tags within the body of the email, and whether or not the email uses images, URLs, UTF-8 character encoding, base64 encoding, or special charsets.

In total, we have 806 binary features derived from our training set. Despite the simplistic approach, our experimental evaluation shows that we are able to iden-
tify some previously undetected phishing emails. Several other approaches such as word stemming, sentence structure analysis, feature extraction from headers or links are not considered for simplicity sake or have been examined extensively in prior work. Further, these features and methods can be explored in future work, which may improve the results of our classifier. However, some features, such as those extracted from links or headers can be easily modified or forged, as previously discussed.

### 3.2 Email to Feature Space Projection

Based on the features found in the Feature Generation, we transform a given raw email into a vector of features. For our training set, we only consider unique feature vectors. In essence, similar phishing emails are projected onto the same point in our feature space, which has much lower dimensionality (806 features to be exact) than the raw emails. Moreover, one such mapped point contributes one sample to the training, while in actuality, representing many “similar” emails. Feature space projection addresses the lexical diversity strategy that phishers commonly employ to bypass filters. Thus emails with a similar cognitive meaning are mapped together in the feature space. It is capturing this notion of similarity among phishing emails that is at the crux of this problem.

### 3.3 Boosting-based Classification

We use an ensemble classifier based on the ML notion of boosting. The classifier that we use is called Random Under-Sampling Boost algorithm (RUSBoost) [15].

It is well suited to our problem where the two classes - phishing and legitimate - are severely imbalanced. Based on conversations with experts in phishing detection at the university, we can safely assume that the number of phishing emails received makes up 1-10% of all emails, depending on the virulence of phishing campaigns that may be present at that time. Thus, for our experimentation, we preserve such an imbalance between the legitimate and phishing data in the training set and use a 90%:10% ratio. RUSBoost handles the class imbalance problem by randomly undersampling the majority class given a certain balance to be achieved. This is preferable to another boosting-based classifier that handles class imbalance, called SMOTEBoost, which oversamples the minority class. This is because the oversampling increases the training time and the method to generate new samples is computationally expensive.

Any boosting-based algorithm uses a weak learner. We explore the use of Gaussian Naïve Bayesian (GNB), decision trees, and perceptions classifiers. At the end of training, the final verdict is a weighted vote from all the weak learners.

### 3.4 Online Learning

As new data is collected and observed, classification models must be updated. The initial model is adjusted, gradually, in response observations. Online learning applies to phishing detection. As new emails are received, some phishing emails fail to be detected by software and are manually discovered by users or system administrators. As these emails are collected, they are fed into SAFE-PC periodically (say, once at the end of a day). The goal is that the model improves without the need to retrain the classifier with the entire training set and new data. Further, we want to use an ensemble method (boosting in our case) keeping in mind the same imbalance between the two classes that we have referred to earlier, the minority phishing class and the majority benign class. Also, from an operational standpoint, it is unreasonable for any phishing detection system to be continuously re-trained, i.e., with each additional sample. Therefore, we take the more operationally feasible algorithm whereby the model is updated for a group of new samples.

After preliminary experimentation with the several boosting algorithms and different weak learners, we determined that online RUSBoosting with Gaussian naive Bayes (GNB) learners produce the best classification performance. The Python implementation of the online RUSBoost algorithm is driven from work presented in [16]. In updating the weak learner as we obtain a new set of labeled data, we keep the number of iterations constant and simply update the parameter of the weak learner, which are probability parameters for respective features. Overall, we find the online learning strategy is beneficial and keeps SAFE-PC agile and evolving in its ability to detect new forms of phishing campaigns.

### 4 Datasets for Our Evaluation

In this section, we describe our datasets and the processing that is needed to make free form data like email into a form that rigorous evaluation can be done on it.

#### 4.1 Collection Methodology

We show the dataset collection process in Figure 2 which represents the current operational flow used by the Messaging/Mail Division of our university’s IT organization. The complete dataset of phishing emails were all manually gathered, analyzed, and annotated by IT analysts and we believe that the labeling for the ground
Fig. 2: Current production dataflow of emails at a tier-1 research institution. Data stores 1, 2, and 3 are passed through Sophos. Those that are caught form part of our Caught Dataset, the other part coming from Data store 4. The messages in Data stores 1, 2, and 3 that are not caught by Sophos form our Uncaught Dataset.

The Caught dataset (Dataset 2) is composed of 388,264 emails, each with header, subject line and message body. There are 190,148 unique emails after feature space projection. These emails were automatically caught by the Sophos filters when we ran them through our installation. These emails range from July 10, 2013 to July 30, 2014. Our Sophos installation had the most up-to-date filters, with the most up-to-date rules, at the time of testing (February 15-19, 2015). Since the mails from early in the dataset still had the benefit of the latest filters in our evaluation, in practice not all of these emails would have been caught by Sophos.

4.3 Dataset 3: Legitimate Emails

The Legitimate or Benign dataset (Dataset 3) is composed of 158,444 benign emails, each with a header, subject line and message body. The number of unique benign emails is 137,684. However, we use a subset of the total legitimate emails for our evaluation. The final set consists 106,182 emails comprised from several public sources.

All of the legitimate emails were collected from public listserv hosted on .edu domains from universities in North America. Half of the of the legitimate emails are from Jeb Bush’s publicly released emails as his time as governor of Florida, with only the subset related to financial transactions (that he redacted soon after publication) being used here. The motivation for using these emails were that they consists of real budget and financial discussions that fit will with our university context and can be considered close to some phishing emails related to financial information. The content includes meeting announcements for university seminars, academic list-servs dealing with research, social activities, announcements and a Mozilla bug reporting list. These emails are used to replicate emails that would be received in an academic environment. Unfortunately, we were unable to use samples of live university emails during the same time period of the phishing dataset for privacy reasons. Prior work in phishing classification use the Enron email set [11] as the benign set, but due to the subject matter of the emails that we sought, more recent emails in relation to our phishing emails with financial information better models a university environment. The final mix of the unique legitimate emails that comprise dataset 3 is Jeb Bush Finance (53%), Listservs on Cytometry (33%), Mozilla (12%), Vision list (4%), and CompSci Colloquium (1%).

