Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Phishing detection and impersonated entity discovery using Conditional Random Field and Latent Dirichlet Allocation

Venkatesh Ramanathan*, Harry Wechsler

Department of Computer Science, George Mason University, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 8 September 2011
Received in revised form
27 October 2012
Accepted 3 December 2012

Keywords:
Boosting
Conditional Random Field
Identity theft
Impersonated entity discovery
Latent Dirichlet Allocation
Machine learning
Named entity
Natural language processing
Phishing

ABSTRACT

Phishing is an attempt to steal users' personal and financial information such as passwords, social security and credit card numbers, via electronic communication such as e-mail and other messaging services. Attackers pretend to be from a legitimate organization and direct users to a fake website that resembles a legitimate website, which is then used to collect users' personal information. In this paper, we propose a novel methodology to detect phishing attacks and to discover the entity/organization that the attackers impersonate during phishing attacks. The proposed multi-stage methodology employs natural language processing and machine learning. The methodology first discovers (i) named entities, which includes names of people, organizations, and locations; and (ii) hidden topics, using (a) Conditional Random Field (CRF) and (b) Latent Dirichlet Allocation (LDA) operating on both phishing and non-phishing data. Utilizing topics and named entities as features, the next stage classifies each message as phishing or non-phishing using AdaBoost. For messages classified as phishing, the final stage discovers the impersonated entity using CRF. Experimental results show that the phishing classifier detects phishing attacks with no misclassification when the proportion of phishing emails is less than 20%. The *F*-measure obtained was 100%. Our approach also discovers the impersonated entity from messages that are classified as phishing, with a discovery rate of 88.1%. The automatic discovery of impersonated entity from phishing helps the legitimate organization to take down the offending phishing site. This protects their users from falling for phishing attacks, which in turn leads to satisfied customers. Automatic discovery of an impersonated entity also helps email service providers to collaborate with each other to exchange attack information and protect their customers.

Abbreviations: AUC, Area Under Receiver Operating Characteristic; CRF, Conditional Random Field; EM, Expectation Maximization; HTML, Hyper Text Markup Language; IP, Internet Protocol; LDA, Latent Dirichlet Allocation; MIME, Multipart Internet Mail Extension; NER, Named Entity Recognition; POS, Part of Speech; ROC, Receiver Operating Characteristic; TDF, Term Document Frequency; URL, Uniform Resource Locator.

* *Corresponding author.*

E-mail addresses: vramanat@gmu.edu (V. Ramanathan), wechsler@gmu.edu (H. Wechsler).

0167-4048/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2012.12.002

1. Introduction

Stealing a person's identity is one of the most profitable crimes committed by criminals. Among 1.3 million complaints received by the Federal Trade Commission in 2009, identity theft ranked first and accounted for 21% of the complaints costing consumers over 1.7 billion US dollars (National Data, 2009). Identity theft has been around for many years while the means of committing it has changed with technology. The traditional way criminals steal a person's identity is by killing the individual. Another way to steal identity is using phone scams, wherein, criminals inform the person that they have won a sweepstake, and convince the user to reveal some personal information to claim the money. The more popular method of identity theft that is prevalent even today is called Dumpster Diving. When people discard letters, financial records, and other personal information in the garbage dump without shredding, criminals scavenge those dumps looking for sensitive information such as credit card, bank account and social security numbers and use that information to commit crimes.

With the advent of internet, the most popular way to steal identity is through "phishing". Like in traditional fishing where fishermen trolls the river in a boat to catch fish, in "phishing", attackers trolls the internet using email message with convincing content as baits to steal users personal information. The email directs the user via a hyperlink to a website owned by criminals that looks very similar to a legitimate website. The user will then be asked to enter personal and financial information either to update existing information or to purchase a product. In reality this lets the criminal to have access to that valuable information which they then use to commit fraud or to sell it to a bidder. Attackers can also trick users into downloading malicious codes or malware after they click on a link embedded in the e-mail. This is a useful tool in crimes like economic espionage where sensitive internal communications can be accessed and trade secrets stolen. Phishing has been around since 1996 but has become more common and more sophisticated. Recent attack on the Gmail system, that stole email accounts of government officials, contractors and military personnel (PC World, 2011) is evidence to the sophistication of such attacks.

Considerable research has been done toward protecting users from phishing attacks. They include firewalls, blacklisting certain domains and internet protocol (IP) addresses, spam filtering techniques, fake website detection, client side toolbars and user education. Each of these existing techniques has some advantages and some disadvantages. For example, the blacklist approach is harder to maintain with every expanding IP address/domain space, the filtering techniques have high misclassification, while users do not pay careful attention to client side toolbar warnings thus falling for phishing attacks.

The goal of our novel approach is not only to detect phishing attack but also the organization that attacker is impersonating. Toward that goal, we have developed a robust multi-stage content driven methodology, which can be implemented as a filter on email servers and web servers, to automatically detect phishing messages and discover the impersonated entity in those messages. The methodology combines the power of natural language processing and machine learning to build the content filter. The methodology employs named entity extraction and topic discovery methods for feature extraction. Named entities are extracted using Conditional Random Field (CRF) and topics are discovered using Latent Dirichlet Allocation (LDA). A robust classifier is built by employing topic distribution probabilities and named entities as features and the classification method, AdaBoost. Once the content is classified as phishing, the methodology employs CRF to identity the impersonated entity, i.e., the organization that this phishing attack is impersonating.

The novel contribution of this paper is multi-stage methodology that detects not only phishing but also discovers the impersonated entity from such attacks. While the detection of phishing helps to protect users from falling to identity theft, the automatic discovery of impersonated organization helps legitimate organization to shut down the fake website before their users potentially fall for one. This keeps legitimate company's customers safe and secure which in turn benefits the company having long lasting customers. It also helps companies to have partnership with other companies to mutually exchange phishing campaign targeted toward each other, which in turn protects respective customers. The combined application of CRF and LDA methods to detect phishing attacks and to discover the impersonated entity is also a novelty of this research. Topic discovery helps to discover impersonated entity and the discovery of impersonated entity helps to discover better topics. This paper is organized as follows. We first review state of the art protection techniques and present their advantages and disadvantages (see section 2). The research methodology is presented in section 3. The multi-stage implementation architecture is presented in section 4. Experiments and results obtained from the openly available dataset are presented in section 5. Section 6 presents a brief overview of autonomic computing while Section 7 proposes a framework for making our developed methodology autonomic and its computational efficiency. Section 8 presents a discussion of the developed methodology and how it can be extended for future research. This paper concludes in section 9. Methods employed by the research methodology namely, CRF, LDA and AdaBoost are presented in appendix A, B and C respectively.

2. Background

The primary motivation for attackers using phishing is to steal identity from users. Several techniques have been developed to protect users from phishing attacks. A review of the state-of-the-art tools is presented below.

The protection at network (router and firewall) level is typically achieved by blocking a range of IP addresses or a set of domains from entering the network. Firewall protects organization's network by blocking traffic directed to certain ports and IP addresses from suspicious ports and IP addresses. A firewall may have rules such as 'block all traffic to any port greater than 1024', 'block all traffic from any IP address that matches 165.25.x.x', etc. DNSBL (2011) is a database widely used for blocking at IP address level by several internet service providers. This list is updated with new IP addresses, after observing for a period of time abusive behavior. Email providers keep this list updated at the firewall level periodically. When a Simple Mail Transfer Protocol (SMTP) connection request comes from one of these blacklisted IP addresses, connection is refused, thus preventing phishing and spam emails from these IP addresses from reaching the end user. DNSBL (2011) is employed for sender level blocking. Firewall rule to block phishing is a reactive approach (instead of a proactive approach). Also, attackers evade this protection technique by hijacking legitimate user's PC and constantly moving from one IP to another IP address. SURBL (2011) and URIBL (2011) provide a list of domains, categorized as phishing or spam, extracted from message bodies. The domain blacklist is created by setting up spam traps and honeypots at several locations and performing content level analysis of messages. Organizations block or spam folder the email message if a SMTP connection request comes from these blacklisted domains or the message body contains these domains. Similar to DNSBL (2011), SURBL (2011) and URIBL (2011) list based blocking is a reactive approach, as list is updated after observing abusive behavior. Snort (2011) is open source intrusion detection system software that is employed at network level. Snort (2011) examines network packet headers and performs anomaly detection. If a large number of connection requests come from certain IP addresses to large number of different ports, Snort can have rule in place to block traffic from them. These rules are threshold driven which require careful consideration. Also, these rules to enforce protection must constantly be updated and require manual intervention. PhishTank (2012) provides a list of confirmed phishing sites. Email providers block phishing emails if the message body contains PhishTank (2012) URLs in them. PhishTank (2012) contains only a partial list of phishing URLs. Moreover, majority of phishing sites (approximately 60%) are short-lived (lasts less than 2 h), according to APWG (2012). Thus, blocking phishing emails using this approach is not fully effective. Several other network level approaches have been developed for spam detection. This includes (i) behavioral blacklisting approach that classifies email message based on spatial and temporal traffic pattern of the email (Feamster, 2008); (ii) network flow approach that identifies spam message based on set of IP packets that passes an observation point in the network during a certain time interval and having a certain set of common properties (Sperotto et al., 2009); and, (iii) geographic coordinate based approach that identifies spam based on region that IP address belong to (Cortez et al., 2010). All the above network level approaches are targeted for email spam detection instead of phishing detection.

Phishing is also prevented by enforcing authentication prior to using the system. There are two levels of authentication, user level and domain level. Typically, the email service provider authenticates user, before he or she sends an email (user level). The domain level authentication is performed in the provider–provider communication (one mail server to the other mail server). The user level authentication is performed using password as credentials. The password authentication can easily be cracked as evidenced by surge in phishing attacks. Microsoft has developed a technology called Sender ID (2006) while Yahoo has a similar technology called DomainKey (2006). Both these techniques perform domain level authentication. In order for these domain level techniques to work, providers on the sender and the recipient side must implement the same technology. Due to lack of agreement between email providers, this technology is not that prevalent.

