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

Christopher N. Gutierrez, Taegyu Kim, Raffaele Della Corte, *Member, IEEE*, Jeffrey Avery, Dan Goldwasser, Marcello Cinque, and Saurabh Bagchi, *Senior Member, IEEE*

Abstract—Phishing attacks continue to pose a major threat for computer system defenders, often forming the first step in a multi-stage attack. There have been great strides made in phishing detection; however, some phishing emails appear to pass through filters by making simple structural and semantic changes to the messages. We tackle this problem through the use of a machine learning classifier operating on a large corpus of phishing and legitimate emails. We design SAFE-PC (Semi-Automated Feature generation for Phish Classification), a system to extract features, elevating some to higher level features, that are meant to defeat common phishing email detection strategies. To evaluate SAFE-PC, we collect a large corpus of phishing emails from the central IT organization at a tier-1 university. The execution of SAFE-PC on the dataset exposes hitherto unknown insights on phishing campaigns directed at university users. SAFE-PC detects more than 70 percent of the emails that had eluded our production deployment of Sophos, a state-of-the-art email filtering tool. It also outperforms SpamAssassin, a commonly used email filtering tool. We also developed an online version of SAFE-PC, that can be incrementally retrained with new samples. Its detection performance improves with time as new samples are collected, while the time to retrain the classifier stays constant.

Index Terms—Phishing attacks, machine learning security, online learning, university-based phishing campaigns

1 Introduction

Phishing continues to be a viable attack vector despite progress in production-deployed phishing filters. 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]. The prevalence is based on the fact that defensive measures tend to be fragile to change [2], [3], e.g., comprised of rigid regular expressions that detect specific patterns in the text. Another challenge is the transient nature of the domains hosting phishing content, making it difficult to rely on detection techniques that use URLs. In APWG's 4th quarter of 2016 report [4], the reported increase in URL redirection as a technique to obfuscate phishing websites exemplifies the challenges associated with detection using URLs and blacklists. These trends suggest that alternative features within emails must be identified to detect phishing messages accurately.

To efficiently handle the barrage of phishing emails, automated solutions attempt to identify and purge these emails. Examples include BrandProtect's Anti-Phishing E-mail Analysis package, SpamAssassin, and Sophos PureMessage. However, the challenge remains to efficiently identify previously unseen phishing emails. A concrete example from our phishing dataset, shown in Table 2, is that an email about a Blackboard update for a new semester was caught in February 2014, but in May 2014, a similar e-mail about a generic Blackboard update was not caught. These hitherto unseen phishing messages subvert filters that rely on known data, blacklists of known IP addresses, or lexical analysis. Subversion only requires minor structural changes to the phishing message and avoiding the use of blacklisted IP addresses.

A drawback of prior work in phishing detection is that it has been based on an email's surface level text, rather than linguistic patterns and subterfuges. Some examples 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 the use of Natural Language Processing (NLP) techniques for phishing detection proposed in [5], a foundation that our work builds upon.

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 phishing campaigns, such as, the presence of keywords like "account", "expire", a commonly used method feature in phish detection [5]. Second, it performs feature engineering to thwart phishing strategies, such as deliberate misspellings, that disrupt feature extraction. Third, it uses NLP techniques to create "higher level" features, such as, through Named Entity Recognition (NER) and Freebase and through synonym substitution. Finally, SAFE-PC builds a ensemble

- C.N. Gutierrez, T. Kim, D. Goldwasser, and S. Bagchi are with Purdue University, West Lafayette, IN 47907.
  E-mail: {gutier20, tgkim, dgoldwas, sbagchi}@purdue.edu.
- R. Della Corte and M. Cinque are with the Universitá degli Studi di Napoli Federico II, Napoli, NA 80138, Italy.
  E-mail: {raffaele.dellacorte2, macinque}@unina.it.
- J. Avery is with Purdue University, West Lafayette, IN 47907, and also with Northrop Grumman Corporation, Falls Church, VA 22042.
  E-mail: avery0@purdue.edu.

*Manuscript received 24 Nov. 2016; revised 9 Feb. 2018; accepted 12 Feb. 2018. Date of publication 20 Aug. 2018; date of current version 9 Nov. 2018.
(Corresponding author: Saurabh Bagchi.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2018.2864993*

classifier customized to handle the unbalanced nature of e-mail datasets as there are likely to be many more legitimate emails than phishing emails. A online variant of SAFE-PC, which is periodically and incrementally retrained as new samples become available, helps protect against phishings campaigns that evolve over time. The online variant demonstrates that the detection performance improves gradually over time with little increase in training time.

We evaluate SAFE-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. The second dataset comprises 388,264 messages that Sophos did catch. In addition to the phishing datasets, we use legitimate emails from universities, public newsgroups, and publicly available financial emails, thereby keeping the domain of legitimate emails relatively equivalent to the phishing dataset. We train and test on non-overlapping time periods from the datasets, with a time gap between the two, evaluating the ability of SAFE-PC to detect variants of older phishing campaigns that were used in training. We find that SAFE-PC can detect 71 percent of the phishing emails that had been incorrectly classified by Sophos while having a false positive rate of 15 percent. We compared these results with SpamAssassin, an advanced machine learning based tool for detecting spam and phishing email messages. Through experimentation, SpamAssassin was non-competitive for phishing detection, with a detection rate of less than 10 percent. We present an in-depth analysis of emails that were not detected and identify strategies that phishers are utilizing to bypass these filters.

While there is much work on spam and phishing detection, our work is novel in the following ways:

1) We provide a method to extract features from free-form emails. These features help to reduce common subterfuges used in crafting phishing emails. We incorporate synonym analysis, Freebase, and NER into our classifier, an amalgamation that has not been presented before.
2) Prior work in NLP for spam or phishing detection fall short in handling the real-world challenges that our data brings out. Specifically, our work is an improvement of [5] in that we make the feature selection portable and based on empirical observation of evolving datasets.
3) We demonstrate feasibility of online learning, thereby enabling a practical deployment in which new manually flagged emails can incrementally train 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 SAFE-PC performs better than Sophos and SpamAssassin on these datasets. We provide insight into the kinds of deceptive techniques that are difficult for automated techniques, including SAFE-PC, to detect.

