That Ain't You: Blocking Spearphishing Through Behavioral Modelling

Gianluca Stringhini and Olivier Thonnard

University College London, Amadeus
g.stringhini@ucl.ac.uk, olivier.thonnard@amadeus.com

Abstract. One of the ways in which attackers steal sensitive information from corporations is by sending *spearphishing* emails. A typical spearphishing email appears to be sent by one of the victim's coworkers or business partners, but has instead been crafted by the attacker. A particularly insidious type of spearphishing emails are the ones that do not only claim to be written by a certain person, but are also sent by that person's email account, which has been compromised. Spearphishing emails are very dangerous for companies, because they can be the starting point to a more sophisticated attack or cause intellectual property theft, and lead to high financial losses. Currently, there are no effective systems to protect users against such threats. Existing systems leverage adaptations of anti-spam techniques. However, these techniques are often inadequate to detect spearphishing attacks. The reason is that spearphishing has very different characteristics from spam and even traditional phishing. To fight the spearphishing threat, we propose a change of focus in the techniques that we use for detecting malicious emails: instead of looking for features that are indicative of attack emails, we look for emails that claim to have been written by a certain person within a company, but were actually authored by an attacker. We do this by modelling the email-sending behavior of users over time, and comparing any subsequent email sent by their accounts against this model. Our approach can block advanced email attacks that traditional protection systems are unable to detect, and is an important step towards detecting advanced spearphishing attacks.

1 Introduction

Spearphishing has become one of the most common ways used by attackers to infiltrate the network of a company, gain access to additional machines in it, or acquire sensitive information. For this type of attack, an email is crafted and sent to a specific person within a company, with the goal of infecting her machine with malware, luring her to hand out access credentials or to provide sensitive information. Recent research showed that spearphishing is a real threat, and that large companies are constantly targeted by this type of attack [37].

In a typical case, spearphishing emails appear to be coming from accounts within the same company or coming from a trusted party, to avoid raising suspicion by the victim [39]. This can be done in a trivial way, by forging the `From:`

field in the attack email. In more sophisticated attacks, however, the malicious emails are sent from an actual employee's email account whose machine has been compromised, or whose account credentials have been stolen. For the attacker, this modus operandi has the advantage that it leverages a user's social connections: previous research showed that users are more likely to fall for scams if the offending message is sent by somebody they trust [16]. Spearphishing attacks are particularly insidious for companies, because they can lead to large-scale compromises in their networks and to high financial losses, due to the sensitive information that might get stolen as a consequence of them [1, 2].

Spearphishing is not spam. The techniques that are currently used to detect spearphishing attacks are totally inadequate to counter them. Nowadays, the systems used to detect spearphishing attacks leverage traditional anti-spam techniques. However, these techniques were designed with a different threat model in mind: blocking unwanted bulk email. Adaptations of anti-spam techniques are not effective in fighting the spearphishing threat mainly for three reasons.

The first reason is that many anti-spam techniques are designed to fight bulk email, typically sent by botnets, and therefore leverage *similarity*. Techniques that leverage the similarities in the email templates or fingerprint the email lists to which bots send emails [24, 32, 43] work well in detecting spam and bot-infected machines, but fall short in detecting one-of-a-kind targeted email attacks, in which an attacker crafts an email tailored to the victim, and sends it only once. Even systems that look for changes of behavior in email accounts leverage the fact that accounts compromised by the same cybercriminals will show a similar behavior [10, 29, 30].

The second reason is that the *origin* of spearphishing emails is often times the correct one: there are numerous techniques that can detect emails sent by senders with a low reputation, and block them [14, 20, 41, 42]. However, if the attack email is coming from a reputable user's mailbox, whose machine has been compromised, these techniques will fail in detecting that email as malicious.

The third reason is that the words used in advanced targeted attack emails are the ones that are typically used in regular business emails. Therefore, techniques that analyze the email content, looking for words that are indicative of spam [9, 22, 27], are ineffective too. Moreover, even if malicious code is used in the email attachment, these binaries will not be off-the-shelf malware samples, and will therefore be unknown to traditional anti-virus and signature-based systems.

A new paradigm in fighting attack emails. Given how different spearphishing emails are compared to traditional spam and phishing emails, we propose a new detection paradigm to fight this threat. Instead of looking for signs of maliciousness in emails (such as offending words or suspicious origin) we want to determine if an email was actually written by the author that it claims to come from. Our approach is based on a simple, yet effective observation: users develop habits when sending emails. These habits include frequent interactions with certain people, sending emails at specific hours of the day, and using certain greetings and modal words in their emails. The core of our approach consists in building a profile for the email-sending behavior of a user. When the user's

account gets compromised, the attack emails that are sent are likely to show differences from this behavioral profile. We implemented our approach in a system, called IdentityMailer.

Anomalies can be more or less evident. An example of a "noisy" attack is a worm that sends an email to the entire address book of a user [38], which is a behavior that typical users do not show. In other cases, attackers might be more careful, and try to mimic the typical behavior of the person that they are impersonating in their emails. What they could do is sending emails only at hours in which the user is typically sending them, and only to those people she frequently interacts with, or even imitate the user's writing style.

To make it more difficult for attackers to successfully evade IdentityMailer, we build the email-sending behavior for a user by leveraging both the emails that the user sent in the past and a set of emails that the other users in the organization authored. In a nutshell, IdentityMailer compares the emails written by the user to the ones written by everybody else, and extracts those characteristics that are the most representative of the user's behavior. For example, certain functional words only used by a given user (and rarely by others) would model her behavior very well. When an attacker tries to learn a victim's sending behavior to mimic it in his attack emails, he only has access to that user's emails (since he compromised her account, or her machine), but not to the ones authored by the other users in the company. Therefore, what he can do is learning the most common habits of the user (such as the email address that is more frequently contacted, and at what time the user generally sends emails), but he has no guarantee that those traits are actually representative of the user's behavior. For example, most people send many emails on Monday morning, after they come back to the office from the weekend. Our system would give a low importance to this fact, since many users show the same behavior. On the other hand, there might be some times at which the user is the only one sending emails. Even if they are not that common, those emails are more representative of the user's behavior. However, an attacker has no way of knowing this, and might replicate the most common behavior in his attack emails.