4.4 Data Cleansing

We removed duplicates within each dataset by using the MD5 hash of the email body. A number of emails in each dataset were clearly spam and it appears that the spam. We implemented a simple spam filter that removes common spam campaigns, which were primarily focused on soliciting medicine to treat erectile dysfunction. Examples of spam include emails with the approximate...
spelling of “cialis” or “viagra.” We achieve this by using a simple regular expression that looks for specific words within each email. Several emails in our raw dataset contained encoded sections, such as the body of the message being encoded in Base 64. We decoded such email and replaced it with the ASCII output.

We also analyze the emails to address strategies that replace the deliberate misspelling or modifications of words to subvert feature extraction. The body of each email has been automatically analyzed to (i) remove punctuation, (ii) substitute the symbols used for misspelling with the correct letters they represent, (iii) remove misused extra whitespace characters, and (iv) correct misspelled words. We define a dictionary of similar looking characters to map erroneous characters to their correct symbol e.g., “0” → “O”, “1” → “l”, “$” → “$”.

Whitespace that appears within words is addressed as follows. Given the body of an email, our algorithm verifies if each word is English with PyEnchant; next, the algorithm concatenates characters, one-by-one, and verifies if the obtained word is an English one. The word concatenation terminates when the word length is greater than a parameter that we set to ten. The algorithm then verifies that the word is a valid English word (here named ENGLISH_WORD) and replaces the word within the body of the email. If the word is Non-English (named NON-ENGLISH_WORD), the word is left alone and the algorithm continues to the next word and terminates when there are no other words in the email body.

We clarify this functionality with the following example. Given the text “u r g e n t l y updat acc ount”, the algorithm starts concatenating the words composing the text and verifies if the obtained words are English one or not. The algorithm stops when generates the word “urgently updat”, since its length is greater than ten characters. During the concatenation, the algorithm finds the words “urge”, “urgent” and “urgently” to be English words; therefore, it select the last one for the new version of the text. The algorithm restarts concatenation to form the following word, i.e., “updat”, and stops when it generates the word “updataccount” (again, its size is greater than 10 characters). Since no English words are found, the algorithm selects the first non-English word for the new version of the text, i.e., “updat”, and restarts concatenation from the next word, i.e., “acc”. The new concatenation stops when the algorithm generates the word “account” since no other words are in the text; however, “account” is identified as an English word and inserts the word into the new version of the text. Therefore, the new text is “urgently updat account”, which does not contain any extra whitespace characters.

The new version of the text is then analyzed to replace the misspelled words via the PyEnchant package. For each misspelled word, PyEnchant provides a list of suggestions, i.e., words that can replace the misspelled one. We calculate the Levenshtein distance 2 between the misspelled word and each suggestion; the suggestion with the lowest distance replaces the missspelled word in the text. The word “updat” replaces “updat” in our example, which has the lowest Levenshtein distance based on the set of possible corrections from PyEnchant.

4.5 Baseline Comparison: Sophos PureMessage

Our university’s central IT organization uses Sophos PureMessage [17] on its Microsoft Exchange servers. This is an enterprise grade email scanning software, with modules for filtering spam, phishing, and malware. We have discussed details of this software in Section 2. Once an email is scanned by all the rules, those that are matched in the email are summed and if the total is greater than a configured threshold, the email is labeled as spam. For our evaluation, we ran Sophos with default settings, which encompass the threshold configured to 50%, i.e., the default value when Sophos is installed, spam, phishing and malware filter modules activated, etc. It should be noted that the rules used by PureMessage are automatically updated by Sophos frequently throughout the timespan of our datasets, with typically tens of updates per hour.

5 Results

We conduct five experiments to evaluate SAFE-PC and compare its performance to SpamAssassin. The first experiment is to evaluate the effectiveness of SAFE-PC in detecting phishing messages that were not caught by Sophos, regarding false positive and false negative. The second experiment is to evaluate if SAFE-PC is still able to detect the phishing messages that were caught by Sophos. The third experiment tests the online learning mode of SAFE-PC. In it, the classifier incrementally updates as new phishing messages are collected through time.

We note that this third experiment performs cross-validation in the time order. We will highlight our cross-validation setting and experiments in Section 5.4. For the first three experiments, three configurations of RUSBoost weak learners are explored: five Gaussian Naive Bayes (RG5), 500 Decision Tree (RD500), and 100 perceptrons (RP100). The fourth and fifth experiments evaluate the performance of SpamAssassin on the same datasets. The final experiment measures the runtime performance of the tools. For the each classifier and experiment, we use an imbalanced dataset to train and test the classifiers. This imbalanced dataset comprises of 90% legitimate emails and 10% phishing emails.

1. PyEnchant is a spellcheck package for Python that is based on the widely used Enchant library (http://pythonhosted.org/pyenchant/).

2. Levenshtein distance between two words is the number of deletions, insertions, or substitutions required to transform the source string into the target string. We used the python-Levenshtein package to compute that distance (https://github.com/ztane/python-Levenshtein/).
5.1 Phishing email topics

Phishing attackers continuously change topics to deceive unsuspecting users. One technique used by attackers is to send emails that are temporally relevant. An example is to send emails about Blackboard toward the beginning and end of semesters. Blackboard is used by students to enroll in courses and check grades (among a host of other course-related functions). Table 2 details popular phishing emails each month (we sample the months in the interest of space), along with the number of phishing emails collected each month. These values are further separated into the number of phishing emails that were caught and the number that were not caught. Low observations in December and July could be either due to an incomplete dataset or a genuinely low volume of phishing activity.