2 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 new phishing messages. We provide a brief overview of these categories below. A more comprehensive review is provided in [6].

Rules Base. Filters use a combination of keywords, syntax checks, 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 [7]. This approach shows that keywords can be identified using automation; however, changing the order of keywords or using different ones compared to prior campaigns would bypass this defense.

A simple method of rule based detection is blacklisting IP and email addresses. Compiled blacklists aggregate data from subscriptions of other individual blacklists. Blacklists can monitor submissions to abuse email accounts (accounts that ingest manual submissions of phish/spam by the end user) and extract key attributes from these submissions that can be used to identify phish. This primarily manual process of identifying and submitting phishing emails is taxing for administrators.

Classifiers. Classifiers detect phishing emails using statistics and pattern matching. Prior phishing emails are used to train the classifier, which is used to new emails. Prior work used machine learning to classify emails and websites as phishing/legitimate [8], [9], [10].

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

Combining Techniques. An example of combining techniques is employed by the tier-1 research university whose data we are using. The Sophos PureMessage filter [11] utilizes a suite of methods to combat malicious emails but does not distinguish between phishing and spam emails. PureMessage employs a malware scanner to detect malware and utilizes blacklists to stop messages from untrustworthy IP addresses. PureMessage allows IT administrators to define their own rules to block messages that are a threat to their systems. The university utilizes roughly 6,200 rules to combat malicious emails. Rules manually created by developers based on phishing emails seen in the past must constantly be updated. The process requires a security expert 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 [12], which can create a regular expression that can be used as a rule.

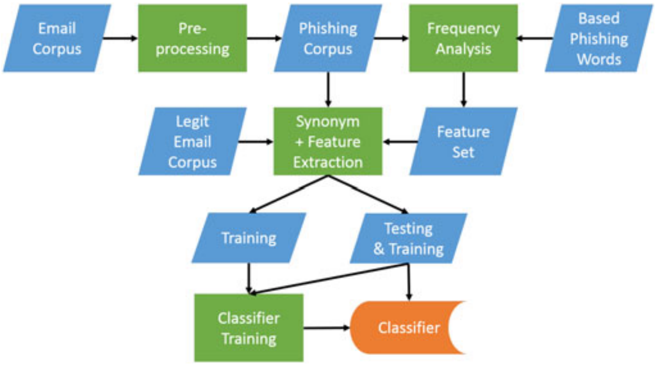


Fig. 1. Workflow of SAFE-PC for the training phase.

The shortcoming of these techniques in practice is they 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 corpus of phishing emails in preparation for training a classifier. The first step is to analyze the corpus to create a rich set of features. The next step transforms each email by *projecting* the raw email into a *feature vector* to remove noise that is inherent in any free-form media, like e-mail. Next, a classifier is trained on all unique feature vectors. Finally, the classifier is used to identify production emails as legitimate or phish. SAFE-PC is capable of updating the classifier (without complete retraining) as users report new phishing emails. Fig. 1 shows the workflow of SAFE-PC for the training phase.

3.1 Feature Generation

A key component of SAFE-PC is the semi-automatic creation of rich features to be used by the classifier. 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 which applies NLP techniques to phishing detection [5], [13]. Our work demonstrates the effectiveness of a number of these features on a much larger and more current data set compared to the dataset used in prior work.

SAFE-PC's feature generation is designed to capture deceptive semantics found in phishing emails. Certain words such as "urgent" and "account" are commonly found in phishing campaigns [5]. These are obtained from prior papers on phishing detection and used in SAFE-PC (*Based Phishing Words* in Fig. 1). The set of common phishing words comprises feature set (i).

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

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, ignoring stop words. Only the words appearing in at least 1 percent of all emails within the training set are selected. Feature set (iii) consists of words collected from the frequency analysis of phishing emails.

Similar to the approaches found in [5], 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 [14]. 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 in 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) [15] to recognize different entity types, such as persons, locations, organizations, within the text. NER has been applied to phishing research in prior work, such as [5], [13]. For a given entity, we query the FreeBase knowledge graph [16] to determine its domain. Freebase returns potential categories the entity belongs to and provides a score of how relevant each category is. We create "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 useful in detecting phishing emails as one phishing technique is to create new phishing emails by replacing one organization's name with another organization from the same domain, e.g., Citibank/Chase Bank. Feature set (iv) consists of one binary feature for each super category. It should be noted that super categories have been created manually by inspecting the categories returned by FreeBase. Similar categories have been grouped together 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 one; 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 representation, providing greater resiliency to fluctuating deceptive language in phishing emails.

Feature set (v) considers key structural components found within an email. These features were derived from empirical experimentation and observing some phishing emails use special encoding or formatting. Specifically, the features consider the number of embedded links (e.g., href

TABLE 1
Number of Features Used in SAFE-PC

Feature set	Number of features
(i) Common phish	48 × 2 (subject & body)
(ii) Our univ words	16 × 2 (subject & body)
(iii) Corpus common words	325 × 2 (subject & body)
(iv) Super categories	3
(v) Structural features	25
Total	806

tags), the total number of HTML tags in the body, 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. Table 1 shows the number of features for each category, which are detailed in the appendix. Despite the simplistic approach, our experimental evaluation shows that we can identify some previously undetected phishing emails. Several other approaches, e.g., 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 methods can be explored in future work, which may improve the obtained results.

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. Similar phishing emails are *projected* onto the same point in our feature vector space, which has much lower dimensionality (806 features) 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.