Server side filters and classifiers typically extract features from the email and train a classifier to classify phishing email vs. good email. SpamAssassin (2011) is a widely used host level filter. This is a rule-based filter that requires constantly changing for the rule to be effective. Attackers figure out the rule being employed and bypass these filters by appropriately constructing the email. PILFER (Fette et al., 2007) is another email classifier that is trained using 10 features extracted from email data. Both these filters have high misclassification rates. phishGILLNET (Ramanathan and Wechsler, 2012), is a robust content classifier, developed by the authors, that employs topic modeling method, Probabilistic Latent Semantic Analysis (PLSA), AdaBoost and Co-Training. phishGILLNET (Ramanathan and Wechsler, 2012) outperformed state-of-the-art phishing classifiers and yielded perfect classification on public corpus email datasets when co-training algorithm was employed. It also has the additional benefit of handling unlabeled examples thus saving time and labor involved in human annotation. This paper is an improvement over phishGILLNET (Ramanathan and Wechsler, 2012) in the sense that it employs topics modeling method LDA (as compared to PLSA) and CRF to not only detect phishing attacks but also discover impersonated entity.

Tools that operate on the client side (i.e., user's machine) include user profile filters and browser based toolbars. SpoofGuard (2004), CatchingPhish (Cordero and Blain, 2006), CallingID (2011), CloudMark (2011), NetCraft (2011), FirePhish (2006), eBay toolbar (2007) and IE Phishing Filter (2011) are some of the client side tools. User profile filters observe user's website visiting pattern and maintain a list of URLs in local database. When a user visit's a URL that is different from his/her website visits, it warns the user with a dialog. Toolbars are built and trained using the typical pattern of phishing website URLs. Some patterns of phishing website include IP address in the URL, long URLs, many dots in the URL, etc. This technique is very susceptible to technology changes (such as IPV4 vs. IPV6, URL shortening services) and hence it is not robust. Moreover, most users ignore the warnings displayed by the dialogs. Client side tools are helpful for users who pay careful attention to warnings and act accordingly. It also helps when the tools are part of a provider who does not have a robust server side filter.

The other form of protection involves educating the user about phishing attacks and pattern of phishing email. The basic mode of educating user is posting help pages in websites and warning the user about phishing. MailFrontier (2011) has setup a website containing screenshots of several phish emails. Robila and Ragucci (2006) evaluated the effect of user education in differentiating phishing and good emails. The authors presented a lecture on how to identify phishing emails and the harm of falling for a phish in an introduction to computing class. At the end of the lecture, the authors presented student with both phishing and good emails. Students were then asked to identify the email type. The study concluded that students identified phishing emails correctly after the lecture. Students also acknowledged the usefulness of the lecture. Similar study was also conducted at the Indiana University (Jagatic, 2007).

Existing protection techniques have some advantages and some disadvantages in stopping the phishing attacks and protecting the end user. Network level protection using domain and IP address blacklisting require periodic updates and are reactive in nature as list can be updated only after observing abuse pattern for some time period. Moreover, attackers can compromise legitimate user's machine to conduct phishing attacks and hence blacklisting may block legitimate user from using the web. Client side tools and filters rely on user interaction and user actions to prevent users from falling to phishing attacks. Most users do not pay attention to warning dialogs and they end up falling for phishing attacks. Among the server side content filters to detect phishing attacks, phishGILLNET developed by the authors (Ramanathan and Wechsler, 2012) outperformed state-of-the-art phishing classifiers. The goal of this research is to improve our earlier work by employing CRF for named entity extraction, LDA for topic discovery, and AdaBoost for classification to not only detect phishing attacks but also discover impersonated organizations. The research methodology is described in the following section.

3. Methodology

The research methodology employs natural language processing and machine learning to detect phishing attacks and to discover the entity/organization that the attackers impersonate during phishing attacks. The multi-stage methodology first extracts in Stage I, (i) named entities, which includes names of people, organizations, and locations; and (ii) hidden topics, using CRF and LDA operating on both phishing and non-phishing data. Next in Stage II, utilizing topics and named entities as features, each message is classified as phishing or non-phishing using AdaBoost. Finally in Stage III, for messages classified as phishing, the impersonated entity is discovered using CRF. A schematic representation of the multi-stage research methodology is shown in Fig. 1. The multi-stage research methodology and motivations for applying CRF, LDA, and AdaBoost methods are described in this section.

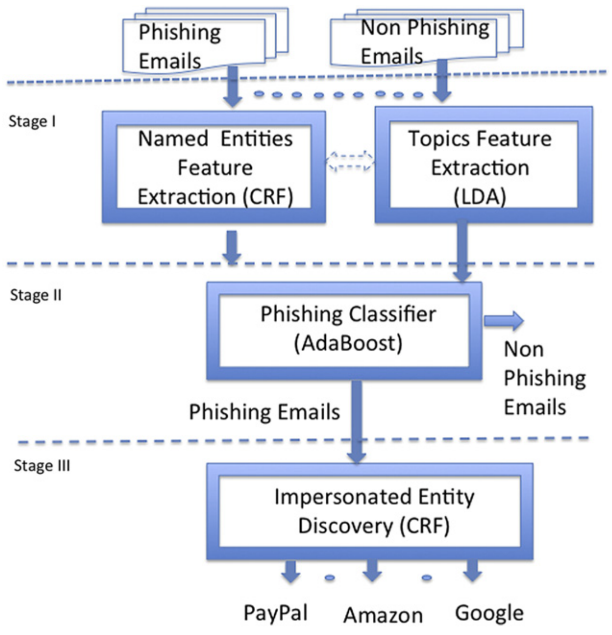
Phishing Emails
Non Phishing Emails
Stage I
Named Entities Feature Extraction (CRF)
Topics Feature Extraction (LDA)
Stage II
Phishing Classifier (AdaBoost)
Non Phishing Emails
Phishing Emails
Stage III
Impersonated Entity Discovery (CRF)
PayPal
Amazon
Google

Fig. 1 – Methodology.

3.1. Stage I – *feature extraction (CRF, LDA)*

The first stage is the feature extraction stage. Two sets of features are extracted during this stage namely, named entities and topics. Named entities are extracted using CRF in *Stage I(a)* and topics are extracted using LDA in *Stage I(b)*.

3.1.1. *Stage I(a) – named entity feature extraction (CRF)*

In this stage, named entities are first extracted, which are then used in *Stage II* as one set of features to build a classifier for phishing detection. Named entities are proper names (or proper nouns) that are names of people, organizations, locations etc. An example of a "phishing email" and a "non-phishing email" are shown below from the author's mailbox. Named entities are in bold and italicized.

3.1.1.1. *Phishing email.*

*"subject: **PayPal** online : message alert!*

Dear Customer

resolution center: your account is limited. regarding this, please follow the link below to resolve this issue:

click here to resolve the problem http://autoplusoman.com/security.html

***PayPal** - number: id832329-paypal/2011*

please allow us 1 to 3 days to resolve your problem."

3.1.1.2. *Non-phishing email.*

*"Dear **Venkatesh Ramanathan,***

You just changed your password.

If you didn't change your password, give us a call right away at 402-935-7733.

Just a reminder:

Never share your password with anyone.

Create passwords that are hard to guess and don't use personal information.

Be sure to include uppercase and lowercase letters, numbers, and symbols.

Use different passwords for each of your online accounts.

Sincerely,

PayPal

Copyright © 2012 ***PayPal****, Inc. All rights reserved.*

PayPal *is located at 2211 N. First St.,* ***San Jose, CA*** *95131."*

In the above examples, ***PayPal*** is the name of an organization, ***Venkatesh Ramanathan*** is the name of a person, ***San Jose*** is the name of a city, and **CA** is the name of a state. In the first stage, we make use of CRF for named entity recognition (NER), which is an information retrieval task that seeks to locate and classify elements in text documents as one of these proper names (see Fig. 1). CRF is the method of labeling proper names (names of people, locations, organizations, etc.) in a body of text. Given a sentence, the method involves determining words that are named entities and appropriately labeling the entity as the correct proper name. A brief theoretical description of CRF is given in Appendix A. A review of the literature reveals that the CRF model has been successfully employed for labeling and parsing sequential data in natural language processing and image processing applications.

Phishing emails typically target financial, social networking, online gaming and email service providers. According to the Anti Phishing Working Group (APWG, 2012), 75% of the phishing attacks target financial institutions, with ***PayPal*** being one of the top phishing targets. If an email contains the name of a financial institution, such as ***PayPal***, there is a high probability that the email is a phishing email. A key motivation for using CRF is its capability to automatically extract named entities from the body of the email, resulting in an improvement to classification accuracy and helping to narrow the search when processing large volumes of emails. It is also possible that the email sent by the financial institution is a legitimate email (see "non-phishing email" example above). The determination whether the email is legitimate or not is made in *Stage II* using organization names as one of the features.

The second motivation for using CRF is its robustness in extracting named entities (such as names of organizations). CRF extracts names of organizations based on the context in which such words appear, the words that precede the given word, and the words that succeed the given word. In the phishing example above, "*is located at*" is the context in the following sentence: "***PayPal*** *is located at 2211 N. First St.,* ***San Jose, CA*** *95131.*" New organizations could emerge and existing corporations could merge to form a new organization. Attackers may start targeting these new organizations. Since CRF extracts names of companies based on context, we do not have to keep a pre-defined list of all company names.