Some interesting (subjective) observations can be made from the data of Table 2. The number of phishing emails significantly drops when students are on breaks (Winter/Summer sessions). The three super-categories that we create (as described in Section 3.1) are Financial Institution, University, and Technology Provider. Of these, we find the financial institution and university features frequently appear as a topic of phishing emails. They are a popular subject in both the caught and uncaught datasets. The presence of financial institution is an expected observation because attackers ultimately want to make money so getting direct access to the source of money results in the most profit for work. Banks that were targeted in these popular campaigns are as follows. The most popular were RHB and RBS, next was Western Union, and then came SunTrust, American Express, Maybank and Standard Bank. Unfortunately, since the nature of our analysis is a post-mortem analysis, most of the links in the phishing email corpus had been removed at the time of our analysis. Among university phishing campaigns, Blackboard was by far the most prevalent. Among IT providers AOL and Google, the latter through Google Docs, appeared most frequently.

Another interesting observation is the lag time between the publicly reported phishing attack and when a significant volume of such attacks appears in our datasets. The lag was typically about three weeks while some popular phishing campaigns had longer lag times. For example, a phishing attack involving Google Docs was reported by Symantec [18] in March of 2014. It did not appear with any noticeable volume at our university until August 2014.

From Table 2, we also see that emails with very similar topics are repeated months after first observation. An example is an email from Blackboard saying that there has been some update or that there is a new message. The phishing email was observed in volume in February 2014 and was caught by the Sophos Filter. But, we saw in May 2014, a very similar email about Blackboard was undetected by the Sophos filter. The message differed slightly from the original as can be seen from the excerpts below. Such an observation exposes the undesirable specificity of some of the message filters. Instructively when a similar Blackboard phishing message appeared in volume in August 2014, it was caught.

Caught phishing message: You have received a new update course form for your new semester on Blackboard Technology system. In order to view this update you are to click the link below.

Missed phishing message: Blackboard Course Online - New course Online has been update in your new class online for your project, in order for you to view click on the link below ..

5.2 Experiment 1: Uncaught Phish

In this experiment, we evaluate the capability of SAFE-PC to detect phishing emails that were not caught by Sophos filters. The raw detection rate should be considered in conjunction with the fact that SAFE-PC creates its features automatically, thus making it easier to update existing features as new phishing campaigns emerge, sometime de novo but often by modifications of existing campaigns.

To conduct this experiment, we segment our dataset 1 consisting of 37,606 emails into training and test datasets as shown in Figure 3. One observation is that the number of unique emails after feature space projection is only a fraction of the total number of emails — a 16.7-fold reduction. The large reduction indicates that a number of distinct emails are identical in our feature space, possibly indicating changes being made by the phishers to deceive the end user.

In segmenting into the training and the test emails, we make the time periods disjoint and further, introduce a gap in time between these two sub-datasets. The gap reflects the latency of collecting/training a classifier and deploying SAFE-PC in a real system. We keep the training and test data sizes to be equal.

Figure 5a shows the confusion matrix from experiment 1 using RUSBoost with 5 Gaussian Naïve Bayes (GNB) classifiers (denoted as “RG5”). The confusion matrix shows what percentage of legitimate emails SAFE-PC classified as legitimate and what percentage of phishing emails SAFE-PC classified as phishing. We find that 71% of phishing emails missed by Sophos were correctly detected with a 15% false positive rate. Although the accuracy is nowhere near the coveted 95% detection rate, we argue that the uncaught dataset consists of the most challenging phishing emails. By inspecting the false negative phishing emails, we discovered some challenges that we previously did not consider in the design of SAFE-PC. Some false negative emails contain very short and cryptic messages with a URL. Determining if these short emails are phishing, spam, or legitimate is difficult even for a trained user (aka us) without visiting the potentially malicious URL. These emails simply do not
TABLE 2: Most popular campaigns separated by month. Individual campaigns are divided within each month by a |

<table>
<thead>
<tr>
<th>Month</th>
<th>Uncaught</th>
<th>Uncaught Subjects</th>
<th>Caught</th>
<th>Caught subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-2013</td>
<td>16,187</td>
<td>Bank account needs updating using website given to validate</td>
<td>10,399</td>
<td>Bank account needs to be updated, click here...</td>
</tr>
<tr>
<td>12-2013</td>
<td>5</td>
<td>Click to read a new message</td>
<td>1</td>
<td>Account is expiring, click to renew</td>
</tr>
<tr>
<td>1-2014</td>
<td>282</td>
<td>Click here to view a product</td>
<td>Account needs upgrade, click here to read more</td>
<td>134</td>
</tr>
<tr>
<td>2-2014</td>
<td>14,862</td>
<td>New inquiry, agreed to subscribe for membership</td>
<td>Bank statement is attached</td>
<td>Bank account updated, verify and update via secure attachment</td>
</tr>
<tr>
<td>3-2014</td>
<td>2,287</td>
<td>Bank card will expire, click here...</td>
<td>Phrase or name and click link</td>
<td>15,085</td>
</tr>
<tr>
<td>4-2014</td>
<td>2,303</td>
<td>Verify online account</td>
<td>23,906</td>
<td>gibberish and links</td>
</tr>
<tr>
<td>5-2014</td>
<td>2,863</td>
<td>blackboard updated, click on link</td>
<td>338,768</td>
<td>gibberish and links</td>
</tr>
<tr>
<td>7-2014</td>
<td>144</td>
<td>Husband died, need to distribute money, reply with personal info</td>
<td>incoming email pending delivery due to upgraded database, use link... Out of the office</td>
<td>1,033</td>
</tr>
<tr>
<td>8-2014</td>
<td>4,981</td>
<td>Message could not be delivered</td>
<td>Want to order products, need to speak in person</td>
<td>Document uploaded to google docs, click here...</td>
</tr>
</tbody>
</table>