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) [17]. It is well suited to our problem where the two classes—phishing and legitimate—are severely imbalanced. Based on conversations with phishing detection experts at the university, we can safely assume that the number of phishing emails received makes up 1-10 percent of all emails, depending on the virulence of phishing campaigns. We preserve such an imbalance between the legitimate and phishing data in the training set and use a 90:10 percent ratio for our experimentation. 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, decision trees

TABLE 2
Most Popular Campaigns Separated by Month

Month	Uncaught	Uncaught Subjects	Caught	Caught subjects
7-2013	16,187	Bank account needs updating using website given to validate | Bank account access suspended, validate yourself using link	10,399	Bank account needs to be updated, click here...
12-2013	5	Click to read a new message	1	Account is expiring, click to renew
1-2014	282	Click here to view a product | Account needs upgrade, click here to read more	134	Recruited to evaluate western union banks and given an assignment to evaluate which includes money sent to them | Login to confirm identity
2-2014	14,862	New inquiry, agreed to subscribe for membership | Bank statement is attached | Bank account updated, verify and update via secure attachment | Received inquiry from verified member, to see the product press link	3,469	Blackboard received an update for new semester, click link...
3-2014	2,287	Bank card will expire, click here... | Phrase or name and click link	15,085	gibberish and a link | Bank account will expire, click here to renew/validate
4-2014	2,303	Verify online account	23,906	gibberish and links
5-2014	2,863	blackboard updated, click on link	338,768	gibberish and links | phrase or fraction of a sentence and a link
7-2014	144	Husband died, need to distribute money, reply with personal info | incoming email pending delivery due to upgraded database, use link... | Out of the office	1,035	message you sent is being held... click here to verify yourself or deactivation | Have money to give from dead husband, provide identifying information...
8-2014	4,981	Message could not be delivered | Want to order products, need to speak in person | Document uploaded to google docs, click here... | Update prices in attached document..	10,478	1 new message in blackboard, sign in | no subject, only a link

Individual campaigns are divided within each month by a |.

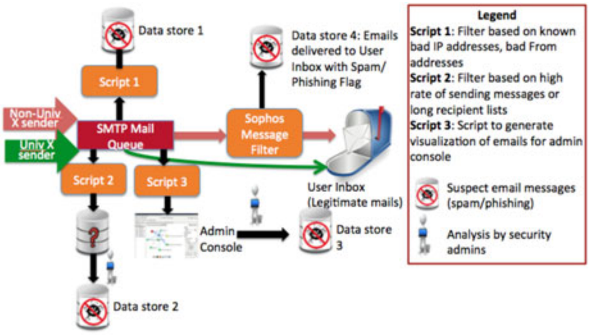

Data store 1
Script 1
Data store 4: Emails delivered to User Inbox with Spam/ Phishing Flag
Non-Univ X sender
Univ X sender
SMTP Mail Queue
Sophos Message Filter
Script 2
Script 3
User Inbox (Legitimate mails)
Admin Console
Data store 3
Data store 2
Legend
Script 1: Filter based on known bad IP addresses, bad From addresses
Script 2: Filter based on high rate of sending messages or long recipient lists
Script 3: Script to generate visualization of emails for admin console
Suspect email messages (spam/phishing)
Analysis by security admins

Fig. 2. Current production dataflow of emails at a tier-1 research institution.

and perceptions classifiers. At the end of training, the final verdict is a weighted vote from all the learners.

3.4 Online Learning

As new data is collected and observed, classification models must be updated and 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 (e.g., once at the end of a day). Online learning aims to improve the model without 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 minority phishing class and the majority benign class. Unless it is specifically designed for quick retraining, from an operational standpoint, it is undesirable for a machine learning phishing detection system to be retrained with the entire phishing corpus plus the newly observed samples. As the phishing corpus increases in size, the latency to train the classifier becomes impracticable. Therefore, we take a more operationally feasible algorithm whereby the model is updated for a *group of new samples*.

After preliminary experimentation with several boosting algorithms and different weak learners, we determined that online RUSboosting with Gaussian Naive Bayes learners produce the best classification performance. The Python implementation of the online RUSBoost algorithm is driven from work presented in [18]. By 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 improving in its ability to detect new forms of phishing campaigns by incrementally retraining the classifier.

4 Datasets for Our Evaluation

In this section, we describe our datasets and the processing that is needed to transform free-form data like email into a form suitable for rigorous evaluation.

4.1 Collection Methodology

We show the dataset collection process in Fig. 2, which represents the current operational flow used by the 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 is reasonably accurate. The analysts are IT staff in a dedicated sub-unit of the central IT organization of Purdue University, namely, the "Messaging Systems Group", who manage and monitor the email servers for abuse. The IT analysts examine suspicious emails identified by any one of several automated detectors (e.g., sending a high volume of emails) or reported from the end user. However, we recognize that the decision to label an email as phish or legit is ultimately empirical. There are two separate, but overlapping, flows for messages; those that originate within our university ("Univ X sender") and those that originate outside ("Non-Univ X sender"). Scripts 1, 2 and 3 are custom scripts written by the university messaging security personnel and help to identify suspicious emails from the flows; suspicious emails are stored in the Data stores 1, 2 and 3, respectively. Emails in the Data stores 1, 2 and 3 are never seen by Sophos in the current operational flow—they are flagged by the custom scripts or via visual inspection by the security admins. We take these data stores and analyze them using Sophos. The subset of data stores 1, 2 and 3 that is caught by Sophos forms *one part* of our *Caught Dataset*, while data store 4, which comprises non-university emails that are caught by Sophos, forms the other part of our Caught Dataset. Data store 4 could have personally sensitive information since these are delivered to the individual user's Inbox, albeit with a flag indicating that the mail is suspect. The subset of data stores 1, 2 and 3 that is not caught by Sophos forms our *Uncaught Dataset*.

4.2 Datasets 1 & 2: Phishing Emails

The Uncaught dataset (Dataset 1) is composed of 37,606 emails, each with a header, subject line, and message body. This set includes 2,248 unique emails after feature space projection. The emails in the Uncaught dataset were *not* detected by Sophos PureMessage. Emails in this dataset span from July 9, 2013 to May 23, 2014.

The Caught dataset (Dataset 2) is composed of 388,264 emails, each with the 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 emails 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.

About half of the legitimate emails were collected from public *listserv* hosted on `.edu` domains from universities in

North America. The motivation for using these is that they are close to the spam emails sent to a university audience. The content includes university seminar announcements, academic listservs dealing with research, social activities, and a Mozilla bug reporting list. Unfortunately, we were unable to use samples of live university emails during the same period of the phishing dataset because of privacy reasons. The other half of legitimate emails are from Jeb Bush's publicly released emails during his time as governor of Florida, with only the subset related to financial transactions. The motivation for using these emails is they consist of real financial discussions that fit with our university context and can be considered close to phishing emails related to financial information. Prior work in phishing classification has used the Enron email set [13] as benign set. However, we wished to have emails that are closer to the phishing ones sent to our target university audience, and so we did not use this email set. The final mix of the unique legitimate emails composing dataset 3 is Jeb Bush Finance (53 percent), Listservs on Cytometry (33 percent), Mozilla (12 percent), Vision list (4 percent), and CompSci Colloquium (1 percent).