The third motivation for using CRF is its ability to automatically extract personalized information (names of people) from a given document. Attackers send out phishing emails in bulk hoping some users would fall for the attack. Thus, most phishing emails are not personalized (see "phishing" example above that has "*Dear Customer*" whereas "non-phishing" email has "*Dear Venkatesh Ramanathan*"). This is because: a) the attackers do not have the users' names, or, b) they know the users' names but they do not know if the users have accounts with the organization that the attackers are impersonating, or, c) they do not want to spend time in composing millions of personalized emails. Emails from legitimate organizations sent to their registered customers are, in general, personalized as they do have names of the people (see "non-phishing" email example). Thus, automatic extraction of names of the people using CRF, which would normally be present in a good email but usually absent in a phishing email, are employed as additional features for building the classifier. There are several exceptions to this. The targeted phishing attacks, known as spear phishing, where attackers do know user's names, are personalized. Similarly, legitimate emails such as ones sent from legitimate organizations to potential/prospective customers, respectfully addressed emails (such *Dear Sir*, *Dear Madam*), etc., may not be personalized. The presence/absence of personalized information is one of the many features employed by our methodology. Hence, it provides one of the signals (not the whole signal) and also helps to narrow the search to detect phishing attacks.

The final named entity, extracted using CRF, is the location information. The motivation for use of the location information is its presence in a non-phishing email and usually absent in a phishing email. In the non-phishing email sample (shown previously), the location where organization is located, namely, ***San Jose***, **CA**, is present while it is absent in the phishing email. It must be noted that it is very easy for an attacker to include the location information in the phishing email. Thus, it is not as robust a feature as the other named entities.

3.1.2. *Stage I(b) – feature extraction (LDA)*

Topics comprise the second set of features extracted in the feature extraction stage (see Fig. 1). Group of words or phrases constitute a topic. For example, a "baseball" topic may comprise words/phrases such as "Yankees", "home run", "major league".

In *Stage 1(b)*, we make use of LDA to discover topics from a collection of phishing and non-phishing messages. The motivation for the use of LDA and some significant differences vis-à-vis PLSA employed in our earlier research (Ramanathan and Wechsler, 2012) for topics discovery is briefly discussed here, while the theory behind LDA is given in Appendix B. LDA is a topic modeling method, similar to PLSA, which employs a bag-of-words approach to discover hidden theme(s)/topics for given documents. Note, a topic model assigns each document a probability distribution over "topics", which are in turn a probability distribution over words/phrases. In PLSA, the latent (hidden) topics are modeled using a weighted likelihood approach instead of using full probability theory, i.e., PLSA is a non-Bayesian version of LDA. This leads to two problems using PLSA (Blei et al., 2003). First, the number of parameters in the PLSA model grows linearly with the size of the corpus, which leads to the problem of overfitting. Second, there is no robust method for the assignment of probabilities to documents outside the training set. LDA overcomes these problems by defining a generative process for each document wherein the topic distributions are assumed to have a Dirichlet prior.

Here, we apply LDA to discover hidden topics from phishing messages. The motivation for the use of LDA for phishing is illustrated through a sample phishing email from the online banking company CIMB Clicks. Words/phrases that comprise a ‘financial phishing’ topic are shown in bold and italicized.

3.1.2.1. Phishing email.

*“subject: CIMB **important** notice - account security **validation expired***

dear customer

your CIMB Clicks account security validation has expired,

*this maybe as a result of wrong or **incomplete data***

*entered during the **lastupdate**.*

*it’s **strongly required** that you should **validat** your*

***bank** account and **confirm** your **internet banking** records.*

***click** on the following **link**:*

http://www.ingelam.cl/respaldo/ingelam.php

fraud prevention unit

legal advisor

CIMB Clicks security dept. team.”

One of the motivations for using LDA is its robustness to changes in word usage. It is good at handling synonyms, different words with similar meanings. In a phishing email, attackers seek immediate attention from the user. It is evident from the above example that the ‘financial phishing’ topic comprises the word ‘***important***’. To seek user attention, attackers may use similar sounding words such as ‘*alert*’ (see ***PayPal*** phishing example), ‘*urgent*’, etc. Attackers may also choose ‘bank account’ vs. ‘checking account’, ‘rejected’ vs. ‘canceled’, ‘financial institution’ vs. ‘electronic payments association’, etc., and compose a phishing message targeting the same organization. An exact word based filter is not robust to such changes in word usage unlike LDA.

LDA is also robust to polysemy, words with different meaning in different context. For example, a ‘financial phishing’ topic comprises the word “***bank***” (see above example). The word “***bank***” in the context of “***river bank***” is a completely different topic. LDA is robust in discovering such differences.

LDA is robust in discovering the threatening theme in a phishing message that requires the user to act immediately and failure to do so will result in serious repercussions. In the ‘financial phishing’ topic, words/phrases that exhibit such theme include “***validation expired***”, “***incomplete data***”, “***strongly required***”, “***confirm***”, “***click***”, etc. This is to convince users to act on those emails by clicking the links contained in the email and filling out the form that require user credentials.

LDA is also robust in discovering topics that contain intentionally misspelled words and conjoined words. The “financial phishing” topic contains misspelled word ‘***validat***’ and conjoined word ‘***lastupdate***’. Attackers employ these techniques to avoid detection by exact word based filters. LDA is robust to such tricks by attackers.

The most powerful feature of LDA is its ability to discover multiple topics from a single document. One of the tricks attackers employ in phishing email is to include non-phishing content using white font color HTML attribute. A snippet of such an example is shown below.

3.1.2.2. Phishing email that contains non-phishing content in white font.

“<html>

....

subject: important notice

dear customer,

we recently reviewed your account, and suspect that your TD Canada

Trust online banking account might have been accessed by an

unauthorized third party.

....

</p>

<span style="FONT-FAMILY: ’Arial’,’sans-serif’;

***COLOR: white**; FONT-SIZE: 24pt">*

The candy business underwent a drastic change in the 1830 when

technological advances and the availability of sugar opened up the market.

The new market was not only for the enjoyment of the rich but also for the

pleasure of the working class as well. There was also an increasing market

for children. Confectioners were no longer the venue for the wealthy and

...

</html>”

In the above example, the “financial phishing” topic targeted to TD Canada Trust bank customer is followed by a non-phishing topic, “confectionary”. The non-phishing content (white font on white background) will not be visible to the user's naked eye. However, this email will pass through filters that treat the document as a whole as most words that are of non-phishing in nature, confectionery in this case, dominate words that are of phishing nature. Since the user will not see the non-phishing topic and the phishing topic sounds legitimate, the user will fall for the phishing attack. However, a server side filter using LDA will discover that there are two distinct topics in the email and assign different probabilities to each topic. Thus, we employ LDA to discover those hidden topics. Once topics are discovered using LDA, the document/topic probability distributions are used as a second set of features for building the phishing classifier.

3.2. Stage II – phishing classifier (AdaBoost)

The second stage involves building a robust classifier by employing the boosting method, AdaBoost (see Fig. 1). Boosting combines many weak and moderately accurate classifiers to build a robust and thus strong classifier for detecting phishing attacks. Boosting also helps to fuse heterogeneous features resulting in improved classification performance. Here, Adaboost is employed on combined feature sets of topic distribution probabilities obtained using LDA and named entities obtained using CRF to build a strong classifier for phishing detection. The motivation for combining LDA and CRF methods is as follows. LDA incorporates bag-of-words approach and hence it does not depend on the order of words in the document. CRF relies on preceding and succeeding words to label a word(s) as a particular entity, organization, name or location. While the fusion of LDA and CRF is done at the classification stage using AdaBoost, future work may involve named entity extraction using CRF providing prior probabilities for LDA to yield better and more efficient ways for topics discovery. Likewise, topics discovered using LDA may be used to do build a more focused named entity extraction using CRF.

3.3. Stage III – impersonated entity discovery (CRF)

Once the classifier classifies a particular message as phishing, the final stage involves automatic discovery of impersonated entities (see Fig. 1). In the phishing examples shown earlier, impersonated organizations include "*PayPal*", "*CIMB Clicks*", and "*TD Canada Trust*". The detection of phishing and subsequent blocking by an email provider helps to protect users from falling prey to identity theft. The automatic discovery of impersonated entities helps legitimate organizations to shut down the fake website before their users potentially fall for one, as their users may be using other email providers that may not have detected and blocked the same phishing messages. This keeps legitimate company's customers safe and secure, regardless of which systems they use, which in turn benefits the entity/company having long lasting and satisfied customers. Automatic discovery also helps companies to engage in partnerships with other companies to mutually exchange phishing information. As the impersonating organization discovery is akin to named entity extraction, CRF is employed for automatic discovery of such entities from phishing messages. The generic NER extraction using CRF, employed in the first stage, can extract more than one entity type but is not robust to discover impersonated organizations from phishing messages. Hence, a custom classifier, that is much more focused and specific for impersonated organizations discovery, is built by employing CRF and trained on phishing messages.

The methodology developed in this research not only identifies a phishing message but also identifies the organization that attackers impersonate in the message. The implementation of the corresponding multi-stage architecture is detailed in the next section.

4. Architecture

The architecture of the developed methodology is shown in Fig. 2. The major architectural components are (i) Parser, (ii) CRF Feature Extractor, (iii) LDA Feature Extractor, (iv) AdaBoost Classifier, and, (v) CRF Impersonated Entity Extractor. Each architectural component's functionality and their sub components will be elaborated below. The Parser component and TDF Matrix Builder sub-component were developed in our earlier research (Ramanathan and Wechsler, 2012). They are described here for the purpose of understanding the overall architecture. All the architectural components are implemented using Java programming language with the use of openly available software, when available.