Total number of Phish emails 37,606
Feature Space Projection
Total number of unique phish emails 2,248
Training Set
899 Phishing Emails (~40% of total)
8,091 Legitimate Emails
Legitimate:Phish=9:1
Gap
450 Phishing Emails (~20% of total)
8,091 Legitimate Emails
Legitimate:Phish=9:1
Testing Set
899 Phishing Emails (~40% of total)
8,091 Legitimate Emails
Legitimate:Phish=9:1

Training and Testing for Uncaught Data Set

Fig. 3: Training and testing datasets for uncaught i.e., the emails that were not detected as phishing by Sophos.

TABLE 2: Most popular campaigns separated by month. Individual campaigns are divided within each month by a |

- The uncaught dataset. This work is complementary to ours and has been the focus of most prior work on phishing detection. Nonetheless, SAFe-PC outperformed the other phishing detection software as we will see in our evaluation results for SpamAssassin.

- Several phishing emails that were classified as legitimate contain ambiguous messages. For example, some of the observed emails in this case contained a link with a short greeting (2-4 words). These types of messages are quite challenging to classify because without context, one cannot safely determine if the email is truly mischievous in nature. For example, receiving an email with only a URL from a colleague after a meeting could be considered OK but receiving an unsolicited email with a URL from a stranger is suspicious. An automated technique will need to bring in more external context to perform well with these challenging phishing emails. Other observations include emails that contain gibberish with a URL. Some of the other emails that were misclassified also appeared to be spam or “clickbait”. For example, Reese Witherspoon Stuns In A Little Black Dress [URL appears here] If you can’t click the above link, move this email to your inbox and then click!. For the few legitimate emails classified as phish, we observed that the email has only a few features. The features that are present (e.g., “thanks”, “continue”, presence of a URL) suggested to SAFE-PC that the email is phishing. Even to our human observation, such emails appeared suspicious.

- Potentially, our classifier is overfitted. To investigate such problem, we trained our classifier with the identical training dataset and predict it again. Then, we checked how much differences between prediction with the training testing dataset. As a result, correct legitimate email prediction rate is 89.9%, correct spam email prediction rate is 66.4% and total correct prediction rate 87.6%.
Because there was 4% difference in the total prediction rate, we believe that our classifier is not overfitted.

5.3 Experiment 2: Caught Phish

SAFE-PC must be able to accurately detect emails that were caught by conventional anti-phishing software such as Sophos. The classifier for this experiment was trained on 388,264 emails. Following the same steps as outlined in the uncaught dataset, the unique emails are extracted from them — this leads to 190,148 unique emails, a factor of ∼2 reduction, much smaller than with the uncaught dataset. The breakdown into training and test sets, along with the date ranges, is shown in Figure 4. As before we train over one time window of emails, introduce a gap in time, and test over the next time window of emails. This is to verify the effectiveness of SAFE-PC with new phishing campaigns.

Total number of Phish emails 388,264

Training Set
5,000 Phishing Emails (≈2.6% of total)
45,000 Legitimate Emails
Legitimate:Phish=9:1
7-10-13 4-24-14
Start End

Gap
180,148 Phishing Emails
≈94.7% of total

Testing Set
5,000 Phishing Emails (≈2.6% of total)
45,000 Legitimate Emails
Legitimate:Phish=9:1
5-13-14 7-30-14
Start End

Training and Testing for Caught Data Set

Fig. 4: The training and testing datasets for caught, i.e., the emails that were detected as phishing by Sophos.

The results are shown in Figure 5b. Thus, SAFE-PC is able to detect 69% of the emails that Sophos caught with the RG5 configuration. SAFE-PC flagged 0% of legitimate email messages as phishing.

For the caught false negative messages, the emails just contain gibberish and a URL. The vectors representing these emails have very few representative features, resulting in the classifier incorrectly labeling these messages. These observations are similar to what was shown in the previous experiment for the uncaught case.

Next, we consider the possibility of overfitting the data. As we did in , we trained and tested on the same dataset and compared the prediction rate with a disjoint training and testing dataset. When using the same training and testing dataset, the correct legitimate prediction rate is 99.7%, correct phishing prediction rate is 73.6%, and overall accuracy is 97.1%. When compared to that classifier trained under disjoint training and testing sets, the overall difference is 0.2%. We conclude that our classifier is not overfitting.

5.4 SAFE-PC-Online

Online learning incrementally retrains the classifier for new observations. Figure 6 illustrates the retraining after incremental observations. We first train SAFE-PC-Online using the training dataset as our first two experiments. Next, we introduce new observations to SAFE-PC-Online using a portion of test dataset from the previous experiments number. The portion of the data at each iteration is denoted by Δ. Then, we predict samples for the remainder of the testing set. In our experiments, we define the size of Δ as 5% of total samples including training, testing, and time gap samples. Just like the previous two experiments, we evaluate the online learning algorithms, separately for the uncaught and the caught datasets.

Note that the evaluation of our online learning scheme is cross-validated realistically. The online learning scheme receives new phishing samples periodically. The testing set is disjoint from the training set, in the temporal sense, such that the testing set always contains phishing emails that were observed after the latest sam-

(a) Uncaught phishing dataset. (b) Caught phishing dataset

Fig. 5: Confusion matrices from the evaluation of SAFE-PC-Batch with RUSBoost with 5 Gaussian Naive Bayes classifiers being used as weak learners.