IdentityMailer performs the analysis when emails are sent, before they are forwarded to the outgoing SMTP server. More specifically, IdentityMailer builds a behavioral profile based on the emails that a user sent in the past (and a set of emails authored by the other people in the organization). Then, every time an email is sent by that account, IdentityMailer checks this email against the profile learned for the account's owner. If the email does not match the learned profile, we consider it anomalous. The account might have been compromised, and the email might actually be an attempted attack. However, the anomaly might also be a false positive. Perhaps the user is working on a deadline, and is sending emails late at night, or is sending a personal email, and using a colloquial language, while the account is primarily used to send work-related emails. False positives are a big problem in traditional anti-spam systems, because they annoy users in the best case, and they prevent them from receiving important information in the worst case.

Luckily, the fact that IdentityMailer operates on the sending side of the email process comes to our aid. Any time an email is flagged as anomalous, we can start a process to verify the identity of a user. This process might be asking the user to answer a security question or some more sophisticated mechanism, such as a two-factor authentication scheme [6]. If the user correctly confirms her identity, we consider the anomaly as a false positive, and we send the email. In addition, we update the user's behavioral profile to include this email, to avoid similar false positives in the future. However, if the user does not solve the challenge, we consider the email as an attack, and we discard it. Of course, having to go through an identity-verification process is annoying for users. However, we think that having users confirm their identity once in a while is a fair price to pay to protect a company against advanced email attacks, as long as the verifications are rare enough (for example, one in every 30 emails).

In summary, this paper makes the following contributions:

- We present IdentityMailer, a novel approach to detect spearphishing emails: instead of looking for signs of maliciousness, we introduce a set of features that are representative of the email-sending behavior of a user, and propose a method to check emails against the learned sending behavior.
- We propose to leverage an identity-verification mechanism to mitigate false positives by IdentityMailer. We argue that such verification process, if reasonably rare, is acceptable for users.
- We tested IdentityMailer on a large dataset of publicly-available emails, as well as on multiple datasets of attack emails. We show that IdentityMailer works well in detecting attack emails.

2 Behavioral Profiles

It is important to accurately learn and model the email-sending behavior of a user, because this allows to perform a better detection of anomalous emails. However, it is not trivial to define user-specific traits that best distinguish a user's sending behavior. To determine these traits, IdentityMailer requires two datasets: a set $\mathbf{M}_u$ of emails written by a user U and a set $\mathbf{M}_o$ of legitimate emails written by other people. By comparing the emails in $\mathbf{M}_u$ to the ones in $\mathbf{M}_o$, we can extract the distinguishing characteristics of the email-sending behavior of U.

$\mathbf{M}_o$ should be composed of both emails sent by people working in the same organization as U, as well as of emails written by people who are completely unrelated to U. As we will explain later, the privacy concerns of our approach are minimal, because we do not save the full email, but only a feature vector associated to it. On one side, having $\mathbf{M}_o$ built from the emails sent by the users working in the same organization as U helps in giving less importance to those characteristics that are common for the people who work in that company. For example, if no user in the organization ever sends emails on a Sunday, it is less peculiar if the user also does not. On the other hand, having emails sent by users who are completely unrelated to U in $\mathbf{M}_o$ helps giving to the model examples of

which behavioral characteristics are uncommon in the organization, but common outside of it. We provide a more detailed description on how we build $\mathbf{M}_o$ in Section 2.2. By using only legitimate emails to build our behavioral profiles, we do not need to have ever observed any attack email to perform detection, similarly to what happens with traditional anomaly-detection systems. This is important, since the number of targeted attack emails is not high compared to legitimate emails [37]. In addition, this makes our approach independent from specific attack schemes.

To build the email-sending behavioral profile for a user, IdentityMailer proceeds in two steps. First, we extract a number of features for each email in $\mathbf{M}_u$ and $\mathbf{M}_o$. As a second step, we leverage the learned feature vectors to build a classification model, which represents the actual behavioral profile. This profile allows us to check any email that the user will send in the future, and determine whether it was likely written by the user, or if it might have been written by somebody else (i.e., a malicious party).

2.1 Features characterizing an email

For each email, we define three types of features: *writing habits*, *composition habits*, and *interaction habits*. Previous research showed that authorship identification is possible by just looking at stylometry features (which are a subset of what we call writing habits) [8]. However, these approaches rely on texts of a certain length [12]. Unfortunately, as we show in Section 4, many emails are short. If IdentityMailer relied only on the writing habits of a user, it would fail in flagging attack emails that are short as anomalous. Therefore, we need additional information for emails that are short in content. In the following, we describe the features that our approach uses to characterize an email.

Writing habits. People have their own style when writing. For example, some people use certain functional words (such as "although") more often than others, or write dates in a certain way. Analyzing a user's style has been used in the past to determine authorship of texts and emails [4, 8, 23]. Similarly, we consider a user's writing style a strong indicator that a certain email was indeed written by that user. An attacker could, in principle, learn the characteristics of his victim's style, and replicate them in the attack emails that she sends. However, previous research showed that imitation of another person's writing style is usually detectable [5].

In the following, we define a number of feature types that help defining a user's writing style. The complete list of writing-habit features can be found in our technical report [33].

1) Character occurrence. These features represent how often a character, or a set of characters, appear in the email text. Given a set of characters $\mathbf{C}$ and an email text M, we define the character occurrence of $\mathbf{C}$ in M o_c as the number of times that any of the characters in $\mathbf{C}$ occur in M, divided by the length of M. Examples of character occurrence features include the frequency of alphabetical letters (such as "a"), the frequency of certain punctuation signs (such as ";"), and the frequency of sets of characters (such as capital letters or cardinal numbers).

2) Functional word occurrence. These features represent how often the person uses specific functional words. We define as functional words those words that do not serve to express content, but instead are used to express grammatical relationships with other words within a sentence. These include adverbs (such as "when"), auxiliary verbs (such as "is"), and prepositions (such as "for"). Some of these features are useful to determine whether a user uses certain functional words in their extended or shortened form, and to what extent (for example, whether she usually uses "don't" instead of "do not"). Given a word FW and a set of words $\mathbf{W}_m$ in an email, we calculate the word occurrence o_{fw} in $\mathbf{W}_m$ as the number of times FW occurs in the email, divided by the size of $\mathbf{W}_m$.

3) Special word occurrence. These features represent how often a user uses certain "special" words in her emails. Special words include full names, dates, and acronyms. Given a regular expression R_{sw} representing the special word, an email M, and a set $\mathbf{W}_m$ containing the words in M, we calculate the special word occurrence o_{sw} of R_{sw} as the number of matches in M for R_{sw}, divided by the size of $\mathbf{W}_m$.