4.4 Data Cleansing

We removed duplicates from each dataset by using the MD5 hash of the email body. Removing duplicates was necessary since we noticed that large portions of emails are repeated because of large number of recipients. This is expected because of the nature of phishing emails where a single crafted mail is sent to a large group of recipients to maximize the return on the investment by the miscreants. Our preliminary analysis showed a performance improvement by removing duplicates for both training time and accuracy. While not conventional, others have removed duplicates for data cleansing [19], [20]. Based on empirical data analysis, we identified many emails in each dataset were spam. We implemented a simple filter that removes common spam emails. Examples of spam include emails with keywords such as "cialis" or "viagra." We remove these emails by using a regular expression that looks for specific words , see the appendix for details. 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 letters to misspell words to subvert detection methods deliberately. Our approach analyzes the body of each e-mail and executes some steps to sanitize the emails, i.e., (i) remove punctuation, (ii) substitute 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", "$ → S".

Whitespace that appears within words is addressed as follows. Given an email body, our algorithm verifies if each word is English with PyEnchant.[1] Next, the algorithm concatenates characters and verifies if the obtained word is English. 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 one and replaces the word in the body. If the word is Non-English, it is left alone; the algorithm continues to the next word and terminates when no other words are in the body. 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 misspelled word in the text.

4.5 Baseline Comparison: Sophos PureMessage

Our university's central IT organization uses Sophos PureMessage [11] on its Microsoft Exchange servers. This is an enterprise-grade email scanning software, with modules for filtering spam, phishing, and malware. Once an email is scanned by all the rules, those that are matched in the email are summed; 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 percent, i.e., its default value, 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 emails that were *not* caught by Sophos. The second experiment evaluates if SAFE-PC is still able to detect the phishing emails that *were* caught by Sophos. The third experiment tests the online learning mode of SAFE-PC: the classifier is incrementally retrained as new phishing messages are collected over time. We note that this third experiment performs cross-validation in 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 Naïve Bayes (RG5), 500 Decision Tree (RD500), and 100 perceptrons (RP100). The fourth and fifth experiments evaluate the performance of SpamAssassin on the same datasets to identify its capability to distinguish between phish and legitimate emails. The final experiment measures the runtime performance of the tools. For each classifier and experiment, we use an imbalanced dataset to train and test the classifiers. This imbalanced dataset comprises of 90 percent legitimate emails and 10 percent phishing emails.

5.1 Phishing Email Topics

Phishing attackers continuously change topics to deceive users. An example is to send emails about Blackboard toward the beginning/end of semesters. Blackboard is used by students to check course-related information. Table 2

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

[2] Levenshtein distance an edit count to transform a source into the target string. We used the python-Levenshtein package to compute that distance (https://github.com/ztane/python-Levenshtein/).

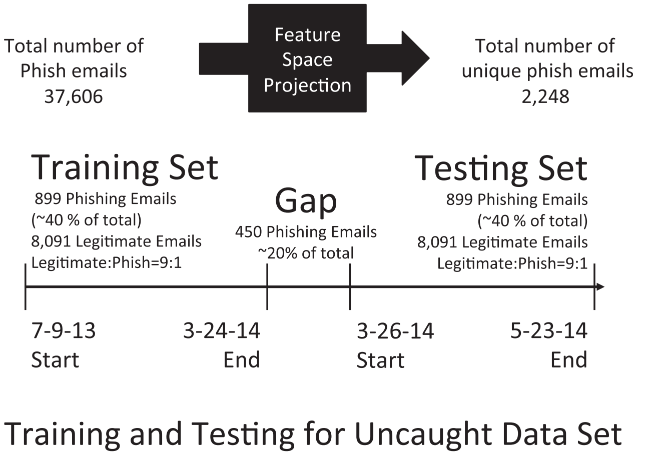


Fig. 3. Training and testing datasets for uncaught i.e., the emails that were *not* detected as phishing by Sophos.

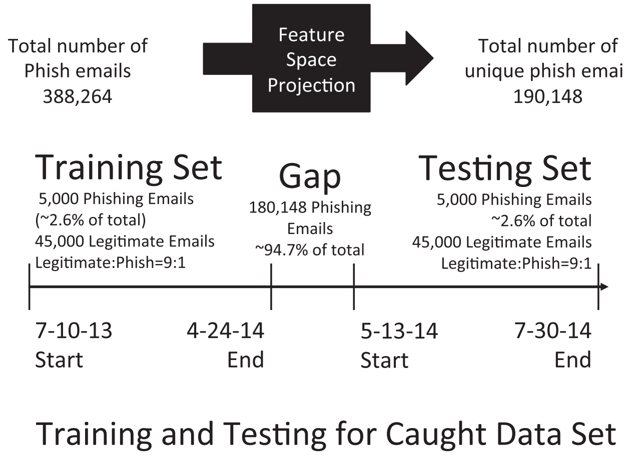


Fig. 4. The training and testing datasets for caught, i.e., the emails that were detected as phishing by Sophos.

details popular phishing emails each month, along with the number of phishing emails collected. These values are further separated into the number of phishing emails that were caught and not caught. Low observations in December and July could be due to an incomplete dataset or a genuinely low 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. We find the financial institution and university features frequently appear in both the caught and uncaught datasets. The presence of a financial institution's name is an expected observation because attackers are motivated by monetary gains. The most popular financial institutions in our dataset are RHB, RBS, Western Union, Among university phishing campaigns, Blackboard was by far the most prevalent. The most popular IT providers are AOL and Google, the latter through Google Docs.