(i) Parser: Raw email data is typically present in Multipart Internet Mail Extension (MIME) format. In our work, we use words and hyperlinks present in the body of the email to build LDA model and CRF feature extractor. Thus, the parser parses the email document and extracts body text, hyperlinks and words present in the body of the email. Parser consists of the following sub components:

MIME Parser: Parses email MIME message and extracts email headers and email body. The parser also extracts any hyperlinks present in the body. In a phishing email, these hyperlinks link to the phishing website.

HTML Parser: MIME message containing HTML documents are included as multipart/html subpart in the email body part. When the MIME parser detects a HTML subpart, it

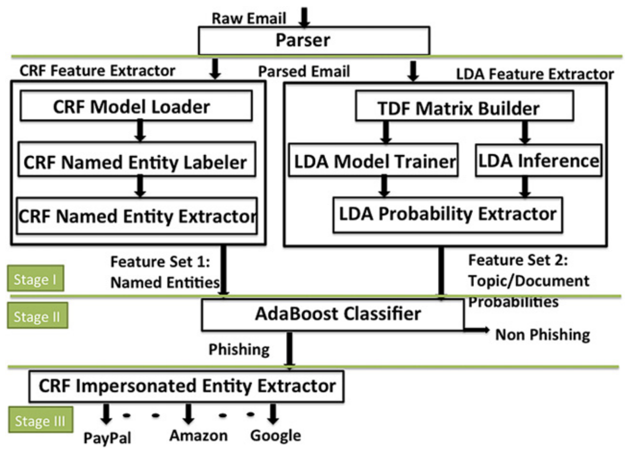

Fig. 2 – Architecture.

invokes the HTML parser to separate out text, style-sheets, hyperlinks and scripts. For the purpose of feature extraction, only text and hyperlinks are considered.

Tokenizer: This tokenizes text present in email body into separate words. Tokenizer utilizes white space (tabs, space, new lines) as token delimiters. The hyperlinks are tokenized after removing non-alpha-numeric characters.

(ii) CRF Feature Extractor: We employ the Conditional Random Field (CRF) to extract named entities. The LDA model does not consider named entities, such as names of people, organizations and locations. Since such proper names do not have any linguistic variation, LDA would not be useful to represent them as feature set. CRF is used to extract such named entities from the text of the email using the NER software written by Stanford's Natural Language Processing Group (Stanford NER, 2011). The CRF feature extractor includes the following sub components.

CRF Model Loader: Stanford's NER software comes with a pre-trained model that has been trained on CoNLL, MUC6, MUC7, and ACE datasets. These datasets are the comprehensive source of data containing news articles from United States and United Kingdom. This component loads the pre-trained model in memory to extract named entities from phishing and good emails.

CRF Named Entity Labeler: Given a new document, CRF named entity labeler component identifies and labels each word to most appropriate named entity. Each word is labeled as one of four entities, namely, location, organization, person or other. The labeling is limited to one entity per word. Some words may belong to more than one entity. For example, 'amazon' is not only an organization but also a rain forest, 'Bush' is a 'person', a 'politician' and also means a 'shrub'. The methodology employed here assigns an entity to a word based on the context in which the word appears and the assignment is limited to one entity, which is fine under most circumstances.

CRF Named Entity Extractor: Entities from the labeled document are extracted by this component. These entities serve as one set of features for building the classifier.

(iii) LDA Feature Extractor: This component builds an LDA topic model from input data consisting of phishing and non phishing data and extracts topic/document distribution probabilities for a given new document. These probability distributions serve as second set of features to build the classifier. This component consists of following sub-components:

TDF Matrix Builder: This component builds the term-document-matrix using words from the body of the email. A term-document matrix describes the frequency of terms that occur in a collection of documents. The rows of the matrix correspond to document (d_i) in the collection and the columns correspond to terms (w_j) that are present in those documents. The terms w_j belong to one of the part-of-speech tags (adjectives, adverbs, nouns and verbs). The matrix entries $n(d_i,w_j)$ denote number of times word w_j occurs in document d_i. Prior to building TDF Matrix the following pre-processing steps must be accomplished.

Stop Words Removal: Stops words are words that do not contain important significance for building the model. Some example stop words include 'the', 'at', 'like', etc. We remove stop words from all the tokenized email body text.

Stemming: Stemming is a method for removing inflexion endings from certain words. For example, word 'consigned', after stemming becomes 'consign'. Here we employed Porter's Stemming (PorterStemmer, 2006) algorithm to words in email body.

Dictionary Lookup: We employed WordNet (2006) dictionary to lookup words in dictionary. WorldNet database has Part of Speech (POS) extractor. It identifies verbs, nouns, adverbs and adjectives. Words found in WorldNet database forms part of the input for building TDF matrix.

Spell Checker: Attackers intentionally misspell words in a phishing email to avoid detection by standard spam filters. For words that are not found in WordNet database, Google Suggest API (2011) is utilized to retrieve words that are similar to the misspelled word.

Levenshtein Distance: Levenshtein Distance (2011) is a metric for measuring the amount by which two words differs. The metric is also called edit distance. It is the minimum edit operations required to transform one word to another. The edit operations include insertion, deletion and substitution of a new character. In a phishing email there are misspelled words, which after edit operation, is found in dictionary. Examples include "vuln'a'rability", "youaccounts", etc. Also, there are terms made of garbage characters that are never found in dictionary. We consider only misspelled words that can be corrected after certain edit operation. After obtaining the suggested words using Google API, Levenshtein distance is computed. Only those words whose edit distance is less than some configured threshold (default value of 5) are further included for building TDF matrix.

Using words, (specifically adjectives, adverbs, nouns and verbs), found directly in dictionary and edited words using Levenshtein's edit operation, the term-document-frequency matrix is created.

LDA Model Trainer: This component trains and builds a LDA topic model. The input to the model is the TDF matrix of the dataset. The trainer employs 90% of the data for building the model and remaining 10% for model's predictive performance. Stanford's Topic Modeling Toolbox (Stanford TMT, 2012) is employed to implement this component. The LDA trainer uses collapsed variational Bayes approximation algorithm (Asuncion et al., 2009), as it is computationally faster and leads to faster model convergence, to build the LDA model. LDA requires number of topics, K, to be specified at initialization similar to cluster analysis. In addition, LDA requires Dirichlet parameters, α, parameter of the Dirichlet prior on the per-document topic distributions, and β, parameter of the Dirichlet prior on the per-topic word distributions, to be specified upfront. The model performance was evaluated by computing perplexity (see Appendix B).

LDA Model Inference: In model inference, previously trained LDA model is employed to compute probability distributions on the new unseen dataset. Here, we use this component to compute per-document topic probability distributions and per-topic word distributions on the test dataset. Stanford TMT's model inference module is used to compute probabilities on the test dataset.

LDA Topic Probability Extractor: This extracts word/topic and topic/document distribution probabilities computed by the LDA model inference sub-component. Our method makes use of the topic/document distribution probabilities as the second set of features to build the classifier. By using these probability distributions instead of actual words, the classifier is expected to be robust in detecting phishing attacks.

(iv) AdaBoost Classifier: In order to build a robust phishing classifier we employ the AdaBoost algorithm. The features used to build the classifier are shown in Table 1. The boosting algorithm employs named entities obtained from the CRF model and per-document topic probability distributions obtained from the LDA model. The CRF model extracts named entities using a pre-trained model. Only proper names, locations, organizations and names of people, are included as one set of features. The per-document topic probability distributions of the LDA model that yielded the lowest perplexity are used as second set of features. AdaBoost is robust in building a classifier using these disparate feature sets. The classifier is built using the WEKA software (WEKA, 2009).

(v) CRF Impersonated Entity Extractor: This component extracts organization that the attacker impersonates in the phishing attacks. The component employs CRF to train and build a custom classifier for this purpose. The classifier that comes with Stanford NER is trained on corpus containing news articles that is not robust in extracting impersonating organizations from phishing messages. Hence, we build a custom classifier using phishing corpus specifically to extract impersonated organizations. Most of the core software component of Stanford NER was used to implement this component. Custom components for parsing, tokenizing, labeling and training were implemented in Java. As detailed in the earlier section, automatic extraction of impersonated organizations helps legitimate companies to enforce phishing website take down, which in turn protects their users.

Table 1 – Features for phishing classifier.

Feature	Description	Source
$P(z_1\|d_i)$	Probability that document d_i belongs to topic z_1	LDA
$P(z_2\|d_i)$	Probability that document d_i belongs to topic z_2	LDA
….	…	….
$P(z_k\|d_i)$	Probability that document d_i belongs to topic z_k	LDA
Location_1	Named entity – location	CRF
…..	….	…
Location_L	Named entity – location	CRF
Name_1	Named entity –name	CRF
….	…	…
Name_N	Named entity – name	CRF
Organization_1	Named entity – organization	CRF
….	…	…
Organization_O	Named entity – organization	CRF

The architecture developed here has two main novel contributions, (i) a phishing detection methodology that combines LDA and CRF to build a robust classifier using AdaBoost, and, (ii) impersonated entity extractor using CRF. Experiments conducted on public corpus to evaluate the architecture developed in this research and the architecture's performance will be reported in the next section.

5. Experiments

The performance of the multi-stage architecture is evaluated and experimental results are reported here. The architecture is evaluated using openly available standard datasets containing phishing and non-phishing data. Evaluation of the phishing classifier is done on email datasets. Evaluation of the automatic discovery of impersonated entity is done on phishing emails, phishing URLs and phishing websites.

5.1. *Datasets and data preparation*

Three publicly available email datasets were used to evaluate the phishing detection architecture: (i) ham (good) emails from SpamAssassin PublicCorpus (2006), (ii) phishing emails from the PhishingCorpus (2006), and, (iii) email archive containing spam and phishing emails from SPAM Archive (2012). One publicly available phishing URLs from PhishTank (2012) were used for automated email labeling, and, phishing URLs and accompanying phishing websites were used impersonated organizations discovery.