4) Generic style characteristics. These features represent generic characteristics of the style of a user. Examples include the type of bullets that the user uses in lists ("1-", "1.", or others), whether she uses a comma as a separator for large digits or not, and whether she uses a space after punctuation. Given a set of regular expressions $\mathbf{R}_{sc}$ representing a style characteristic, an email M, and a set $\mathbf{W}_m$ containing the words in M, we define the style characteristic s_c as the number of matches of the regular expressions in $\mathbf{R}_{sc}$ in the email M, divided by the size of $\mathbf{W}_m$.

5) Style metrics. These features capture information about the style of entire emails. Some features are rather simple, such as the number of paragraphs in the email. Others are more advanced, and depict the expressiveness of the language used in the email. Examples are the *Sichel measure* or the *Yule metric*, which describe how complex the vocabulary used by an author is. These metrics have been already used in previous work [40, 44].

Composition and sending habits. Other habits that users develop regarding their email-sending behavior do not have to do with their writing style, but rather with their way of composing emails. In the following, we describe this type of features.

1) Message characteristics. These features capture specific habits that the user has in the emails that she writes. Examples of such habits are including the original email at the end of a reply, including quotes to the original email interleaved with the text, or adding a signature at the end of the email. Message-characteristic features are boolean, meaning that they are set to 1 if a certain behavior is present in an email, and to 0 otherwise.

2) Time characteristics. Users tend to send emails at specific times of the day, and only during specific days. For example, most people working in an office will send emails between 9 am and 5 pm, from Monday to Friday. Given this observation, an email sent at midnight on a Saturday would be very suspicious. These features keep information about when an email has been sent. In particular, they

look at the day of the week and at the hour at which the email was composed. Similarly to other composition-habit features, time-characteristic features are boolean. We define seven features for the days of the week, and 24 features for the hours of the day.

3) URL characteristics. Some users include URLs in their emails. Users include links to pages that are needed for their job, or to websites that they consider interesting or entertaining. Over time, the domains of the URLs that a user includes in her emails tend to belong to a limited set (as previous research already noted [10]). On the other hand, if the user sent an email with a URL pointing to a domain that she has never included before, this might be suspicious.

To instantiate URL-characteristic features, we need a set of domains $\mathbf{L}_u$ that the user, as well as the other people in her organization, referenced in the past. This helps identifying those resources that are "internal" to the organization (which should be referenced often in the company's emails), and those that are not. We also include an "*other*" feature to take into account those domains that were never referenced by anybody in the organization. Similarly to the other composition-habit features, URL-characteristic features are boolean, and are set to 1 if that domain is referenced in the email, and 0 otherwise.

Interaction habits. The last type of features involve the social network of a user. Typical users will send most emails to a handful of contacts, who are coworkers or close friends. Having an email sent to an address that was never contacted before might be suspicious, especially if that user does not usually interact with many users.

To characterize the social network of a user, we look, for each email, at the email addresses that the email is addressed to (the `To:` field), as well as at the addresses that the email is sent to in carbon copy (the `CC:` field). We define four types of interaction-habit features, representing the addresses and the domains that a user sends emails to. The *recipient address list* features take into account the email addresses that an email is addressed to, while the *recipient domain list* ones look for the domains that those email addresses belong to. The idea behind this distinction is that if a user sends an email to an address that she has never referenced before, but that belongs to an organization that she often interacts with, this is less suspicious than an email addressed to a completely unknown domain. The *carbon copy address list* and the *carbon copy domain list* features work in the same way, but take into account the addresses in the `CC:` field of the email, rather than the ones in the `To:` field.

To instantiate the interaction-habit features, we need a list $\mathbf{L}_a$ of email addresses that the user, as well as the other people in the same organization, contacted in the past. It is important to look at the email addresses that the user has never contacted, but some of her coworkers have. This is because having a user sending an email to an executive she has never contacted before is very suspicious, and might be a sign of spearphishing. In addition, to account for those addresses and domains with which nobody in the organization has interacted before we add, for each of the four feature types, an "*other*" feature.

Similarly, we leverage a list $\mathbf{L}_d$ of domains to which the users in the organization have written emails in the past.

Interaction-habit features are boolean: they are set to 1 if an email is addressed to the address (or domain) represented by a given feature, and to 0 otherwise. If, for any of the four feature types, all features of that type have a value of 0, the "*other*" feature is set to 1.

2.2 Building Behavioral Profiles

To learn the distinguishing characteristics of the email-sending behavior for a user U, IdentityMailer compares the feature vectors built from the emails sent by the user ($\mathbf{M}_u$) to the feature vectors built from a set of legitimate emails sent by other people ($\mathbf{M}_o$). The challenge in picking $\mathbf{M}_o$ is to select a set of emails that is representative enough to make the most characteristic features of the behavior of the user stand out.

Given a user U who wrote a set of emails $\mathbf{M}_u$, we pick the set of emails $\mathbf{M}_o$ as follows. First, for each user U_i in the organization (other than U), we keep a set of emails that U_i has sent in the past. We call this set $\mathbf{M}_{ui}$. In addition, we consider a "special" user U_x. The set of emails $\mathbf{M}_{ux}$ corresponding to the user U_x consists of emails that were not written by the users in the organization. This set of emails could be a subset of the emails that were received by the company's mail server, or a set of publicly-available legitimate emails. Second, for each email in $\mathbf{M}_u$, we pick a random email written by another user U_i and add it to $\mathbf{M}_o$. We change the user U_i for each email in $\mathbf{M}_u$, in a round-robin fashion. By doing this, we ensure that the distribution of emails written by different users in $\mathbf{M}_o$ is uniform.

After having collected $\mathbf{M}_u$ and $\mathbf{M}_o$, we train a classifier to learn the email-sending behavioral profile of user U. To this end, we leverage Support Vector Machines (SVMs) trained with Sequential Minimal Optimization (SMO) [25]. The SMO algorithm is an iterative algorithm used to efficiently solve the optimization problem required for training SVMs. In Section 4.2 we analyze the classifier in detail.

Since the email-sending behavior of a user is likely to slightly change over time (for example, as the user makes new social connections), in IdentityMailer we keep updating the behavioral profile, by adding the new emails that the user sends. The identity-verification mechanism that we describe in Section 3 allows us to be sure that the emails that we add to the behavioral profile have been genuinely written by the user. Having a behavioral profile that is not static is important also because the behavioral profile for a user gets more accurate as the number of emails sent by the user increases. However, the strength of the model also depends on how consistent a user is in her email-sending habits. As we will discuss in Section 4.2, the features that we defined all contribute in defining the email-sending behavior of a user. The weight of the different features actually depends on each user's specific habits, and cannot be generalized.