Another observation is the latency between the publicly reported phishing attack and when a significant volume of such attack appears in our datasets. The lag was typically about three weeks while some phishing campaigns had longer lag times. For example, a phishing attack involving Google Docs was reported [21] in March 2014; it did not appear 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 e-mail was observed in volume in February 2014 and was caught by the Sophos filter. However, we saw in May 2014, a very similar email about Blackboard was undetected by Sophos. The message differed slightly from the original as can be seen in the excerpts below, which exposes the undesirable specificity of some filters. Instructively when a similar Blackboard phishing message appeared in volume in August 2014, it was caught.

Caught 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 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

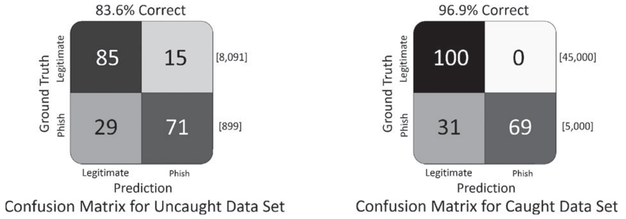
5.2 Experiment 1: Uncaught Phish

We evaluate SAFE-PC to detect phishing emails that were *not* caught by Sophos. The raw detection rate should be considered in conjunction with the fact that SAFE-PC creates its features using semi-automated methods, thus making it easier to update existing features as new phishing campaigns emerge.

We segment our dataset 1 consisting of 37,606 emails into training and test datasets as shown in Fig. 3. 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 some emails are identical in our feature space, possibly indicating changes being made by the phishers to deceive the end user.

By separating the training and the test emails, we ensure the data in the training set occurs at an earlier date compared to the emails in the test dataset and further, introduce a gap in time between these two sub-datasets. The gap reflects the latency of collecting, training, and deploying SAFE-PC on a real system. We keep the training and test data sizes to be equal.

Fig. 5a shows the confusion matrix from experiment 1 using RUSBoost with 5 Gaussian Naïve Bayes (GNB) classifiers (denoted as ”RG5”). We find that 71 percent of phishing emails missed by Sophos were correctly detected with a 15 percent false positive rate. We argue that the uncaught dataset consists of the most challenging phishing emails, which accounts for the lower than ideal results. 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. These emails simply do not contain enough analyzable content, e.g., having a single URL with a line of text around it. We found through manual inspection that it is challenging to determine the nature of the message without visiting the URL or understanding the context. It should also be noted that our filtering process is not perfect;



(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 Bayesian classifiers being used as weak learners.

some malicious or phishing emails may get included in the legitimate dataset. Thus, noise in our dataset could also contribute to misclassifications. Another item to note is that SAFE-PC does not analyze the structure of URLs to identify deceptive naming conventions (e.g., amazon.hackers.org rather than amazon.com). Features extracted from URLs could help reduce the high false positive rates for the uncaught dataset. This approach is complementary to our methodology. 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 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 acceptable, but receiving an unsolicited email with a URL from a stranger is suspicious. An automated technique will need to collect 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 e-mail to your inbox and then click!*. For the few legitimate emails classified as phish, we observed that the email only had a few features. The features that were present (e.g., “thanks”, “continue”, the presence of a URL) suggested to SAFE-PC that the email is phishing.

Potentially, our classifier could suffer from overfitting, which could lead to misclassification if future samples deviate slightly. We evaluate our classifier with the training dataset, a practical and widely used method for determining overfitting [22]. The true positive result of legitimate email detection using the training data is 89.9 percent. The true positive result of phishing email detection rate is 66.4 percent and combined correct prediction rate 87.6 percent. The prediction rate for the training set is only a 4 percent difference compared to the testing set evaluation. We conclude that our classifier is not overfitting.

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 of training and test sets, along with the date ranges, is shown in Fig. 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 emails, some potentially with new phishing campaigns. The results are shown in Fig. 5b. SAFE-PC is able to detect 69 percent of the emails that Sophos caught with the RG5 configuration. SAFE-PC flagged 0 percent of legitimate email messages as phishing.

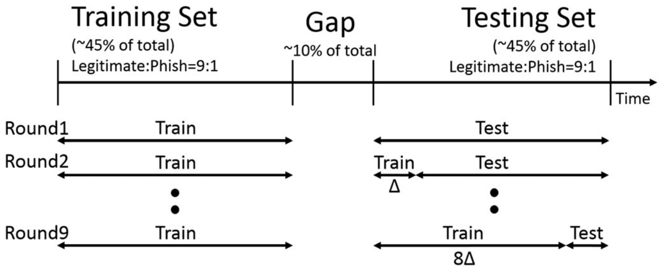


Fig. 6. SAFE-PC-Online: Training and Testing sets at each round.

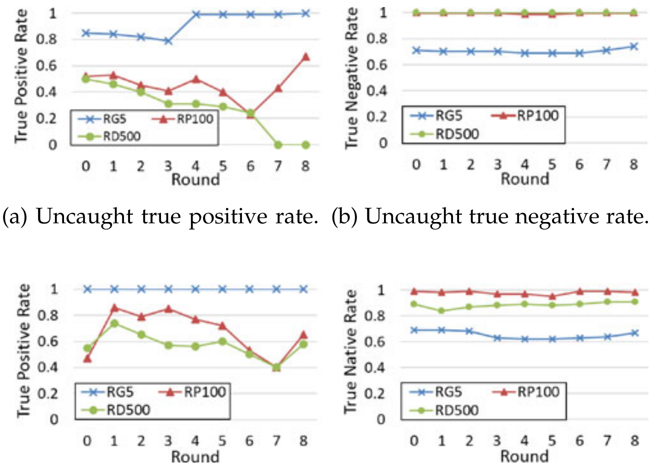
For the caught false negative messages, the emails just contain gibberish and a URL. Similarly to what was shown in the uncaught experiments, the analysis of emails having cryptic sentences and a URL is not trivial; this type of emails does not contain enough analyzable content, generating vectors with very few representative features that led our classifier to incorrectly label the emails. A closer look into the misclassified messages revealed that also their manual classification is challenging without analyzing and/or visiting the URL.