SpamAssassin PublicCorpus (2006) contains a total of 6047 messages, of which, 4150 messages are good and the remaining are spam. These messages were collected by the SpamAssassin project for the years 2002–2003 and made available to the research community. For evaluation, spam messages were not used (only 4150 good messages are used instead).

The dataset PhishingCorpus (2006) contains 4550 phishing emails. These emails were collected by an individual for the period 2004–2007 and donated to the research community. For evaluation, all the phishing emails from this corpus were used.

SPAM Archive (2012) contains emails collected by Bruce Guenter using various bait accounts since 1998. For the purpose of evaluation, we employed emails from January 2012 and February 2012. This accounted for approximately 87,000 emails. Most emails in SPAM Archive are spam messages. As SPAM Archive does not provide separate phishing emails, an automated process was performed here to isolate phishing messages from spam messages described as follows.

PhishTank (2012) provides confirmed phishing URLs that are verified and labeled as phishing by human experts. Two sets of phishing URLs from PhishTank were created for our experiments. One set included all phishing URLs for the year 2010. This accounted for approximately 196,000 URLs. The other set of phishing URLs included August 2011 through February 2012. This accounted for approximately 58,000 phishing URLs. The phishing websites corresponding to the second set of phishing URLs, captured using a web crawler were also utilized here for architecture evaluation. Furthermore, a list of confirmed phishing domains were downloaded from SURBL (2011). Using phishing URLs from the second set and phishing domains from SURBL, emails that contain a phishing URL or a phishing domain in SPAM Archive were separated into a separate phishing email corpus. This accounted for approximately 2200 phishing emails out of the total of 87,000 emails in SPAM Archive.

Thus, there is a total of 4550 phishing emails from PhishingCorpus, 4150 good emails from SpamAssassin Corpus, 2200 phishing emails from SPAM Archive and 84,800 spam emails from SPAM Archive. These emails were used to validate the phishing classifier. Number of phishing URLs in the dataset are 196,000 for the year 2010 and 58,000 for the years 2011–2012 and number of phishing websites are 58,000 for the years 2011–2012.

All the email messages were parsed using a MIME parser to separate email headers from email body. Multipart messages containing HTML parts were further parsed using a HTML parser to extract the body text and hyperlinks. For evaluation, only messages that contain either body text or hyperlinks were considered. Thus, messages without message body and messages that failed parser were not included for building models. For building models and validating phishing classifier, phishing emails included all messages from PhishingCorpus and separated phishing emails from SPAM Archive, and, non phishing emails included good emails from SpamAssassin and spam emails from SPAM Archive.

For building the impersonated organizations discovery model using CRF, the phishing data in the dataset used included 6750 phishing emails, 254,000 phishing URLs and 58,000 phishing websites. To build this model and verification using CRF, we needed datasets that has corresponding impersonated organizations identified upfront. Authors manually identified impersonated organization from 3000 phishing emails (2000 from PhishingCorpus (2006) and 1000 from SPAM Archive (2012)). Some fraction of the phishing URLs from PhishTank have their impersonated organizations pre identified. We considered 50,000 such phishing URLs from 2010 dataset and 30,000 phishing URLs from 2011 to 2012 dataset that had impersonated organization in them. The phishing websites included 30,000 websites from the year 2011 and 2012. The same phishing email message parsed that was used for evaluating phishing classifier was used for impersonated organizations discovery evaluation. Furthermore, phishing URLs were tokenized by removing non-alpha-numeric characters and phishing websites were parsed using the HTML parser to extract the text portion of website content. The tokenized phishing URLs and parsed phishing websites were used as well for evaluation of the impersonated entity discovery.

5.2. *Training and testing*

The generic named entity extraction using Stanford NER (2011) was performed using the pre-trained model included with the software. This pre-trained model was trained on corpus containing news articles that included CoNLL, MUC6, MUC7, and ACE datasets. The LDA topics discovery model was built by varying the number of topics '*K*' from 5 to 200. The Dirchlet prior probability distribution parameters α and β were initialized to 0.1. The maximum number of iterations for convergence was set to 500. The *k*-fold cross validation strategy was employed to build and evaluate the topics model, with a *k* value of 10. Thus, the model was built on 90% of the dataset and validated on the remaining 10%. This process was repeated 10 times and the average values are reported here.

The classifier model is built using AdaBoost algorithm. WEKA (2009) software is used to build the AdaBoost classifier. The ARFF file, format used by WEKA, is prepared using the combined topic features from LDA and named entity features from CRF. For an email message that does not contain a specific feature, missing value notation was used in the preparation of ARFF file. The *k*-fold cross validation was used to build the AdaBoost classifier. The number of folds that was used for the experiment was 10. The maximum number of iterations was set to 1000 and the AdaBoost weight threshold was set to 100. The weak learner for AdaBoost was Random Forest algorithm. Experiments were conducted on older (2006) and newer (2012) datasets to evaluate the temporal robustness. As the pre-trained CRF model is not robust enough to discover impersonated entity (organizations), we built a custom model to perform discovery, using CRF, by employing just phishing messages. Impersonated organizations discovery models were built by using different proportions (50/50 split) of the same dataset. Models were also built using different years data to evaluate temporal robustness and different types of data (phishing emails, phishing URLs, phishing websites) to evaluate robustness to data types.

5.3. *Performance measures*

The performance of the LDA topics model is evaluated using perplexity. The perplexity for an LDA model is computed using the equation given in Appendix B. The model with the lowest perplexity is the one that generalizes well for classifying new/unseen data. The classification performance of phishing classifier is evaluated using the standard measures of performance described as follows. As it is a binary classification problem, the task is to learn how to classify unseen examples into one of two categories, positive categories and negative categories, True Positive (TP) means the actual and predicted categories are positive, True Negative (TN) means actual and predicted categories are negative, False Positives (FP) means the predicted should have been negative instead classified as positive, and. False Negatives (FN) means predicted should have positive instead classified as negative. Commonly used performance metrics in classification problems are accuracy, precision, recall, specificity and F measure. They are defined as follows. Accuracy is a measure of how accurate the learned system makes prediction on unseen test instances ($\text{Accuracy} = (\text{TP}+\text{TN})/(\text{TP}+\text{FP}+\text{TN}+\text{FN})$). Precision is defined as the proportion of true positives against all the positive results ($\text{Precision} = \text{TP}/(\text{TP}+\text{FP})$). Recall is the ratio of number of instances correctly classified to total number class instances ($\text{Recall} = \text{TP}/(\text{TP}+\text{FN})$). Specificity measures the proportion of negatives that are correctly identified ($\text{Specificity} = \text{TN}/(\text{TN}+\text{FP})$). F measure is the harmonic mean of precision and recall estimates ($F = 2^*(\text{Precision}\times\text{Recall})/(\text{Precision}+\text{Recall})$). ROC curve is a plot of true positive rate vs. false positive rate. Each point on the ROC curve represents different tradeoff between false positives and false negatives. The slope of the line tangent to curve is defined as the cost ratio. If the two ROC

Table 2 – LDA model performance.

Number of topics	Perplexity	Computation time (min)
Dataset – 4.5 K phishing (PhishingCorpus, 2006), 4.1 K non-phishing (SpamAssassin, 2006)		
5	553.71	1
10	433.36	1
50	260.36	3
100	245.73	6
200	232.27	15
Dataset – 2.2 K phishing, 84.8 K non-phishing (SPAM Archive)		
200	873.12	65

curves do not intersect, it implies that one method dominates the other method. The two-dimensional depiction of classifier performance in a ROC curve is reduced to single scalar value representing expected performance by computing the area under the ROC curve (AUC). The AUC measure has an important statistical property. The AUC of a classifier is equal to the probability that a classifier will rank a randomly chosen positive example higher than the randomly chosen negative example. The performance measure for evaluation of the impersonated organizations discovery model is the fraction of phishing messages in which impersonated organization was correctly discovered.

5.4. *Results*

Results from experiments conducted to evaluate the multi-stage architecture are presented in Tables 2–4. In Table 2, the performance of the LDA topics model is shown. The topics model was evaluated on older (2006) and newer (2012) to show the temporal robustness of LDA. As it can be seen, the perplexity for a 200 topics model is 232.27 on year 2006 dataset and 873.12 on year 2012 dataset. As the perplexity did not reduce significantly when number of topics was increased, the 200 topics model was used to build the phishing classifier. The phishing classifier was built using the combined feature set, the topics distribution probabilities from LDA and named entities extracted using CRF. The classifier was built using AdaBoost with a Random Forest weak learner. Results from the 10-fold cross validation is presented in Table 3. The AdaBoost classifier was built for varying proportions of phishing email in the dataset, 50%, 40%, 30%, 20% and 10%. The classification *F*-measure for the combined feature set obtained for 50%, 40% and 30% splits were 0.961, 0.987 and 0.988 respectively. The area under the ROC measure (AUC) varied from 0.979 to 1.0 for varying proportions of phishing email in the dataset. When the proportion of phishing emails in the dataset is less than or equal to 20%, the AdaBoost classifier yields perfect classification (i.e., no misclassification). Thus, it shows the effect of combined topic features and named entities as feature sets and applying boosting in overall performance improvement. Results were obtained on different year data to show the temporal robustness of the classifier. We conducted experiments on a newer (Jan 2012–Feb 2012) public email corpus that included newer phishing attacks that target social networking sites (such as Facebook) and online gaming sites (such as Sulake) that are not part of the year 2006 dataset. The classification performance in this dataset also yielded *F*-measure of 100%, thus showing robustness of the phishing classifier.