3 Detecting Anomalous Emails

After having built the email-sending behavioral profile for a user, IDENTITYMAILER checks any email that the user tries to send against it. To do this, we go through the following algorithm:
Step 1: For each email M that the user U sends, we extract a feature vector $\mathbf{V}_m$.
Step 2: We compare $\mathbf{V}_m$ against the behavioral profile for U, which we call $\mathbf{BP}_u$. If $\mathbf{V}_m$ complies with $\mathbf{BP}_u$, we declare the email as being written by U, and go to step 4. Otherwise, we consider M as anomalous, and go to step 3.
Step 3: To make sure that the email has not been written by the user, we perform an identity-verification process for U. If the user correctly confirms her identity, M is considered as a false positive. We go to step 4. If, on the other hand, the user fails in confirming her identity (or decides not to, because she recognizes an attack), the email is considered as malicious, and discarded. In the next section, we describe how we envision the identity-verification process to happen.
Step 4: We add $\mathbf{V}_m$ to the set of feature vectors that are used to calculate $\mathbf{BP}_u$. This information will be used the next time that the behavioral profile is updated.

It is not necessary to update the behavioral profile for a user for every email that she sends. The reason is that, although the email-sending habits of a user change over time, they do not change that fast. In addition, as we will discuss in Section 4.4, updating the behavioral profile for all users may require a certain amount of time and resources. For these reasons, we envision the behavioral profile update as a batch process that could be performed daily or weekly.
Verifying a User's Identity. One of the main challenges that anti-spam systems have to face are false positives. Flagging a legitimate email as spam has a high impact on the user, because it might prevent that user from seeing that email at all. This happens because traditional anti-spam techniques operate on the receiving side of the email process, and often times it is impossible to verify that the sender of an email is who he actually claims to be. Operating on the sending side, on the other hand, has the advantage that we can ask the user whether she intended to send a certain email that looks suspicious, before the emails is actually sent.

In IDENTITYMAILER, we propose to start an identity-verification process when an account tries to send an email that looks anomalous. This verification process might be answering a security question or a more advanced method, such as a text message sent to the user's mobile phone as part of a two-factor authentication process [6]. Each method has advantages and disadvantages. However, analyzing the single identity-verification methods that one could implement goes beyond the scope of this paper. For our purposes, we just assume that, by going through the identity-verification process, the user can prove her identity with a high confidence.

Of course, we are aware that having to go through such a process might annoy users. However, we think that if the number of identity-verification processes

that a user has to go through is reasonably low, this is a fair price to pay to significantly increase the security of a company. In Section 4.2 we perform an analysis of how often a user would have to go through an identity-verification process, on average, and show that this number is reasonably low.

4 Evaluation

In this section, we evaluate the effectiveness of IdentityMailer. First, we describe the evaluation datasets that we used in our experiments. Then, we perform an analysis of the classifier used to build the email-sending behavioral profiles, and we show how the behavioral profiles built by IdentityMailer are useful to detect attack emails sent by compromised accounts.

4.1 Evaluation Datasets

To test IdentityMailer we leveraged a number of email datasets. First, we leveraged the *Enron corpus* [19] as a large-scale dataset of legitimate emails. This publicly-available dataset contains the emails sent by the executives of a large company, over the time of multiple years. In total, there are 148 users who sent emails in the dataset, for a total of 126,075 emails. The Enron dataset is representative of the type of emails sent in a large corporation (sending times, language, interactions), and this makes it suitable for our testing purposes. In the remainder of the paper, we call this dataset $\mathbf{D}_1$. As a second dataset of legitimate emails we used a set of emails that were donated to a large security company by their customers, for research purposes. This dataset is composed of 1,776 emails. The emails in this dataset are useful to complement $\mathbf{D}_1$ and give diversity. In particular, they are useful to populate $\mathbf{M}_{ux}$, as we explained in Section 2.2. We call this dataset $\mathbf{D}_2$. We use the datasets $\mathbf{D}_1$ and $\mathbf{D}_2$ for training purposes. In particular, for each user in $\mathbf{D}_1$, we build an email-sending behavioral profile, by leveraging both the emails in $\mathbf{D}_1$ and in $\mathbf{D}_2$.

For testing purposes, we needed a number of emails sent by compromised accounts, and preferably that were part of a targeted attack. The problem here is that, unlike regular spam, it is not easy to collect a large dataset of such emails. To overcome this problem, we manually selected three datasets of malicious emails. These emails come from a set of malicious messages that were not blocked by the anti-spam software of a large security company, and that were submitted by their customers for checking. The first dataset, that we call $\mathbf{S}_1$, is composed of generic spam emails. Such emails typically advertise goods or services, such as stock trading, pharmaceuticals, and dating sites. The main difference between the emails in $\mathbf{S}_1$ and common spam is that a state-of-the-art system failed in detecting them as malicious, and therefore we can consider them as "hard" to detect; we test IdentityMailer on this dataset to show that, although the system has not been designed to fight traditional spam, it performs well in detecting it, in case it was sent by compromised email accounts. $\mathbf{S}_1$ is composed of 43,274 emails. The second dataset, that we call $\mathbf{S}_2$, is composed of malicious emails (mostly phishing scams) that were sent by email accounts that

had been compromised. We selected these emails by looking at emails in the dataset that were malicious, but that had valid DKIM and/or SPF records [20,42]. In particular, these emails were sent by compromised accounts in the same domain as the customer who submitted them. In total, $\mathbf{S}_2$ contains 17,473 emails. The third dataset, which we call $\mathbf{S}_3$, is a dataset of sophisticated spearphishing emails. Such emails try to lure the user into handing out corporate-specific sensitive information (such as access credentials) to a malicious party, usually via social engineering. As we said, spearphishing emails are particular insidious to companies, because it can lead to high financial losses. $\mathbf{S}_3$ contains 546 emails. The emails in $\mathbf{S}_2$ and $\mathbf{S}_3$ closely resemble the threat model that we are trying to counter with IdentityMailer. In the next sections, we leverage these datasets to evaluate the effectiveness of IdentityMailer.

4.2 Analysis of the Classifier

In this section, we first describe how we selected the features used by IdentityMailer. Then, we investigate how well these behavioral profiles can determine if an email has been actually written by a user. As a third step, we show that the writing habits are usually not enough to detect whether an email is forged or not.