Next, we consider the possibility of overfitting the data. As we did for the uncaught experiments, we trained and tested on the same dataset and compared the prediction rate with a disjoint training and testing dataset (as we had done for our regular experiments). When using the same training and testing dataset, the correct legitimate prediction rate is 99.7 percent, correct phishing prediction rate is 73.6 percent, and overall accuracy is 97.1 percent. When compared to that classifier trained under disjoint training and testing sets, the overall difference is only 0.2 percent. We thus conclude that our classifier is not overfitting.

5.4 SAFE-PC-Online

Online learning incrementally retrains the classifier for new observations. Fig. 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. 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 percent (5,000) of total samples including training, testing, and time gap samples. As the previous experiments, we evaluate the online learning algorithms, separately for the uncaught and caught datasets.

Note that the analysis of our online learning scheme is cross-validated realistically. The online learning scheme receives new phishing samples periodically. The testing set



(a) Uncaught true positive rate. (b) Uncaught true negative rate.

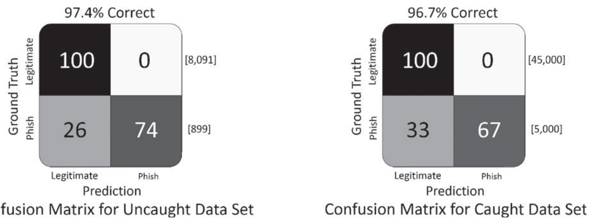
(c) Caught true positive rate. (d) Caught true negative rate.

Fig. 7. SAFE-PC-Online test result at each round.

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 sample in the training set. There may be temporal dependencies because of the deceptive semantics that campaigns use over a period. Practitioners collect and train classifiers periodically, and our analysis 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 experimentation, we evaluated several classification algorithms to work well with RUSBoost. Our initial experiment explored 15 different configurations of the online classifier, corresponding to 3 different ensemble models—RUSBoost, AdaC2, and Cost-sensitive Boosting—and 5 different weak learners—Perceptron, Naïve Bayes, Decision Tree, Multi-Layered Perceptron, and Naïve 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 [18]. We also applied parameters to our classifiers. In terms of ensemble model, we use RUSBoost3 and 0.1 for C_N and 1.0 for C_P. For weak learners, we applied the zero bias parameter to Perceptron and 10 learning rate parameter to Multi-Layered Perceptron. We only show three representative classifier configurations in Fig. 7. In this figure, R denotes RUSBoost, G Gaussian naïve 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. Therefore, we choose the fastest executing boosting algorithm, RUSBoost. However, the choice of the weak learner does have an impact. Perceptron and decision tree perform the best (multi-layer perceptron gives incremental improvement) in detecting phishing emails, while naïve Bayes gives a reasonably good performance. However, with regard to classifying legitimate emails, both perceptron and decision tree perform poorly, while naïve Bayes performs almost perfectly. RUSBoost with decision tree and perceptron learners gave high false positives with benign emails—about 30 percent for the uncaught dataset and 50 percent 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 Fig. 7, we see the poor result even with 500 decision trees. For SAFE-PC-Batch, the decision tree weak learner gives competitive detection performance compared to naiive 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 naïve Bayes, even using a small number of weak learners gives good performance—five weak learners in both datasets seem sufficient. The small number of weak learners has an execution time performance advantage because the learners have to be executed sequentially. RUSBoost with 5 naïve Bayes classifiers (RG5) misclassifies few legitimate emails and for the phishing emails at the last round, it detects 74 percent for the uncaught dataset and 67 percent for the caught dataset. Considering that in this domain, it is critically important to have a low false positive rate, we believe naïve Bayes is a suitable weak learner for online learning in SAFE-PC.



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

Fig. 8. Confusion matrices from the evaluation of SAFE-PC-Online. Train = 8Δ, Test = Δ.

We present the confusion matrix for the SAFE-PC-Online results separately for the uncaught and caught datasets in Fig. 8. For the confusion matrix, we take the last round of online learning, i.e., one where the last 5 percent of the dataset is used for testing. We find that with the online learning, as SAFE-PC-Online learns new legitimate emails, in this final round, it does not misclassify any legitimate email. For the uncaught dataset, the phishing email detection is 74 percent, while for the caught dataset, it can detect about two-thirds of them. Expectedly, uncaught performance is improved as SAFE-PC-Online is trained with more samples and it learns to model evolving phishing campaigns. The online algorithm performs better for the uncaught phishing emails than the caught ones. This can be explained by the fact that as new training data is fed into the classifier, the uncaught emails get progressively easier to detect for SAFE-PC-Online.

We observe a large increase in the true positive rate between rounds 3 and 4. According to our analysis, phishing samples at round 3 have distinctive feature vectors which are hard to accurately predict, likely corresponding to some new phishing campaigns. However, after SAFE-PC-Online is trained with these new samples, the performance improves. Therefore, our SAFE-PC-Online is not only

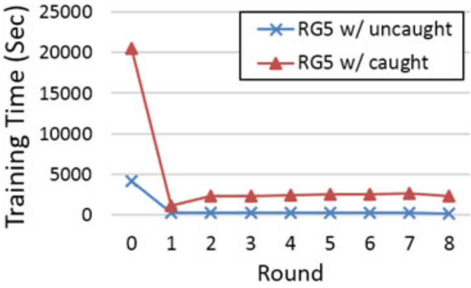


Fig. 9. Time to train SAFE-PC-Online as we increase the number of rounds. The time to train each round is approximately constant overall.

TABLE 3
Training Set and Testing Set for Uncaught and Caught Experiments (Unique Emails) for SpamAssassin

	Uncaught		Caught	
	Training	Testing	Training	Testing
Legitimate	8,091	8,091	45,000	45,000
Phishing	899	899	5,000	5,000

superior to SpamAssassin [23], 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 check if our online classifier is overfitted. To perform it, we apply the identical scheme used in testing dataset prediction. Regarding uncaught prediction, the average uncaught prediction rate of the training dataset for all rounds is 88.6 percent, and that of the testing dataset for all rounds is 89.6 percent. On the other hand, the average caught prediction rate of the training dataset for all rounds is 96.7 percent, and the testing dataset for all rounds is 96.5 percent. 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 to train SAFE-PC-Online at each round and show the result in Fig. 9. Initially, there is a large fixed cost for the training, on 45 percent of the dataset. However, after that, as each additional Δ worth of samples is added, the training is incremental, and not proportional to the total amount of training data. 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 to do the incremental retraining.