The results from the impersonated entity discovery model are shown in Table 4. These results were obtained by training a model using CRF that consisted of only phishing messages. Models were trained and tested on different data types (phishing emails, phishing URLs, phishing websites) to show the robustness of our approach to disparate data types. In addition, we also show the temporal robustness by training and testing on different years data. As it can be seen from Table 4, the best discovery was achieved when trained and tested on email datasets. The best discovery was 88.1%, i.e., the model was able to discover impersonated organization in 88.1% of the messages that were tested, when trained on 2006 emails and tested on year 2012 emails. Our approach also yielded discovery rate of 76.1% on phishing URLs and 81.6% on phishing websites. The reason for reduced discovery rate on URLs and websites as compared to emails was due to CRF's reliance on words that precede and succeed and well formed sentences to discover the impersonated entity. We present next a brief overview of autonomic computing.

Table 3 – Classification (using CRF, LDA, AdaBoost) performance.

% of phishing emails	True positive rate (TPR)	False positive rate (FPR)	Precision	Recall	F-measure	Area under ROC (AUC)
Training/testing strategy: 10-fold cross validation						
Data source: PhishingCorpus (2006), SpamAssassin, 2006						
Max data: 4.1 K phishing, 4.1 K non-phishing						
Weak learner for AdaBoost: random forest						
50%	0.961	0.039	0.961	0.961	0.961	0.979
40%	0.987	0.013	0.987	0.987	0.987	0.997
30%	0.988	0.012	0.988	0.988	0.988	0.997
20%	1.0	0.0	1.0	1.0	1.0	1.0
10%	1.0	0.0	1.0	1.0	1.0	1.0
Training/testing strategy: 10-fold cross validation						
Data source: SPAM Archive (2012)						
Max data: 2.2 K phishing, 84.8 K non-phishing						
Weak learner for AdaBoost: random forest						
2.5%	1.0	0.002	1.0	1.0	1.0	0.996

Table 4 – Impersonated entity discovery (using CRF) performance.

Training	Testing	% message with correctly discovered impersonated entity	% message with incorrectly discovered impersonated entity
Data: phishing email Year: 2006 Size: 1 K Source: PhishingCorpus	Data: phishing emails Year: 2006 Size: 1 K Source: PhishingCorpus	86.2	0.0
Data: phishing email Year: 2006 Size: 2 K Source: PhishingCorpus	Data: phishing emails Year: 2012 Size: 1 K Source: SPAM Archive	88.1	0.0
Data: phishing URLs Year: 2010 Size: 25 K Source: PhishTank	Data: phishing URLs Year: 2010 Size: 25 K Source: PhishTank	74.5	0.0
Data: phishing URLs Year: 2010 Size: 50 K Source: PhishTank	Data: phishing URLs Year: 2011–2012 Size: 30 K Source: PhishTank	76.1	0.0
Data: phishing websites Year: 2011–2012 Size: 15 K Source: crawled pages of PhishTank URLs	Data: phishing websites Year: 2011–2012 Size: 15 K Source: crawled pages of PhishTank URLs	81.2	0.0
Data: phishing websites Year: 2011 Size: 23 K Source: crawled pages of PhishTank URLs	Data: phishing websites Year: 2012 Size: 7 K Source: crawled pages of PhishTank URLs	81.6	0.0

6. Autonomic computing

The evaluation of the multi-stage research methodology and the results reported in Section 5 are limited to offline processing. Additional critical issues to consider are (i) how the developed methodology can operate in real-time, (ii) how can it adapt to changes in phishing attacks, and (iii) how can it automatically tune itself. One approach to achieving these goals is using Autonomic Computing. Autonomic computing is an approach to self-managed computing system with a minimum human interference. It is an initiative started by IBM in early 2001 (Autonomic Computing, 2012). This initiative was developed to overcome the growing complexity in managing large scale computing systems. The system that is autonomic makes decisions on its own using high level operating policies. The operating policies are created by the system administrator. The four key functional components of autonomic computing are self-optimization, self-protection, self-configuration and self-healing. Self-optimization involves automatic monitoring and control of resources utilized by the system that ensures system operates in optimal fashion meeting desired performance measures. Self-protection requires identifying attacks targeted to bring the system down and protecting the system from such attacks in a proactive manner. Self-configuration requires the system to be able to configure itself upon start-up, restart, shut down, etc. in an automatic fashion. Self-healing requires the system to automatically discover faults due to hardware or software and automatically correct such faults by applying software upgrades and software patches. We next present a framework for making our methodology autonomic.

7. Self-managing shield

We propose a framework, called self-managing shield, to make the developed multi-layered methodology autonomic (see Fig. 3). The scope of this framework is limited to email processing. The framework consists of four functional modules, which includes SMTP server, online evaluator, offline modeler and an autonomic engine. We evaluate this framework using a simulated mail client, and report processing times.

7.1. *SMTP server*

Inbound emails from email clients are processed by the SMTP server. Email providers typically have several thousand instances (processes) of SMTP server. For the purpose evaluation in this research we employed open source software, Postfix (Postfix, 2012).

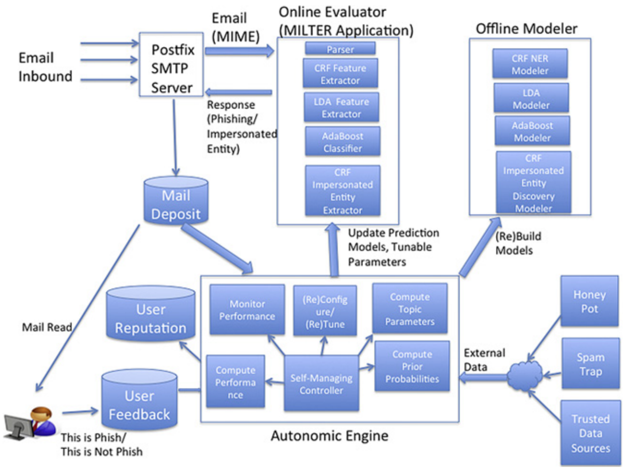

Fig. 3 – Self-managing shield.

7.2. *Online evaluator*

The online evaluator module is the core module that is responsible for classifying email messages as phishing (or not phishing), and extracting impersonated entity, if it is a phishing message. In this framework, we built a mail filter (MILTER, 2012) application, which is a plug-in interface supported by the open source, Postfix. There exist several MILTER applications such as to verify authenticity of the sender, block unwanted emails, etc. Here, we have developed our online evaluator using MILTER to classify incoming emails. Online evaluator was implemented as a multi-threaded application, spawning a thread for each incoming request. The application gets the entire mail message in MIME format from the SMTP server. This module implements all the functional components such as parser, CRF and LDA feature extractors, AdaBoost classifier and CRF impersonated entity extractor. Prediction models are loaded in memory to optimize the processing time for classification. Once this module classifies the message and extracts the impersonated entity, a response containing the information is sent to Postfix SMTP server.

7.3. *Mail deposit*

The SMTP server deposits the mail and records the response. For the purpose of evaluation, this response was written to disk. In a real email system, there will be databases to store incoming emails and response.

7.4. *Offline modeler*

Models are built using components in the offline modeler. This module is not in the critical path that process inbound email messages i.e., the processing time for inbound email processing will not be affected by this module and, hence, called offline. Prediction models (LDA, CRF) are built periodically upon initiation by the autonomic engine.

7.5. *Autonomic engine*

The core intelligence of (i) initiating the offline modeler, (ii) computing actual performance and expected performance, (iii) controlling the frequency of prediction model update in the online evaluator, is done by the autonomic engine.

The self-protection aspect of autonomic engine requires robust prediction models that are proactive. As the attacker's motivation and mode of operation changes over time driven by technology changes such as dial-up to broadband, email based communication to social networking centric communication, plain text messages to HTML enriched messages, password based authentication to openID authentication, etc., the system should have robust prediction models that should observe such changes and predict future attacks before it occurs. This framework plans to achieve that by gathering external data from spam trap, honeypots and other data sources, and use that data to update prediction models, if necessary. Spam trap and honeypots setup bait accounts that attracts attackers. By using messages from these sources, we can see if there is a phishing campaign that is imminent.

The self-configuration aspect of autonomic engine requires system and model parameters be updated without human involvement. Some of the parameters that require self-configuration include number of topics K, prior probability estimates and prediction models (LDA, CRF Named Entity, CRF Impersonated Entity). The autonomic engine plans to incorporate self-configuration by (i) (re)computing prior probability estimates for the LDA model using external data (ii) automatically determining number of topics K using external data sources, (iii) automatically determining if new topics have emerged by varying number of topics dynamically and checking the performance measures, (iv) checking if the CRF impersonated entity discovery model adapts to the formation of new entities, including merge and split of existing entities. For example, to update if number of topics K needs to be adjusted, the performance measure perplexity is computed with and without external new data and by varying K. If the perplexity improves with a change, that change is adopted. The updated prediction models and model parameters are automatically updated in the online evaluator for subsequent message processing.

The self-optimization aspect of autonomic engine requires system to continually operate above desired threshold. The autonomic engine plans to continually optimize misclassification rates based on response from user feedback and external data sources. Most email clients have feedback mechanism in place to report issues when a good email shows up in spam folder or when a phishing email shows up inbox. Only feedback from users with good reputation is included for performance evaluation. This is achieved by using a user reputation database. The framework will use existing user feedback and the response recorded in mail deposit to determine if the classification label (phishing or non phishing) predicted by online evaluator agrees (or disagrees) with user and then computes actual performance. If the actual performance does not match the predicted model performance (from previous building of prediction models) or below the desired threshold, the samples that are misclassified are fed back for re-training models using offline modeler. Prediction models are built until misclassified samples are classified correctly. When the predicted performance improves, the models are updated in the online evaluator for subsequent message processing. The newer prediction models are refreshed/reloaded in the memory of the online evaluator.