Instantiation of the features. As we explained in Section 2.1, some of the features used by our approach are specific to the organization in which the system is run. In particular, we need to know which email addresses and domains have been contacted by the users in the organization in the past, as well as the domains that have been referenced in the body of the emails, as part of URLs. We leveraged the dataset $\mathbf{D}_1$ to calculate the sets $\mathbf{L}_u$, $\mathbf{L}_a$, and $\mathbf{L}_d$. In particular, $\mathbf{L}_u$ was composed of 595 domains, $\mathbf{L}_a$ of 22,849 email addresses, and $\mathbf{L}_d$ of 3,000 domains. Note that, in a production environment, the size of $\mathbf{L}_u$, $\mathbf{L}_a$, and $\mathbf{L}_d$ would increase over time, since the users in the company would post more URLs, and contact new people. This means that the number of features used by IdentityMailer increases over time as well. However, this is not a problem, since the auxiliary lists are stored in a centralized location.

Accuracy of the classifier. To evaluate to what extent the behavioral profiles built by IdentityMailer are representative of the sending behavior of users, we proceeded as follows. First, for each user U in $\mathbf{D}_1$, we extracted the sets $\mathbf{M}_u$ and $\mathbf{M}_o$ for that user, following the algorithm described in Section 2.2. As we said, we use the emails sent by U as positive examples, and a mix of emails from $\mathbf{D}_1$ and $\mathbf{D}_2$ as negative examples.

After having a training set for each user, we performed a 10-fold cross validation on them, to investigate the accuracy of the behavioral profiles generated from them. The 10-fold cross validation gives us an idea of how the system would behave in the wild, while encountering previously-unseen emails. In particular, it gives us an estimate of how many emails written by the user would be flagged as malicious because of a change in behavior by the user, as well as how many attack emails would actually be missed by IdentityMailer. In this experiment, a false positive would indicate an email that was authored by the user,

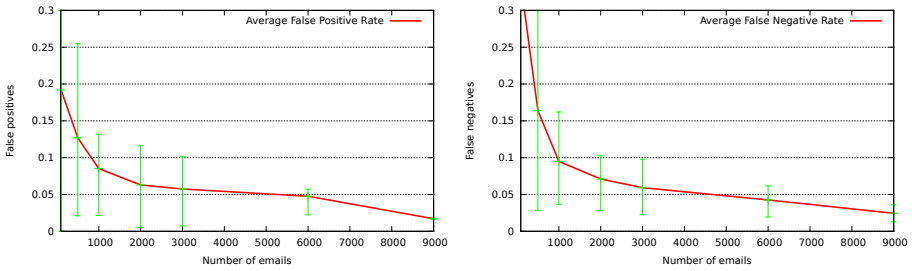

Fig. 1: Analysis of false positives (left) and false negatives (right) on the ten-fold cross validation. The X axis shows the number of emails that a user has sent in the past. As it can be seen, both false positives and false negatives decrease as the user sends more emails.

but that IdentityMailer considered as anomalous. In this case, IdentityMailer would have started an identity-verification process for the user, who would have correctly confirmed her identity, and have the email sent. We want the number of false positives to be low, because having to confirm her identity too often would annoy the user. Conversely, a false negative would indicate an email that had been written by somebody else, but that IdentityMailer mistakenly attributed to the user. We want false negatives to be as low as possible, because, in a real scenario, each of them would correspond to an attack that went undetected.

Intuitively, there are two factors that influence the robustness of the behavioral profile for a user. The first factor is the number of emails that a user has sent in the past. Having a larger number of examples of a user's sending style and habits makes the model more representative and less prone to false positives and false negatives. The second factor is how consistent is the sending behavior of a user. A user who always sends emails in the morning, and only to a small fraction of recipients, will be a lot more recognizable than a user who uses her account for both professional and personal use, and quite frequently sends emails at night.

The number of emails sent by the users in $\mathbf{D}_1$ varies substantially. On average, each user in $\mathbf{D}_1$ sent 840 emails, with a standard deviation of 1,345. The largest number of emails sent by a user in $\mathbf{D}_1$ is 8,926. As Figure 1 shows, the accuracy of the email-sending behavioral profile built by IdentityMailer increases as the user sends more emails. The error bars in the figure show that the accuracy of a behavioral profile does not only depend on the number of emails, but also on the user style and habits. For users who have consistent habits, IdentityMailer can reach almost no false positives and false negatives. On the other hand, certain users who have more variable habits end up having higher false positives and false negatives than average. However, this variability gets lower as the number of emails sent by the user increases.

Figure 1 (left) shows the average number of false positives generated during the 10-fold cross validation, broken down by the number of emails sent in the past by each user. As we explained, a false positive in this context would result in the user being required to solve an identity-verification mechanism. As it can be seen, a user who sent 1,000 emails in the past would have to confirm her identity every 12 emails that she sends, on average. Increasing the history of sent emails, a user who sent 8,000 emails would have to confirm her identity on average once every 58 emails that she sends. Given the number of emails that an average corporate user sends nowadays — 33 per day, according to a recent report [36], reaching this amount of interaction history does not take long. As we mentioned, these are average numbers. Users with a more stable email-sending behavior already reach 2% false positives after having sent 1,000 emails. This means that, on average, they would only have to go through the identity-verification process once every 50 emails they send. We think that these numbers are reasonable in a corporate environment, where the hassle of confirming a user's identity is repaid by having the users protected from identity theft.

Similarly, Figure 1 (right) shows the number of false negatives for the 10-fold cross validation. As it can be seen, having sent 1,000 emails in the past allows IdentityMailer to build a model that can block 90% of the emails that have not been written by the user. Recall gets better as the number of sent emails increases. The behavioral profile of a user who sent more than 8,000 emails has an average recall of 96%. The accuracy of IdentityMailer is lower than the one of state-of-the-art anti-spam systems. However, as we said, the purpose of our system is very different than the one of anti-spam techniques. We are trying to ensure that no email is sent on behalf of a user, if she did not compose it. As we have already discussed, current anti-spam techniques are not appropriate to deal with such attacks.

Analysis of the features. Previous research showed that it is possible to identify the author of an email just by looking at stylometry features (what we call *writing habits* in this paper) [8]. However, Forsyth et al. showed that such approaches are only reliable in presence of a consistent amount of text [12]. In particular, they identified the amount of text after which stylometry-based author identification becomes reliable to 250 words. Unfortunately, 78% of the emails in $\mathbf{D}_1$ are shorter than that. In particular, 50% of the emails in that set are shorter than 100 words.