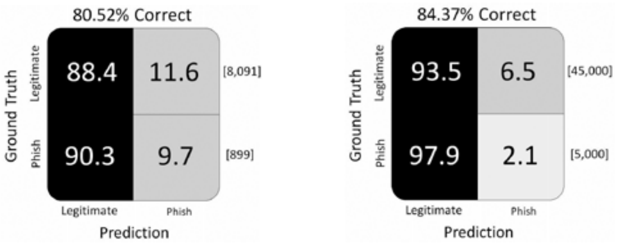
5.5 SpamAssassin Experiments

SpamAssassin [23] is a widely-deployed open-source spam filter. It allows identifying 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 [24], [25], [26]. SpamAssassin is the only open source and widely-used 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 a Bayesian classifier to identify spam. The rules leverage regular expressions to check the header and body of each email, e.g., it checks the gap time between the *Date:* and the *Received:* header fields, 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 catalog), if the email contains some blacklisted link, etc. The Bayesian classifier tries to identify spam emails. 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. If the obtained scores are higher than a configured threshold, the analyzed email is labeled as spam. We used the default threshold (i.e., 5.0) and rules, and enabled both 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 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. Fig. 10a shows the confusion matrix for the uncaught emails.

SpamAssassin correctly detected 9.7 percent of the phishing emails showing that most of the phishing emails bypassed its tests and reached the inbox of the users. On the other hand, 11.6 percent 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*.



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

Fig. 10. Confusion matrices from the evaluation of SpamAssassin.

For the caught experiment with SpamAssassin, we obtained the confusion matrix shown in Fig. 10b. We found that only 2.1 percent of caught phishing emails were correctly detected, while 6.5 percent of the legitimate email messages were mistakenly flagged. A closer inspection of the misclassified emails indicated the same results as for the uncaught dataset. 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.

5.6 Performance

The time to train the SAFE-PC's classifier is 418 ms on average per email and the time for prediction is 296 ms per email, on an 8-core VM (3.1 GHz CPU and 12 GB 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 them 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 benign); a large part of this cost is the remote lookup involved in its "online test". Even if SpamAssassin can train faster, it does not use online training, so the entire corpus is required to retrain. Training speed of SAFE-PC may be superior to SpamAssassin when continual training is required. Further, machine learning algorithms of SAFE-PC are implemented using python; therefore, if we use optimized codes based on C, it would be much faster.

6 Discussion

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

Defeating Feature Extraction. A logical question that arises is how would an adversary defeat SAFE-PC if she were aware of the algorithm. 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 aggregate emails from the same sender to the same receiver within a short time window into one thread, and treat them as a single data point. Since we extract many text-based features, the adversary can embed images with text. 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.

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 [14], words, and phrases in phishing emails can be mixed and matched 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 some studies [2], [3]. 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 with tweaks to phishing messages.

Prior work that is similar to ours is [5], where natural language processing analyzes emails to test if they are benign, phish or unknown. This serves a starting point for our work. We have already pointed out, in Section 3.1, the shortcomings of their approach. Also, the approach allows inferring the *intention* of each email received by a user by analyzing both sent and received emails from the user, i.e., the context. Therefore, it would be impossible to perform that kind of analysis for any medium to a large-sized community, like our university, which has a large number of end users. The approach is likely to run into scalability and privacy bottlenecks. Verma et al. employs a t-test and WordNet to select features present in phishing emails and then use these features for detection [13]. 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. 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.

Another early and relevant work is [24], where the authors present an automated solution to decide whether some communication is deceptive. While the approach shares some similarities with SAFE-PC, it does not consider a wide set of features (10 features as opposed to our 806 features) and does not analyze a large number of phishing messages (860 emails). We believe our approach of using synonyms makes developing phishing campaigns to bypass SAFE-PC more difficult.

Prior work has applied online learning to classify phishing emails [27]. This work uses a technique similar to our proposal, 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. In addition, this work lacks 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 [28]. 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 emails for the classification; the limited number of phishing and legitimate emails mines the generality of the technique.

8 Conclusions

This paper proposes a methodology to combat phishing emails. We developed a system called SAFE-PC for detecting new phishing campaigns, which are evolved from prior ones. SAFE-PC uses real-world phishing and legitimate email datasets from the central IT organization of a tier-1 research university, a total of 425 K phishing and 158 K legitimate emails. It extracts features from each message's header and body, imbued with an understanding of structures of phishing emails. Then it applies a RUSBoost classifier using the phishing and legitimate emails. We perform a thorough evaluation using the real-world corpus and compare SAFE-PC to the 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.

Appendix
Preprocessing and Feature Details

Filters Words for Spam. 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 SAFE-PC. Below are the words (and their associated synonyms) used in SAFE-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, dari, 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, inconvenience, 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].

Acknowledgments

The authors would like to thank Paul Wood (Purdue), Keith McDermott and Brian Berndt (Information Technology at

Purdue - ITaP), and Jonathan Fulkerson (Northrop Grumman) for their valuable help for this paper. This material is in part supported by the US National Science Foundation (Grant Number CNS-1548114) and Northrop Grumman through their Cybersecurity Research Consortium. Any findings and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] G. Aaron and R. Rasmussen, "Global phishing survey: Trends and domain name use in 2016," 2016, http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf

[2] B. Gupta, A. Tewari, A. K. Jain, and D. P. Agrawal, "Fighting against phishing attacks: State of the art and future challenges," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3629–3654, 2017.

[3] A. Aleroud and L. Zhou, "Phishing environments, techniques, and countermeasures: A survey," *Comput. Secur.*, vol. 68, pp. 160–196, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404817300810

[4] G. Aaron and R. Rasmussen, "Phishing activity trends report: 4th quarter 2016," 2014, https://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf

[5] R. Verma, N. Shashidhar, and N. Hossain, "Detecting phishing emails the natural language way," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2012, pp. 824–841.

[6] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, Oct.–Dec. 2013.

[7] G. Park and J. M. Taylor, "Using syntactic features for phishing detection," arXiv:1506.00037, 2015.