The autonomic aspects presented here are isolated to optimizing phishing classification performance. There are self-tuning and self-optimization that addresses other performance measures such as message throughput, CPU utilization, energy usage, etc. These aspects are outside the scope of this research.

7.6. *Computational efficiency*

In order to evaluate the performance of self-managing shield, we built an email client simulator that connects to the SMTP server (Postfix, 2012), sends a MIME message, process the response and disconnects. As our goal is to verify if our proposed methodology is computationally efficient for real-time processing, only the 'online evaluator module' performance times are recorded and reported in Table 5. The client is a Mac OS X machine (version-10.6.5, CPU-2.66 GHz Intel Core i7, RAM-4 GB). The 'online evaluator' is run on a Cent OS machine (version-linux 2.6.18 kernel, CPU-1.99 GHz Intel Core 2 Duo CPU, RAM-4 GB). The SMTP Server is run on a different Cent OS machine with the same configuration. We simulated a steady workload of 1200 messages per second (equates to approximately 100 million messages per day) and recorded the performance times of the online evaluator. The prediction models were loaded in memory to avoid hitting the database or the disk. The online evaluator spawned a thread for each incoming request with a maximum configurable thread pool of 1000. The computation times of individual components of the online evaluator and the total processing times are tabulated in Table 5. The average per message processing time recorded was 42 ms. This would be the additional latency that our methodology will introduce for filtering out email messages. The CPU utilization of the online evaluator was 8%. Large email service providers process billions of email messages. These providers will also have thousands of machines to handle that load. Additional experiments to simulate other workload patterns (such as burst traffic, different request distributions, etc.) are left for future research. Though our evaluation was done on a small scale, it is evident that our methodology is computationally efficient and introduces a very low latency to email message processing.

Table 5 – Processing time for online evaluator.

Workload: 1200 messages/second
Average CPU utilization: 8%

Component	Average per message processing time (milliseconds)
Parser	6
NER Extractor (CRF)	8
Topic Extractor (LDA)	18
Classifier (AdaBoost)	3
Impersonated Entity Extractor (CRF)	7
Total	**42**

8. Discussion

In this section, we first present the strengths of our proposed research methodology in comparison to our earlier research (Ramanathan and Wechsler, 2012). Next, we discuss how the architecture can be expanded for future research.

8.1. *Strengths of the proposed research methodology*

We present here a comparison of the multi-stage architecture with phishGILLNET (Ramanathan and Wechsler, 2012). The classification performance obtained by the phishing classifier (*F*-measure of 100%) was also obtained by our earlier research, phishGILLNET (Ramanathan and Wechsler, 2012). While phishGILLNET (Ramanathan and Wechsler, 2012) obtains the *F*-measure using expensive co-training, the current research obtains the same performance by combining LDA and CRF.

LDA does not consider ordering of words while CRF does consider order. As CRF and LDA are complementary to each other this yields robust results. Furthermore, LDA overcomes the limitations of PLSA employed by phishGILLNET (Ramanathan and Wechsler, 2012) (see section 3).

In addition, the proposed methodology can discover impersonated entities. Results for impersonated entity discovery show that our methodology discovers entity from disparate data sources, phishing emails, phishing URLs and phishing websites. By employing CRF, our methodology is able to discover newer entities (such as Facebook, Sulake) that were not present in the 2006 corpus. Thus, our methodology is robust in impersonated entity discovery, as attackers mostly target entities that are popular and currently successful. These entities could change from time to time.

Our methodology is also able to discover variations of a given entity, such as, paypal, paipal, paypa1, etc., which attackers employ while creating phishing URLs to resemble a legitimate entity.

Our methodology is domain neural. It can be employed to detect phishing attacks and discover impersonated entity at social networking posts (Facebook, Twitter, etc.), instant messages, chat, blog posts, etc. As long as the content is available in text, MIME and HTML formats, this architecture can handle all of them.

8.2. *Future enhancements*

Future enhancements to our methodology should consider the following:

8.2.1. *Robust parser*
We consider only messages that we were able to successfully parse using MIME and HTML parser. We also accounted for only text and hyperlinks present in the "body" of those messages. Some phishing messages could be in image form and some messages, in an adversarial fashion, are constructed to fail parsing. Future work should expand on the architecture advanced here and allow for such messages. While we considered only text attachments, future research should also consider email attachments in formats such as word document, Adobe PDF, etc.

8.2.2. *Integrated CRF and LDA*

The multi-stage architecture described in this paper, independent of each other, extracts named entities using CRF and discovers topics using LDA. The outputs from the first stage are then combined using AdaBoost for the classification stage. LDA incorporates in its model two parameters, α, parameter of the Dirichlet prior on the per-document topic distribution and, β, parameter of the Dirichlet prior on the per-word topic distribution. Choosing the right priors helps to discover better topics more efficiently and minimizes the chance of getting stuck in local optima. One avenue for future research is to estimate prior probabilities for LDA model using CRF. Knowing the impersonated entity could also help to discover topics. For example, say PayPal is the impersonated entity in the majority of the phishing messages. Since PayPal is an online payment service, the priors can be appropriately chosen to discover online payment phishing topics from the dataset. LDA does not rely on order of words while CRF does. Thus, and integrated LDA and CRF has the potential to yield better results.

8.2.3. *Scalability*

One of the issues to consider for future evaluation is scaling. Our research has shown that we were able to get good results on a smaller setup with 2012 dataset. A typical email server process billions of message a day. As the goal of this research is to implement this architecture as a server side filter on a large scale email system, further evaluations should be conducted to determine how this architecture performs. One way to evaluate scalability is to implement the architecture using MapReduce (2012) framework on a distributed cluster computing platform.

9. Conclusions

A robust multi-stage phishing detection and impersonated entity discovery methodology is developed in this research. CRF is used to extract named entities. Named entities are used as one set of features. Topics are discovered using LDA. The per-document topic probability distributions obtained from the LDA topic model are used as a second set of features. The combined probability estimates and named entities are used to build a strong classifier using AdaBoost. The 10-fold cross validation is employed to build and validate the phishing classifier. The boosting method resulted in no misclassification on the test set when the percentage of phishing emails is less than 20%.

On messages that are classified as phishing, the impersonated entity is discovered by building a model that employs CRF. Impersonated entity is discovered from disparate data types (phishing emails, phishing URLs and phishing websites). Results show that the discovery rate was highest (88.1%) on phishing emails. This is due to CRF's dependence on words that precede, words that succeed, and well-formed sentences to perform robust discovery. The discovery rate in phishing URLs was the lowest (76.1%).

The unique combined approach that employs CRF and LDA to yield perfect classification for phishing detection, and also enables impersonated entity discovery using CRF, are the novel contributions of this research. Robust content-driven phishing detection developed in this research helps service providers to implement this architecture as a server side filter to eliminate phishing messages before they get to the user. Upon discovery of an impersonated entity in a phishing message, an organization can communicate the phishing attack to the entity that is the target of the attack. This in turn helps that entity to initiate phishing website take down and other counter measures, thereby protecting customers. A simulated experiment also shows that our approach is computationally efficient and introduces a very small latency to email message processing.

Appendix A. Conditional Random Field (CRF)

Given a vector of input features $x = \{x_1, x_2, \ldots x_n\}$ and a vector of output variables $y = \{y_1, y_2, \ldots y_m\}$, a generative model estimates the joint probability distribution $p(y, x)$ and a discriminative model estimates the conditional probability distribution $p(y|x)$. The main difference between the two is that the discriminative model does not include a model of $p(x)$. The Conditional Random Field (CRF) is a discriminative, uni-directed graphical model that models the probability distribution $p(y|x)$. The nodes can be divided into two disjoint sets x and y, the observed and output variables, respectively. Lafferty (2001) defines the probability of a particular label sequence y given observation sequence x to be a normalized product of potential functions, each of the form, expressed as follows:

$$\exp\left[\sum_j \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \mu_k s_k(y_i, x, i)\right] \tag{A.1}$$

where, $t_j(y_{i-1}, y_i, x, i)$ is the transition feature function of the entire observation sequence and labels at positions i and $i-1$ in the sequence; $s_k(y_i, x, i)$ – state feature function of the label at position i, and the observation sequence; LAMBDA$_j$, MU$_k$ – parameters estimated from training data.

In order to simplify the above equation, the following two additional expressions are introduced:

$$s(y_i, x, i) = s(y_{i-1}, y_i, x, i) \tag{A.2}$$

$$F_j(y, x) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, x, i) \tag{A.3}$$

In the above expressions, $f_j(y_{i-1}, y_i, x, i)$ is either a state function $s(y_{i-1}, y_i, x, i)$ or a transition function $t(y_{i-1}, y_i, x, i)$. Thus, the probability of y given observed variable x can be written as follows:

$$p(y|x, \lambda) = \frac{1}{Z(x)}\exp\left(\sum_j \lambda_j F_j(y, x)\right). \tag{A.4}$$

$Z(x)$ is called the normalization factor. The CRF model parameters are estimated using log likelihood function, expressed using the following expression:

$$L(\lambda) = \sum_k \left[\log \frac{1}{Z(x^{(k)})} + \sum_j \lambda_j F_j(y^{(k)}, x^{(k)}) \right]. \quad (A.5)$$

The parameters cannot be determined analytically. Instead, parameters of CRF are obtained using an iterative procedure such as iterative scaling or gradient methods (Lafferty, 2001).