Given the short length of emails, we use two other types of features in IdentityMailer: *composition habits* and *interaction habits*. We wanted to investigate how much these features help in making correct detections and whether it is true that writing-habit features are not enough. To show this, we performed the same 10-fold cross validation that we ran to evaluate the classifier, but this time we only used writing-habit features. The results show that writing-habit features are indeed not enough to perform an accurate detection. For a user who sent 1,000 emails in the past, the average number of false positives is 22%, almost three times higher than with the full classifier. The lowest rate of false positives reached in this case is for users who have sent at least 8,000 emails, but it is

still around 9.8%, almost six times higher than what we obtained with the full classifier. Clearly, if stylometry-based methods might be useful in forensic cases, they are not enough to determine whether an email has been sent by an attacker or not.

4.3 Detecting Attack Emails

We evaluated IdentityMailer on the attack datasets $\mathbf{S}_1$, $\mathbf{S}_2$, and $\mathbf{S}_3$. First, we created the email-sending behavioral profiles for each user U in $\mathbf{D}_1$, as explained in Section 2.2. Then, for each email in $\mathbf{S}_1$, $\mathbf{S}_2$, and $\mathbf{S}_3$, and each user U, we edited the `From:` field in the email to look like it was sent by U, and ran IdentityMailer on it, to see whether the email would have been flagged as anomalous or not, in case user U sent it. Since IdentityMailer does not take into account header fields such as the `X-Mailer` one, but only the set of recipients of the email, and the hour and day that the email was written at, no additional editing is required.

As it happened for the validation of the classifier, the performance of IdentityMailer depends on how many emails each user has sent in the past, as well as on how consistent the behavior of a user is while sending emails. In general, an email history of 200 messages is enough to reach a true positive rate of 80%, while histories of 1,000 emails or more lead to 90% detection rate. As a peak, IdentityMailer reaches 98% true positives for certain users. This number is still lower than what traditional anti-spam techniques can detect. However, as we mentioned, state-of-the-art systems are inadequate to detect this kind of threats, and fail in detecting them as malicious most of the time. Therefore, IdentityMailer fills this gap well, detecting most of these advanced attack emails as malicious.

4.4 Performance of IdentityMailer

A critical part of evaluating anti-spam techniques, and systems that deal with email in general, is understanding how much delay would be introduced by the technique in the delivery process. IdentityMailer has to undergo two main operations to detect malicious emails: building a behavioral profile for a user and checking each sent email against this behavioral profile. Building the behavioral profile for a user is not time-critical, because we expect the behavior of a user to be constant, or change slowly. For these reasons, the server can update the behavioral profile of users in batch, for example once a day, and during periods in which the email activity is minimal. According to our experiments, building the behavioral model for a user in $\mathbf{D}_1$ takes, on average, 34 seconds, and can take up to 141 seconds for certain users.

On the other hand, checking emails for maliciousness is more time-critical, because it actively delays emails as they get sent. On average, IdentityMailer requires 0.22 seconds to extract the feature vector for an email and compare it to the behavioral profile for that user. This time is comparable to state-of-the-art content-based anti-spam systems — `SpamAssassin` requires 0.5 seconds on average to process an email [3]. We consider this performance acceptable, also

because the number of emails that a typical organization sends is four times lower than the number of emails that it receives, and therefore IdentityMailer would have to process less emails than the ones that anti-spam systems have to [36].

5 Discussion and limitations

Our results show that IdentityMailer is successful in detecting and blocking attack emails that appear to have been written by a user, but have actually been authored by an attacker. However, as most detection systems, IdentityMailer has some limitations, as well as some caveats that an organization should keep in mind while operating it.

The main limitation is that, to be effective, IdentityMailer requires an email history of 1,000 emails or more. This makes it hard to protect, for example, the new hires of a company. We argue that email is such a pervasive communication medium that it should not take too long to collect a large number of emails from a new employee. In addition, a new employee is probably not going to be a good target for an attacker, who would favor more influential people in the company. Those people, however, will have a long email-sending history, and IdentityMailer will protect them well. Another possible limitation in a corporate setting is that high-ranked executives might delegate their assistants to write some emails on their behalf. This practice might generate false positives, because IdentityMailer would detect that those emails were not written by the owner of the account. A possible mitigation here is to learn multiple email-sending behaviors, one for each of the people using an account, and not generating an alert if the email results to be written by any of those users.

Another limitation of IdentityMailer is that writing-habit features are specific to the English language. If our approach had to protect the employees of a company whose main language is different than English, we would have to develop another set of language-specific features. Previous research showed that this is feasible even for Asian languages, that have completely different characteristics than English [47].

Another problem that we have to consider is the privacy of users. The email sending behavior is built not only by leveraging a user's personal emails, but also by leveraging the ones sent by her coworkers too. However, for how we designed IdentityMailer, the feature vectors built from the emails are kept within the server, and are never seen by the users. Also, the server has to only keep the feature vector relative to an email, and not the email itself. Therefore, we believe that the privacy concerns revolving around IdentityMailer should be minimal.

Another concern is that some domains, such as large webmail services, have a very diverse set of users, and it might be challenging to model their behavior well. For IdentityMailer, we focus on corporate users, assuming that their behavior is more consistent than the one of general-purpose email providers. In addition, large webmail services have access to additional signals that are not

included in our threat model (such as login patterns and IP addresses), which can also be leveraged to build a behavioral profile.

Another limitation is that an attacker could try is imitating the email-sending behavior learning phase of our system. To this end, he might leverage the emails that other users sent to the victim in the past as $\mathbf{M}_o$. The attacker can find these emails, for example, in the *Inbox* of the victim's mailer program. In principle, this technique could help in making the attack more successful, and evade IdentityMailer. However, the information that an attacker can learn from the emails received by a user in the past is rather limited. For example, it does not give any information on what the social network of the other users in the company looks like, and it only shows the behavior that third party showed when interacting with that specific user. An attacker might get additional knowledge of the company's emails by compromising additional employee email accounts. If he obtained access to enough accounts, he might be able to replicate the learning process of IdentityMailer and evade our system. However, an attack of such breadth is hard to set up, and once an attacker gets such a pervasive presence is the company's network, there is not much that our approach can do. In our technical report we show that IdentityMailer is in general difficult to evade by an attacker, unless he achieved a complete view on the email that have been sent by the company in the last months [33].

6 Related Work

Our approach protects the identity of users against attackers sending emails on their behalf. To this end, we borrow some ideas from anti-spam techniques, as well as from the field of forged text detection and authorship identification. In the following, we discuss how our approach is related to previous work, and elaborate on the novelty of our method.