Fig. 6: SAFE-PC-Online: Training and Testing sets at each round. At each round, an additional Δ samples are added to the training set and taken from the testing set.
ple in the training set. There may be temporal dependen-
cies because of the deceptive semantics that campaigns
use over a time period. Practitioners collect and train
classifiers periodically and our evaluation is designed to
reflect the steps that system administrator would take to
protect users.

As a result, our classifier shows not only reasonable
results as described in Fig. 6 but also maintains such
results under periodic retraining.

**Learning Algorithm Selection:** Through experimen-
tation, we evaluated several classification algorithms to
work well with RUSBoost. Our initial experiment ex-
plorated 15 different configurations of the online clas-
sifier, corresponding to 3 different ensemble models—
RUSBoost, AdaC2, and Cost-sensitive Boosting—and 5
different weak learners—Perceptron, Na"ıve Bayes, Deci-
sion Tree, Multi-Layered Perceptron, and Na"ıve Binary.
Open source implementations in scikit-learn were only
available for 3 of these configurations and we wrote our
own following closely the algorithm descriptions in [15],
[19], [20], [21]. The infrequent use of imbalanced boosting
together with online learning explains the fact that we
did not find existing implementations of the classifier
configurations. We only show three representative clas-
sifier configurations in Figure 7. In this figure, R denotes
to RUSBoost, G Gaussian na"ıve Bayes (GNB), D decision
tree, and P perceptron.

We find that the choice of the boosting algorithm does
not have a significant effect on the performance. There-
fore, we choose the fastest executing boosting algorithm,
RUSBoost. However, the choice of the weak learner has
an impact. Perceptron and decision tree perform the
best (multi-layer perceptron gives incremental improve-
ment) in detecting phishing emails, while na"ıve Bayes
gives a reasonably good performance. However, with
regard to classifying legitimate emails, both perceptron
and decision tree perform poorly, while na"ıve Bayes
performs almost perfectly. RUSBoost with decision tree
and perceptron learners gave high false positives with
benign emails — about 30% for the uncaught dataset
and 50% for the caught dataset. Further, in the uncaught
dataset, the false positives rise with additional training
for these two weak learners. The results were optimized
by exploring the number of these weak learners, e.g., in
Figure 7, we see the poor result even with 500 decision
trees. In SAF\textsuperscript{E}-PC-Batch on the other hand, decision tree
as the weak learner gives competitive detection perfor-
mancess to the more powerful learners (like na"ıve Bayes).
However, this was a non-incremental decision tree which
does not fit our online learning use case. The incremental
form of the decision tree loses its performance advantage,
likely because it is not able to have accurate samples
through boosting (because the normalization factor for
the distribution is not known in the online mode) and
the effect of small inaccuracies tend to get magnified
in the (multi-level) decision tree. On the other hand,
with na"ıve Bayes, even using a small number of weak
learners gives good performance — five weak learners
in the both datasets seem sufficient. The small number
of weak learners has an execution time performance
advantage because the learners have to be executed se-
quentially. RUSBoost with 5 na"ıve Bayes classifiers (RG5)
misclassifies few legitimate emails and with respect to
the phishing emails at the last round, it detects 74% for
the uncaught dataset and 67% for the caught dataset.
Considering that in this domain, it is critically important
to have low false positive rate, we believe na"ıve Bayes is
a suitable weak learner for online learning in SAF\textsuperscript{E}-PC.

In line with the result for SAF\textsuperscript{E}-PC-Batch, we also
present the confusion matrix for the SAF\textsuperscript{E}-PC-Online
results, separately for the uncaught and the caught
datasets in Figure 8. For the confusion matrix, we take
the last round of online learning, i.e., one where the
last 5% of the dataset is used for testing. We find that
with the online learning, as SAF\textsuperscript{E}-PC-Online learns new
legitimate emails, in this final round, it does not misclas-
sify any legitimate email. For the uncaught dataset, the
phishing email detection percentage is 74%, while for the
caught dataset.

Expectedly, uncaught performance is improved as SAF
\textsuperscript{E}-PC-Online.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig7.png}
\caption{SAF\textsuperscript{E}-PC-Online test result at each round.}
\end{figure}
Uncaught phishing dataset. (b) Caught phishing dataset
Fig. 8: Confusion matrices from the evaluation of SAFE-PC-Online, $\Delta = 8$

Fig. 9: Time to train SAFE-PC-Online as we increase the number of rounds. Expectedly and reassuringly the time to train in each round stays approximately constant with the increasing number of the round.

after SAFE-PC-Online is trained with these new samples, the performance improves. Therefore, our SAFE-PC-Online is not only superior to SpamAssassin [22], but also SAFE-PC-Online can adaptively adjust new feature vector patterns without complete retraining.

**Overfitting**: We also measured prediction results with training datasets to proof our online classifiers are not overfitted. To perform it, we apply the identical scheme used in testing dataset prediction. In terms of uncaught prediction, the average uncaught prediction rate of the training dataset for all rounds is 88.6% and that of the testing dataset for all rounds is 89.6%. On the other hand, the average uncaught prediction rate of the training dataset for all rounds is 96.7% and that of the testing dataset for all rounds is 96.5%. Due to such negligible difference between prediction rates of both data, we believe our online classifier also does not show the overfitting problem.