Appendix B. Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a natural language processing method that discovers topics from a collection of documents. Documents are represented as random mixtures over latent topics and each topic is represented by a distribution over words. LDA (Blei et al., 2003) is built on the foundations of PLSA (Hoffman, 2001). Given parameters α and β, the LDA model is expressed as the joint probability distribution of a topic mixture θ, a set of N topics z, a set of N words w using the following expression:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \quad (B.1)$$

where, $p(z_n|\theta)$ is θ_i for the unique i such that $z_n = 1$. Integrating over θ and summing over z, the marginal distribution of a document is expressed as follows:

$$p(w|\alpha, \beta) = \int p(\theta|\alpha) \left[\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right] d\theta \quad (B.2)$$

The probability of a corpus D, computed from the product of marginal probabilities of individual documents, is expressed as follows:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left[\prod_{n=1}^{N_d} \sum_{z_{dn}} (p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \right] d\theta_d \quad (B.3)$$

The parameters α and β are corpus level parameters assumed to be sampled once in the process of generating a corpus. The document level variables θ_d are sampled once per document. The word level variables w_{dn} and z_{dn} are sampled once for each word in the document. Several algorithms have been developed to solve LDA that requires estimation of the posterior probability distribution of hidden topic variables. It includes expectation-maximization algorithm, expectation-propagation algorithm, and collapsed Gibbs sampling (Asuncion, 2009). The performance of the LDA model on the test dataset is evaluated using perplexity. Perplexity for a LDA model is computed using the following expression:

$$Perplexity = \exp\left[-\frac{\sum_d \log p(w_d)}{\sum_d N_d} \right] \quad (B.4)$$

where, N_d the number of words in document d and $p(w_d)$ is the per-word topic probability distribution of document d.

Appendix C. AdaBoost

AdaBoost is one of the powerful machine learning algorithms developed by Freund and Schapire (1999). It is an algorithm for constructing a strong classifier as a linear combination of simple weak classifiers. The algorithm is detailed as follows:

Given input training data (x_1, y_1), (x_2, y_2),….,(x_m, y_m), where x_i belongs to feature space X and y_i belongs to label set $Y = \{-1,+1\}$ and 'm' is the number of training samples:

$$\text{Step 0: Initialize weights for the first iteration}, D_1(i) = 1/m \quad (C.1)$$

For iteration index $t = 1,\ldots.T$, where T is the maximum number of iterations,

Step 1: Train a weak learner using distribution D_t.

Step 2: Obtain weak hypothesis:

$$h_t : X \rightarrow \{-1, +1\} \quad (C.2)$$

with error:

$$\varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i) \quad (C.3)$$

Step 3: Compute:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \quad (C.4)$$

Step 4: Updates weights for this step:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t} \quad (C.5)$$

where Z_t is the normalization factor.

The final strong classifier, which is a weighted majority of W weak hypothesis, is given as:

$$H(x) = \text{sgn}\left(\sum_{w \in W} \alpha_w h_w(x) \right) \quad (C.6)$$

The basic AdaBoost algorithm has been further been extended to create two variations, namely, AdaBoost.MH and AdaBoost.MR. AdaBoost.MH optimization criteria is to minimize the Hamming loss, while that of AdBoost.MR is to minimize the ranking loss. Also, the original binary classification AdaBoost algorithm, has been expanded for multi-class classification problems in AdaBoost.M1 and AdaBoost.M2.

REFERENCES

APWG. Anti phishing working group [accessed 30.09.12], http://www.antiphishing.org; 2012.

Asuncion A, Welling M, Smyth P, Teh YW. On smoothing and inference for topic models. In: Proc. of the twenty-fifth annual conference on uncertainty in artificial intelligence, Corvalis, OR; 2009. p. 27–34.

Autonomic Computing [accessed 20.06.12], http://en.wikipedia.org/wiki/Autonomic_computing; 2012.

Blei DM, Ng AY, Jordan MI. Latent Dirichlet allocation. Journal of Machine Learning Research 2003;3:993–1022.

CallingID. Your protection from identity theft, fraud, scams and malware. Available from: http://www.callingid.com/Default.aspx; 2011 [accessed 21.07.11].

CloudMark. Available from: http://www.cloudmark.com/en/products/cloudmark-desktopone/index; 2011 [accessed 26.07.11].
Cordero A, Blain T. Catching phish: detecting phishing attacks from rendered website images. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.9084&rep=rep1&type=pdf; 2006 [accessed 26.07.11].
Cortez P, Correia A, Sousa P, Rocha M, Rio M. Spam filtering using network-level properties, advances in data mining, applications and theoretical aspects. LNCS 2010;6171:476–89.
DNSBL. Spam database lookup. Available from: http://www.dnsbl.info/; 2011 [accessed 21.07.11].
DomainKey. Tools and library for email servers and clients. Available from: http://domainkeys.sourceforge.net/; 2006 [accessed 21.07.11].
eBay Toolbar. Available from: http://download.cnet.com/eBay-Toolbar/3000-12512_4-10153544.html?tag=contentMain;downloadLinks; 2007 [accessed 21.07.11].
Feamster N. Fighting spam, phishing, and online scams at the network level. In: Proceedings of the 4th Asian conference on internet engineering; 2008. p. 39–40.
Fette I, Sadeh N, Tomasic A. Learning to detect phishing emails. In: Proceedings of the 16th international conference on world wide web; 2007. p. 649–56.
FirePhish. Anti-phishing extension. Available from: https://addons.mozilla.org/en-US/firefox/addon/firephish-anti-phishing-extens/; 2006 [accessed 26.07.11].
Freund Y, Schapire R. A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence 1999;14:771–80.
Hofmann T. Unsupervised learning by probabilistic latent semantic analysis. Machine Learning 2001;42:177–96.
IE Phishing Filter. Available from: http://support.microsoft.com/kb/930168; 2011 [accessed 21.07.11].
Jagatic TN, Johnson TN, Jakobsson M, Menczer F. N.A. Social phishing. Communications of the ACM 2007;50:94–100.
Lafferty J. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Machine learning-international workshop; 2001. p. 282–9.
Levenshtein Distance. Available from: http://en.wikipedia.org/wiki/Levenshtein_distance; 2011 [accessed July 26.07.11].
MailFrontier. Phishing IQ test – UK edition. Available from: http://survey.mailfrontier.com/survey/phishing_uk.html; 2011 [accessed 21.07.11].
MapReduce [accessed 01.06.12], http://en.wikipedia.org/wiki/MapReduce; 2012.
MILTER [accessed 10.09.12], http://www.milter.org; 2012.
National Data. Deter. Detect. Defend. Avoid ID theft. Available from: http://www.ftc.gov/bcp/edu/microsites/idtheft/reference-desk/national-data.html; 2009 [Accessed 21.07.11].
Netcraft. Anti-phishing toolbar. Available from: http://toolbar.netcraft.com/; 2011 [accessed 26.07.11].
PC World. Google says phishers stole e-mail from US officials, others. Available from: http://www.pcworld.com/businesscenter/article/229202/google_says_stole_email_from_us_officials_others.html; 2011 [accessed 21.07.11].
PhishingCorpus. Available from: http://monkey.org/~jose/wiki/doku.php; 2006 [accessed 21.07.11].
PhishTank. Available from: http://www.phishtank.com; 2012 [accessed 01.06.12].
PorterStemmer. The porter stemming algorithm. Available from: http://tartarus.org/~martin/PorterStemmer/; 2006 [Accessed 26.07.11].
Postfix, http://www.postfix.org; 2012 [accessed 10.09.12].
Ramanathan V, Wechsler H. phishGILLNET – phishing detection methodology using probabilistic latent semantic analysis, AdaBoost and co-training. EURASIP Journal on Information Security 2012;2012:1.
Robila SA, Ragucci JW. Don't be a phish: steps in user education. In: Proceedings of the 11th annual SIGCSE conference on innovation and technology in computer science education; 2006. p. 237–41.
Sender ID. Email authentication technology. Available from: http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx; 2006 [accessed 21.07.11].
Snort. Network intrusion prevention and detection system. Available from: http://www.snort.org/; 2011 [accessed 21.07.11].
SPAM Archive. Public email corpus. Available from: http://untroubled.org/spam/; 2012 [accessed 30.04.12].
SpamAssassin. Open source email filter. Available from: http://spamassassin.apache.org/; 2011 [accessed 19.07.11].
SpamAssassin PublicCorpus. Available from: http://spamassassin.apache.org/publiccorpus/; 2006 [accessed 21.07.11].
Sperotto A, Vliek G, Sadre R, Pras A. Detecting spam at the network level, the internet of the future. LNCS 2009;5733:208–16.
SpoofGuard. Available from: http://crypto.stanford.edu/SpoofGuard/; 2004 [accessed 21.07.11].
Stanford NER. Named entity recognition and information extraction. Available from: http://nlp.stanford.edu/ner/index.shtml; 2011 [accessed 21.07.11].
Stanford TMT. Topic modeling toolbox. Available from: http://nlp.stanford.edu/software/tmt/tmt-0.4/; 2012 [accessed 30.04.12].
SURBL. URI Reputation data. Available from: http://www.surbl.org/; 2011 [accessed 21.07.11].
URIBL. Realtime URI blacklist. Available from: http://www.uribl.com/; 2011 [accessed July 2011].
WordNet. A lexical database for English. Available from: http://wordnet.princeton.edu/; 2006 [accessed 26.07.11].
WEKA. Data mining with open source machine learning software in Java. Available from: http://www.cs.waikato.ac.nz/ml/weka/; 2009 [accessed 21.07.11].

Venkatesh Ramanathan earned his PhD in Computer Science from George Mason University, Fairfax, VA, USA, in 2012. His expertise and research interests include machine learning, natural language processing, biometrics, information retrieval, computer security, and privacy.

Harry Wechsler is a Professor at the Department of Computer Science, George Mason University, Fairfax, VA, USA. He earned his PhD in Computer Science from the University of California, Irvine, in 1975. His research interests include biometrics, computer vision, data mining, human–computer interfaces, intelligent systems, data mining, machine learning, pattern recognition, computer security, and privacy. Dr. Wechsler has authored over 280 scientific papers and several books. He is a Fellow of IEEE and IAPR.