Spam Filtering: Existing work on spam filtering can be distinguished in two main categories: *origin-analysis* and *content-analysis* techniques. Origin-analysis techniques try to determine whether emails are good or bad by looking at their origin. Examples of characteristics that are indicative of a malicious emails can be the IP address or autonomous system that the email is sent from, or the geographical distance between the sender and the recipient [14, 26, 32, 41]. Other origin-based techniques include *Sender Policy Framework* (SPF) and *DomainKeys Identified Mail* (DKIM) [20, 42]. These techniques try to determine whether an email is actually coming from the address it claims to come from, by looking at the sender IP, or at a signature in the email headers. Origin-based techniques are widely deployed, because they allow servers to discard spam emails as soon as the malicious end connects to the mail server, saving resources and time. In addition, they reach good coverage, because most spam is sent by hosts that are part of a botnet, and therefore have a low reputation [34]. However, in the scenario in which IdentityMailer works, origin-based techniques are useless, because the only thing they can do is confirming that an email has been sent by a certain account, regardless if it is a compromised one or not.

Content-analysis techniques look at the words in the message itself to determine if it is spam or not. Proposed methods include Naïve Bayes, Support Vector Machines, or other machine learning algorithms [9, 22, 27, 28]. Other systems detect spam by looking at malicious URLs in the email [17, 45]. Content-analysis techniques work well in detecting spam, however are too computationally intensive to be applied to every email that a busy mail server receives [35]. In IdentityMailer, we solve this problem by analyzing emails as they get sent. We claim that this analysis is feasible, because the number of emails that a mail server sends is lower than the number of emails that it receives. Another problem of traditional content-analysis techniques is that they look for words that are indicative of spam. In the presence of a targeted attack, there might be no such words, since an attack email will use a language that is similar to the one used in everyday business emails. This is why in IdentityMailer we learn the typical sending behavior of a user, and match it against the emails she sends, to detect attacks.

A number of systems have been proposed to counter specific types of spam, such as phishing. Such systems either look at features in the attack emails that are indicative of phishing [11], or at characteristics of the web page that the links in the email point to [46]. IdentityMailer is more general, since it can detect any type of attack emails that is sent by compromised accounts. In addition, existing phishing techniques fail in detecting those emails that rely on advanced social engineering tactics, instead of redirecting the user to a phony login page.

Another category of spam detection techniques looks at the way in which spammers use the TCP or SMTP protocol [18, 31]. These techniques work well in practice against most spam, but are focused on detecting hosts that belong to a botnet, and are therefore useless in detecting the type of attacks that IdentityMailer is designed to prevent.

Forged Message Detection: A large corpus of research has been performed on determining the author of an email. These techniques typically leverage stylometry and machine learning and return the most probable author among a set of candidates [4, 7, 8, 13, 15]. From our point of view, these approaches suffer of two major problems: the first one is that they typically need a set of possible authors, which in our case we do not have. The second problem is that email texts are often times very short, and this does not allow to determine an author by just looking at stylometry [12]. Lin et al. proposed a system that looks at the writing style of an email, and is able to tell whether that email was written by an author or not [21]. This approach solves the first problem, but does not solve the second one, in which we have emails that are too short to make a meaningful decision. To mitigate this problem, in IdentityMailer we leverage information other than stylometry, such as the typical times in which a user sends emails, or her social network.

Stolfo et al. presented Email Mining Toolkit (EMT) [29, 30]. This tool mines email logs to find cliques of users who frequently contact each other. After learning the cliques, the system flags as anomalous emails that are addressed to people outside them. Although EMT leverages an idea similar to IdentityMailer's

interaction features, it is tailored at detecting large-scale threats, such as worms spreading through email. The fact that IdentityMailer leverages other types of features allow our system to detect subtle, one-of-a-kind attack emails.

Egele et al. proposed a system that learns the behavior of users on Online Social Networks and flags anomalous messages as possible compromises [10]. Because of the high number of false positives, their system can only detect large-scale malicious campaigns, by aggregating similar anomalous messages. As we have shown, IdentityMailer is able to detect attacks that are composed of a single email, and that have not been seen before.

7 Conclusions

We presented IdentityMailer, a system that protects the identity of corporate users, by checking if an email has been written by the owner of an email account. This work is the first step towards the protection of individuals and companies against advanced email attacks, such as spearphishing. IdentityMailer does this by learning the typical sending behavior of the account's owner and checking any email that the account sends against this profile. We showed that IdentityMailer is able to detect attacks that state-of-the-art systems are unable to detect.

Acknowledgments

This work was supported by a Symantec Research Labs Graduate Fellowship for the year 2012. We would like to thank the anonymous reviewers for their useful comments. We would also like to thank the people at Symantec, in particular Marc Dacier, David T. Lin, Dermot Harnett, Joe Krug, David Cawley, and Nick Johnston for their support and comments. We would also like to thank Adam Doupè and Ali Zand for reviewing an early version of this paper. Your feedback was very helpful.

References

1. Hacking attack at RSA targeted Flash flaw. `http://www.ft.com/cms/s/2/96518afc-5cb1-11e0-ab7c-00144feab49a.html`.
2. Shamoon was an external attack on Saudi oil production. `http://www.infosecurity-magazine.com/view/29750/shamoon-was-an-external-attack-on-saudi-oil-production/`.
3. SpamAssassin: performance. `http://http://wiki.apache.org/spamassassin/UsingNetworkTests`.
4. Abbasi, A., Chen, H., and Nunamaker, J. F. Stylometric identification in electronic markets: Scalability and robustness. *Journal of Management Information Systems* (2008).
5. Afroz, S., Brennan, M., and Greenstadt, R. Detecting hoaxes, frauds, and deception in writing style online. In *IEEE Symposium on Security and Privacy* (2012).