**Training Speed**: We quantify the time it takes to train SAFE-PC-Online at each round and show the result in Figure 9. Initially, there is a large fixed cost for the training, on 45% of the dataset. But subsequent to that, as each additional $\Delta$ worth of samples is added, the training is incremental, and not proportional to the total amount of training data. This is precisely the goal of an online learning algorithm. So operationally, when a new batch of data is brought to SAFE-PC-Online, corresponding to 5,000 new emails received at the central mail server in our case, SAFE-PC-Online will only take about 40 minutes to recalibrate its model. This can be read from the figure for the caught dataset. The time to classify the incremental set of uncaught mails is less than 4 minutes and the reduced time is proportional to the smaller size of the uncaught corpus compared to the caught corpus. It seems operationally feasible to deploy SAFE-PC-Online considering the short period of time to do the incremental retraining.

### 5.5 SpamAssassin Experiments

SpamAssassin [22] is a widely-deployed open source spam filter. It allows to identify spam emails by using a variety of local and network mechanisms, such as rules for header and text analysis, link blacklist, and Bayesian classification. Despite its aim to detect spam emails, SpamAssassin is also considered as a tool to classify phishing emails and is commonly compared in proposed anti-phishing solutions [23], [24], [25]. Notwithstanding the many solutions proposed by industry and research community, SpamAssassin is the only open source, widely-used, commercial grade anti-phishing tool available for comparison. For these reasons, we conducted a comparative analysis with the recent version of SpamAssassin (version 3.4.1) over the same datasets used for SAFE-PC.

SpamAssassin uses a set of rules and network tests as well as a Bayesian classifier to determine whether an email is spam or not. The rules leverages regular expressions to perform checks on the header and body of each email, e.g., it checks the gap time between the *Date:* and the *Received:* header fields, the presence in the body of words that refer to selling medicine, etc. Network tests allow the access to online resources to improve SpamAssassin accuracy, for example by verifying if the email is listed in Razor (a distributed, collaborative, continually updated spam catalogue), if the email contains some blacklisted link, and so on. The Bayesian classifier tries to identify spam emails by looking for *tokens*, i.e., words or short character sequences that are commonly found in spam or ham; once trained with spam and ham email samples, the classifier correlates the use of tokens with spam and ham emails and then uses Bayes’ theorem to calculate a probability that an email is spam or not. The result of each detection mechanism, i.e., each rule, network test and the Bayesian classifier, is a score; the scores can be either positive or negative, indicating spam or ham, respectively. An email is matched against all detection mechanisms and SpamAssassin combines the results into a global score. The higher the score, the higher the probability that the message is spam. If the obtained scores is greater than a configured threshold, the analyzed email is labeled as spam by adding X-Spam-Status: Yes in the header of the email. We used the default threshold (i.e., 5.0) and rules, and enabled both
TABLE 3: Training set and Testing set for uncaught and caught experiments (unique emails) for SpamAssassin.

<table>
<thead>
<tr>
<th></th>
<th>Uncaught</th>
<th>Caught</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training</td>
<td>Testing</td>
</tr>
<tr>
<td>Legitimate</td>
<td>8,091</td>
<td>45,000</td>
</tr>
<tr>
<td>Phishing</td>
<td>899</td>
<td>5,000</td>
</tr>
</tbody>
</table>

the Bayesian classifier and the network tests provided by SpamAssassin.

In order to compare SpamAssassin with SAFE-PC, we used the same uncaught, caught, and legitimate raw email datasets. Table 3 shows the used datasets.

For the uncaught experiment, we trained the classifier with both the legitimate and the uncaught training set and ran SpamAssassin on the related testing sets. Figure 10a shows the confusion matrix for the uncaught emails.

![Confusion Matrix](image)

(a) Uncaught phishing dataset. (b) Caught phishing dataset

SpamAssassin correctly detected 9.7% of the phishing emails showing that most of the phishing emails bypassed SpamAssassin’s tests and reach the inbox of the users. On the other hand, 11.6% legitimate e-mails are not correctly classified by SpamAssassin. Through a closer look into the misclassified emails, we observed that most misclassified phishing emails are basically: (i) fake emails from Blackboard, (ii) emails with a small phrase or name or random words and a link, and (iii) fake emails requiring the verification of an account. These kinds of emails are often not recognized by SpamAssassin because they do not contain any particular spam/phishing topic or words or blacklisted link, and present a well-formed header and body. For the legitimate emails, there are two main reasons for the misclassification, i.e., the absence of some fields in the header (e.g., To: or Date:) and the presence of a malformed Message-Id: in the header.

For the caught experiment with SpamAssassin, we obtained the confusion matrix shown in Figure 10b. We found that only 2.1% of caught phishing emails were correctly detected, while 6.5% of the legitimate email messages were mistakenly flagged. A closer inspection of the misclassified mails indicated the same results as for the uncaught dataset. In addition, misspellings of common words also contributed to the misclassification.

The obtained results show the ineffectiveness of SpamAssassin against phishing emails and points out the need to develop more customized extensions for phishing detection as part of its toolchain. We are aware that there is a commercial plug-in called PhishPatrol from Wombat Security, which however is neither free nor open source.

5.6 Performance

SAFE-PC is meant to be used in real world detection systems and thus, the time to classify an incoming message is an important metric. The time to train our classifier is 418 ms on an average per email and the time for prediction is 296 ms per email, on an 8-core VM, each of 3.1 GHz and a total of 12 GB of RAM. The prediction time of SAFE-PC is 15573X slower than Sophos, though the classification time may still be tolerable in many deployments, and we do not need to retrain emails which are trained previously. Note that Sophos PureMessage uses rules and weights to determine if an email is phish, which partially explains that higher performance compared to SAFE-PC. Further, we cannot compare the training time for Sophos because it is not clear how the rules and weights are derived. In future work, we will order the features by importance and consider if a subset of the features can give lower but acceptable accuracy and precision. In comparison, SpamAssassin is much faster in training but slower in prediction, with training time per email of about 13 ms and prediction time per email of 1.6 s (for phishing) and 970 ms (for legitimate). A large part of this cost is the remote lookup involved in the “online test” of SpamAssassin. Although SpamAssassin can train faster, it requires does not provide online training. Training speed of SAFE-PC may be superior to SpamAssassin in the condition where continual training is required. Furthermore, machine learning algorithms of SAFE-PC are implemented using python. So, if we use optimized codes based on C and sklearn. It would be much faster.