[8] R. Dazeley, J. L. Yearwood, B. H. Kang, and A. V. Kelarev, "Consensus clustering and supervised classification for profiling phishing emails in internet commerce security," in *Knowledge Management and Acquisition for Smart Systems and Services*. Berlin, Germany: Springer, 2010, pp. 235–246.

[9] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *Proc. 17th Annu. Netw. Distrib. Syst. Secur. Conf.*, 2010, pp. 1–14.

[10] G. Park, "Text-based phishing detection using a simulation model," Ph.D. dissertation, Comput. Inf. Technol., Purdue Univ. West Lafayette, IN, USA, 2013.

[11] Email security and anti spam protection for data loss prevention. [Online]. Available: http://www.sophos.com/en-us/products/email.aspx, Sophos, Last accessed: Sep. 13, 2018.

[12] txt2regex: The console regular expression wizard. [Online]. Available: http://aurelio.net/projects/txt2regex/, Aurelio, Last accessed: Sep. 13, 2018.

[13] R. Verma and N. Hossain, "Semantic feature selection for text with application to phishing email detection," in *Proc. Int. Conf. Inf. Secur. Cryptology*, 2013, pp. 455–468.

[14] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2009.

[15] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meet. Assoc. Comput. Linguistics: Syst. Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[16] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 1247–1250. [Online]. Available: http://doi.acm.org/10.1145/1376616.1376746

[17] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "RUSBoost: Improving classification performance when training data is skewed," in *19th Int. Conf. Pattern Recognit.*, Tampa, FL, 2008, pp. 1–4.

[18] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3353–3366, Dec. 2016.

[19] M. Sabhnani, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context," in *Proc. Int. Conf. Mach. Learn.: Models Technol. Appl.*, 2003, pp. 209–215.

[20] M. Dredze, R. Gevaryahu, and A. Elias-Bachrach, "Learning fast classifiers for image spam," in *Proc. Conf. Email Anti-Spam*, 2007, pp. 2007–487.

[21] Sophos Inc., "Sophos pure message," https://www.sophos.com/en-us/products/puremessage.aspx, Last accessed: Sep. 13, 2018.

[22] D. D. Gutierrez, *Machine Learning and Data Science: An Introduction to Statistical Learning Methods with R*. Basking Ridge, NJ, USA: Technics Publications, 2015.

[23] Spamassassin. (2015). [Online]. Available: http://spamassassin.apache.org/, Apache

[24] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 649–656.

[25] D. L. Cook, V. K. Gurbani, and M. Daniluk, "Phishwish: A stateless phishing filter using minimal rules," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2008, pp. 182–186.

[26] T. Ronda, S. Saroiu, and A. Wolman, "Itrustpage: A user-assisted anti-phishing tool," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 4, pp. 261–272, Apr. 2008. [Online]. Available: http://doi.acm.org/10.1145/1357010.1352620

[27] G. L'Huillier, R. Weber, and N. Figueroa, "Online phishing classification using adversarial data mining and signaling games," in *Proc. ACM SIGKDD Workshop CyberSecur. Intell. Informat.*, 2009, pp. 33–42.

[28] M. Chandrasekaran, K. Narayanan, and S. Upadhyaya, "Phishing email detection based on structural properties," in *Proc. NYS Cyber Secur. Conf.*, 2006, pp. 1–7.

Christoper N. Gutierrez received the PhD degree in computer science from Purdue University, in 2017 with a focus on systems security and deception. He is currently a security solutions research scientist with Intel Labs in Hillsboro, Oregon, where he is developing solutions to protect automotive systems.

Taegyu Kim received the BS degree in electronics and communications engineering from Kwangwoon University, in 2013 and the MS degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), in 2015. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, Purdue University. His research interests include cyber-physical system security and mobile security.

Raffaele Della Corte (S'15-M'17) received the BS, MS, and PhD degrees in computer engineering from the Federico II University of Naples, Italy, in 2009, 2012, and 2016, respectively. He is currently a postdoctoral researcher with the Department of Electrical Engineering and Information Technologies, Federico II University of Naples. His research interests include data-driven failure analysis, on-line monitoring of software systems, and security. He is a member of the IEEE.

Jeffrey Avery received the bachelor's, master's, and PhD degrees all in computer science. He has also completed a number of successful internships, including stints with Army Research Laboratory, Johns Hopkins Applied Physics Laboratory, McAfee, Inc., and Sypris Electronics, Inc. His research interests include network security, anti-phishing, and deception. He is currently a software engineer in the Future Technical Leaders program at Northrop Grumman.

Dan Goldwasser received the PhD degree from the University of Illinois, in 2012. He is an assistant professor with the Department of Computer Science, Purdue University. His interests include the area of natural language processing and applications of machine learning for textual data. His work focuses on real world applications, such as machine comprehension and discourse analysis, which require machine learning algorithms capable of connecting raw textual data, world-knowledge, and dynamic real-world behaviors.

Marcello Cinque received the PhD degree in computer engineering from the University of Naples, Italy, in 2006. Currently, he is an associate professor with the Department of Electrical Engineering and Information Technology, University of Naples Federico II. He is/has been chair and/or TPC member of several technical conferences/workshops on dependable systems, including IEEE DSN, EDCC, and DEPEND. His interests include dependability assessment of critical systems, model-based verification, and log-based failure and misuse analysis.

Saurabh Bagchi is a professor with the School of Electrical and Computer Engineering and the Department of Computer Science, Purdue University in West Lafayette, Indiana. He is the founding director of a university-wide resilience center at Purdue called CRISP (2017-present). He serves on the Steering Committee of the premier conference in the field of dependability, DSN. He is an ACM distinguished scientist, in 2013, a senior member of IEEE, in 2007 and of ACM, in 2009, a distinguished speaker for ACM, in 2012, and an IMPACT Faculty fellow at Purdue. He is the recipient of an IBM Faculty Award, in 2014, a Google Faculty Award, in 2015, and the AT&T Labs VURI Award, in 2016. He was elected to the IEEE Computer Society Board of Governors for the 2017-2019 term. His research interests include distributed systems and dependable computing.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.