6. Aloul, F., Zahidi, S., and El-Hajj, W. Two factor authentication using mobile phones. In *IEEE/ACS International Conference on Computer Systems and Applications* (2009).
7. Calix, K., Connors, M., Levy, D., Manzar, H., MCabe, G., and Westcott, S. Stylometry for e-mail author identification and authentication. *Proceedings of CSIS Research Day, Pace University* (2008).
8. Corney, M. W. Analysing E-mail Text Authorship for Forensic Purposes.
9. Drucker, H., Wu, D., and Vapnik, V. N. Support vector machines for spam categorization. In *IEEE transactions on neural networks* (1999).
10. Egele, M., Stringhini, G., Kruegel, C., and Vigna, G. COMPA: Detecting Compromised Social Network Accounts. In *Symposium on Network and Distributed System Security (NDSS)* (2013).
11. Fette, I. and Sadeh, N. and Tomasic, A. Learning to Detect Phishing Emails.
12. Forsyth, R., and Holmes, D. Feature finding for text classification. In *Literary and Linguistic Computing* (1996).
13. Frantzeskou, G., Stamatatos, E., Gritzalis, S., Chaski, C. E., and Howald, B. S. Identifying authorship by byte-level n-grams: The source code author profile (scap) method. *International Journal of Digital Evidence* (2007).
14. Hao, S., Syed, N. A., Feamster, N., Gray, A. G., and Krasser, S. Detecting Spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine. In *USENIX Security Symposium* (2009).
15. Iqbal, F., Hadjidj, R., Fung, B., and Debbabi, M. A novel approach of mining write-prints for authorship attribution in e-mail forensics. *Digital Investigation* (2008).
16. Jagatic, T. N., Johnson, N. A., Jakobsson, M., and Menczer, F. Social phishing. *Communications of the ACM* (2007).
17. John, J. P., Moshchuk, A., Gribble, S. D., and Krishnamurthy, A. Studying Spamming Botnets Using Botlab. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2009).
18. Kakavelakis, G., Beverly, R., and J., Y. Auto-learning of SMTP TCP Transport-Layer Features for Spam and Abusive Message Detection. In *USENIX Large Installation System Administration Conference* (2011).
19. Klimt, B., and Yang, Y. Introducing the Enron Corpus. In *CEAS* (2004).
20. Leiba, B. DomainKeys Identified Mail (DKIM): Using digital signatures for domain verification. In *CEAS* (2007).
21. Lin, E., Aycock, J., and Mannan, M. Lightweight client-side methods for detecting email forgery. In *Workshop on Information Security Applications (WISA)* (2012).
22. Meyer, T., and Whateley, B. SpamBayes: Effective open-source, Bayesian based, email classification system. In *CEAS* (2004).
23. Narayanan, A., Paskov, H., Gong, N. Z., Bethencourt, J., Stefanov, E., Shin, E. C. R., and Song, D. On the feasibility of internet-scale author identification. In *IEEE Symposium on Security and Privacy* (2012).
24. Pitsillidis, A., Levchenko, K., Kreibich, C., Kanich, C., Voelker, G. M., Paxson, V., Weaver, N., and Savage, S. botnet Judo: Fighting Spam with Itself. In *Symposium on Network and Distributed System Security (NDSS)* (2010).
25. Platt, J., et al. Sequential minimal optimization: A fast algorithm for training support vector machines.
26. Ramachandran, A., Feamster, N., and Vempala, S. Filtering Spam with Behavioral Blacklisting. In *ACM Conference on Computer and Communications Security (CCS)* (2007).

27. Sahami, M., Dumais, S., Heckermann, D., and Horvitz, E. A Bayesian approach to filtering junk e-mail. *Learning for Text Categorization* (1998).
28. Sculley, D., and Wachman, G. M. Relaxed Online SVMs for Spam Filtering. In *ACM SIGIR Conference on Research and Development in Information Retrieval* (2007).
29. Stolfo, S. J., Hershkop, S., Hu, C.-W., Li, W.-J., Nimeskern, O., and Wang, K. Behavior-based modeling and its application to email analysis. *ACM Transactions on Internet Technology (TOIT)* (2006).
30. Stolfo, S. J., Hershkop, S., Wang, K., Nimeskern, O., and Hu, C.-W. Behavior profiling of email. In *Intelligence and Security Informatics*. 2003.
31. Stringhini, G., Egele, M., Zarras, A., Holz, T., Kruegel, C., and Vigna, G. B@BEL: Leveraging Email Delivery for Spam Mitigation. In *USENIX Security Symposium* (2012).
32. Stringhini, G., Holz, T., Stone-Gross, B., Kruegel, C., and Vigna, G. BotMagnifier: Locating Spambots on the Internet. In *USENIX Security Symposium* (2011).
33. Stringhini, G., and Thonnard, O. That ain't you: Detecting spearphishing emails before they are sent. *arXiv preprint arXiv:1410.6629* (2014).
34. Symantec Corp. Symantec intelligence report. `http://www.symanteccloud.com/mlireport/SYMCINT_2013_01_January.pdf`, 2013.
35. Taylor, B. Sender reputation in a large webmail service. In *CEAS* (2006).
36. The Radicati Group. Email Statistics Report. `http://www.radicati.com/wp/wp-content/uploads/2011/05/Email-Statistics-Report-2011-2015-Executive-Summary.pdf`.
37. Thonnard, O., Bilge, L., O'Gorman, G., Kiernan, S., and Lee, M. Industrial espionage and targeted attacks: understanding the characteristics of an escalating threat. In *Symposium on Recent Advances in Intrusion Detection (RAID)* (2012).
38. Threatpost. New Email Worm Turns Back the Clock on Virus Attacks. `http://threatpost.com/en_us/blogs/new-email-worm-turns-back-clock-virus-attacks-090910`, 2010.
39. Trend Micro Inc. Spear-Phishing Email: Most Favored APT Attack Bait, 2012.
40. Tweedie, F., and Baayern, R. How variable may a constant be? Measures of lexical richness in perspective. *Computers abd the humanities* (1998).
41. Venkataraman, S., Sen, S., Spatscheck, O., Haffner, P., and Song, D. Exploiting Network Structure for Proactive Spam Mitigation. In *USENIX Security Symposium* (2007).
42. Wong, M., and Schlitt, W. RFC 4408: Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. `http://tools.ietf.org/html/rfc4408`, 2006.
43. Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., and Osipkov, I. Spamming Botnets: Signatures and Characteristics. *SIGCOMM Comput. Commun. Rev. 38* (August 2008).
44. Yule, G. The statistical study of literary vocabulary. *Cambridge University Press* (1944).
45. Zalewski, M. p0f v3. `http://lcamtuf.coredump.cx/p0f3/`, 2012.
46. Zhang, Y. and Hong, J.I. and Cranor, L.F. Cantina: a Content-based Approach to Detecting Phishing Web Sites.
47. Zheng, R., Li, J., Chen, H., and Huang, Z. A Framework for Authorship Identification of Online Messages: Writing-Style Features and Classification Techniques. *Journal of the American Society for Information Science and Technology* (2005).