6 Discussion

Here we discuss some issues with the current state of SAFE-PC and their possible resolutions.

Adversarial Machine Learning. A logical question that arises is how would an adversary defeat SAFE-PC if she were aware of the algorithm. We consider a few such situations and discuss how feasible each is or not. One attack is to split the phishing features over two or more emails so that neither email is flagged. However, this is not typically done for legitimate emails from professional organizations. The occurrence of sending an email and then sending a second one which says “Oops I forgot to send the link.” happens only with sloppy personal communication. Even in the rare cases a message is split into two or more emails, they will be expected (by the
human recipient) to come from the same sender. We can then aggregate mails from the same sender to the same receiver within a short window of time into one thread and treat them as a single data point. Since we are extracting many features that are text based, the adversary can embed images with text in them. To deal with this setting, SAFE-PC can perform OCR for all embedded images and if it fails, can indicate that the recipient should not act on any text that may be embedded in the image. A third attack scenario is to obfuscate words such as by using spaces, UTF character substitution, etc. This kind of attack can be countered by a more elaborate, albeit tedious, tokenization of the words, thus creating the words that feed into our current classifier without any change to it. Other obvious attempts to bypass the classifier risk the chance of changing the meaning of the emails significantly enough that it renders the phishing campaign less effective. For example, changing a word to a different word that is not a synonym, in an attempt to defeat our synonym extraction functionality, will suffer from this drawback for the adversary.

Synthetic generation of Phishing emails. Data is the key component for any machine learning application. In SAFE-PC, synthetic data can be used to introduce new email messages and thus create a more robust classifier. Using tools such as Python’s NLTK [12], words and phrases in phishing emails can be mixed and matched in order to create new email messages with features that have previously never been seen. Different structural components of emails, such as different signature locations, images and other media, altering header fields, etc. can also be mixed and matched. This will lead to a more robust classifier that can potentially identify previously unseen phishing emails in the wild with more accuracy.

This would introduce more variety in the dataset, and could expose new techniques that attackers could employ in future phishing campaigns. Training on these synthetic emails before they have been seen in the wild will make classifiers more robust in detecting previously unseen phishing attacks.

7 Related Work

Phishing emails have been automatically clustered and then classified in a number of studies. At a high level, these studies fall short of a convincing evaluation showing that the heuristics used in their approach are sound and capable of detecting phishing emails that are different from the ones that they have been trained on. It is indubitable that the approaches, including ours, employ various machine learning techniques and it is only through empirical validation on large real-world datasets can an approach become credible (as opposed to say a theoretical proof of soundness, which may be possible in some other problems). As we have seen in our work, we have had to refine various aspects of SAFE-PC’s algorithms — the features that are used, what classifier is used and how — in an iterative process as we have analyzed real-world datasets. Also, another aspect of the existing work is that they do not consider the fundamental fact of multiple kinds of very different phishing messages and the fact that phishing campaigns evolve over time with tweaks to phishing messages.

Prior work that is similar to ours is [3], where the authors apply natural language processing to analyze emails to test if they are legit, phish or unknown. This serves a useful and important starting point for our work. We have already pointed out, in Section 3.1, the shortcomings of their approach - weak basis for creating the word features and lack of higher level language features. Related to this work is research conducted by Verma et al. that employs a t-test and WordNet to select features present in phishing emails and then use these features to detect phishing emails [11]. The techniques used in this approach to select features provide structure to the feature selection problem, which has previously been ad hoc, but the dataset used in the study is composed of phishing emails from 2007. The outdated phishing dataset includes phishing emails that use outdated deceptive techniques. This technique of using statistics to identify features was shown to provide accurate detection, but because of the outdated dataset, we argue the ability to detect more current phishing emails with updated phishing semantics/techniques would depend on retraining the classifiers with a new dataset. Thus, our technique, which includes feature selection based on prior phishing emails from the same environment as testing emails as well as online retraining to continuously update phishing classifiers fills this gap.

One of the earliest papers, and highly cited, is [23]. In it, the authors present an automated solution to decide whether some communication is deceptive. The decision “based on information from within the email or attack vector itself (an internal source), combined with information from external sources. This combination of information is then used as the input to a classifier, the result of which is a decision on whether the input contained data designed to deceive the user” [23].

While the approach shares some similarities with SAFE-PC, it does not consider a wide set of features (10 features as opposed to our use of 806 features) nor does it analyze a large number of phishing messages (860 emails). We believe our approach of using a large number of features as well as synonyms of these features makes reverse engineering the classifier to develop phishing campaigns to bypass SAFE-PC more difficult. The large number provide more coverage of text that could be present in phishing emails. A smaller feature set could still be effective in detecting phishing emails, but provides less coverage and thus could be easier to reverse engineer and develop a phishing campaign that bypasses the tool.

Prior work has applied online learned to classify phishing emails [26]. This work is using a similar technique to the one we present, but the dataset used in experimentation contains a limited number of phishing
emails and is an older dataset. Thus, detection results achieved using this dataset may be based on the ability to detect out-of-date phishing techniques. Another limitation of this work is the lack of semantic features to classify phishing emails.

Using semantic information and annealing to identify and weight phishing features is another technique that creates structure around feature selection [27]. Weighting specific features require retraining as phishing campaigns change over time. This gap is filled by our research, which adds retraining to the classification process. Also, this work only uses 400 total emails to perform the classification. This limited number of phishing and legitimate emails adds uncertainty to the generality of this technique.

In [28], the authors develop a set of features particularly characteristic of phishing attacks, including graphical features, and then they use machine learning techniques to determine the relative importance of the features for labeled training data. This yields a classifier integrating the features in such a way that new emails can be classified correctly with a high degree of confidence. Through active learning, they seek to keep the filters updated as new phishing campaigns emerge. The paper makes important contributions in the extraction of graphical features from email messages.

8 CONCLUSIONS
This paper proposes a methodology to combat the prevalent threat of phishing messages sent through email. We developed a system called SAF-E-PC for detecting new kinds of phishing campaigns, that are evolved from prior phishing campaigns. SAF-E-PC uses real world phishing and legitimate email datasets from the central IT organization of a tier-1 research university, a total of 425K phishing and 158K legitimate emails. It first does data cleansing, then extracts features from each message’s header and body, imbued with an understanding of structures of phishing emails, e.g., that word placement is often less important than presence of a word and a word and its synonyms should be considered as the same feature. Then it applies a RUSBoost classifier using the phishing emails and legitimate emails. We perform a thorough evaluation using the real-world corpus and compare SAF-E-PC to the state-of-the-art email protection software from Sophos, deployed on our central mail routing servers, and SpamAssassin. Our insights bring out some common phishing strategies that are bypassing current tools.

9 ACKNOWLEDGEMENTS
The authors would like to thank Paul Wood for his comments on early drafts of this paper; Keith Mc Dermott and Brian Berndt from Information Technology at Purdue (ITaP) for their valuable insight in understanding the anti-phishing efforts and data preparation used in our experimental evaluation; Jonathan Fulkerson for his valuable discussion regarding organization the the paper.

REFERENCES


APPENDIX

PREPROCESSING AND FEATURE DETAILS

Below are the details for removing spam messages within our dataset as well as the root words used for our synonym-based features.

Filters for Spam emails A sample list of words used to filter out spam messages from our datasets is as follows: pharm, viagra, cialis, levitra, drug, pill, med, yourlatestauto, newvisioninfo, Glaucoma, thelatestmay-scores, dibzee, newmaytv-specials, life-insurance-policy, betspetstuf, forocriminologos, esminkch, Maid-Service, cellulite, TV-segment, thebestnewmay-scores, online-health, Bride-Can, Window-Installation, your-backyard, Celtrixa, Kelly-Blue.

Feature words used in SAE-PC Below are the words (and their associated synonyms) used in SAE-PC. Each feature is either found in the body or in the subject of the email. able, access, account, accounts, action, activate, activated, activation, active, activities, admin, administrator, advise, alert, alerts, allie, allied, allow, alt, alternative, animal, answered, anti, anything, apologize, asthma, attach, attached, attachment, attempt, attention, authorize, aware, away, back, balance, bank, banking, banks, banner, believe, best, better, beyond, bill, biz, blackboard, blog, canada, canopy, captain, card, cards, care, cars, carson, cervices, change, changed, check, chief, children, choosing, chord, click, clicking, college, complete, computer, confirm, confirmation, continue, convenient, correct, correctly, could, credit, creek, critical, customer, customers, daly, dangerous, hari, data, days, deactivate, deactivated, dear, decline, department, depression, description, details, different, digital, dir, dire, direct, director, disposition, doctor, document, domain, double, due, easy, effective, email, emails, ensure, error, even, every, everything, executive, expire, failure, fastest, feature, federally, filelocker, filename, financing, find, finder, first, following, food, found, frank, franklin, fraud, freeze, full, function, gallery, get, good, got, gothic, greg, group, growth, hammad, head, head, hello, help, helpful, hen, high, hold, honda, however, husky, identify, images, immediately, important, inc, includes, inconvenient, incorrect, ind, individuals, information, interruption, invoicing, issue, itap, kindly, know, latest, lead, learn, legal, let, letting, life, like, lin, link, locked, lodge, log, logging, logo, logos, long, longer, mai, mail, main, maintenance, make, making, mall, man, managing, many, marks, may, media, medium, member, men, message, method, minimum, miss, mistake, mobile, mohammad, monday, monitored, month, monthly, moodle, much, multiple, need, needed, never, new, notice, notification, number, officer, often, oil, okay, one, operation, pack, page, pain, passion, paul, pay, paying, payment, payments, people, peoplesoft, phone, photo, please, point, policy, power, present, primary, problem, problems, proceed, process, professional, profile, prompt, protect, protection, proxy, questions, reach, reader, really, reasons, receive, recently, redirect, reed, register, registered, rel, repayment, reply, request, requested, require, reset, resolve, respond, restore, reverse, right, riley, risk, roman, rural, safeguard, safety, said, school, scripts, secure, securely, security, see, send, sent, serious, server, service, services, settings, several, sex, shelter, show, signin, sincerely, site, society, soft, something, standard, statement, still, strong, strongly, subject, super, sure, suspend, suspended, suspension, tab, take, target, team, technical, texts, thank, thanks, think, throat, time, times, title, today, took, toyota, transfers, treatment, trust, try, two, ufa, ultimate, unable, unauthorized, university, unpaid, unusual, update, updated, upgraded, uploads, urgent, use, user, username, using, validate, value, verification, verify, version, via, view, voice, want, way, website, well, wishes, without, work, working, world, would, [Purdue Mascot 1], [Purdue Mascot 2], [Indiana], [West Lafayette], [Purdue University], [Purdue University Portal], and [Purdue University Payroll].