Beyond the Lock Icon: Real-time Detection of Phishing Websites Using Public Key Certificates

Zheng Dong*, Apu Kapadia*, Jim Blythe† and L. Jean Camp*
*School of Informatics and Computing
Indiana University
Bloomington, IN 47405
†Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292

***Abstract*—We propose a machine-learning approach to detect phishing websites using features from their X.509 public key certificates. We show that its efficacy extends beyond HTTPS-enabled sites. Our solution enables immediate local identification of phishing sites. As such, this serves as an important complement to the existing server-based anti-phishing mechanisms which predominately use blacklists. Blacklisting suffers from several inherent drawbacks in terms of correctness, timeliness, and completeness. Due to the potentially significant lag prior to site blacklisting, there is a window of opportunity for attackers. Other local client-side phishing detection approaches also exist, but primarily rely on page content or URLs, which are arguably easier to manipulate by attackers. We illustrate that our certificate-based approach greatly increases the difficulty of masquerading undetected for phishers, with single millisecond delays for users. We further show that this approach works not only against HTTPS-enabled phishing attacks, but also detects HTTP phishing attacks with port 443 enabled.**

***Index Terms*—certificates, machine learning, security.**

I. Introduction

Phishing remains a serious problem that threatens the security and privacy of online users despite attempts to thwart such attacks since 1995 [1]. Phishing attacks usually involve an attacker masquerading as a legitimate online entity to steal confidential information from the unsuspecting victims.

The efficacy of phishing depends upon the ability of the adversary to confuse the victim. Phishing websites often display content similar or identical to their legitimate counterparts and replicate the layout and 'look-and-feel' of these websites. These attacks often start with fraudulent emails sent to a group of online users to lure them to the fraudulent websites. In successful attacks victims click on the email hyperlinks which link to the phishing websites, and then victims disclose their private credentials (such as passwords or credit card information) or download malware.

Ideally enforcement of the requirement for an SSL/TLS connection in every important online transaction could defeat many of the current phishing attempts. Ideally, before a certificate can be issued the certificate authority (CA) needs to conduct a two-step verification, on domain ownership and legitimacy of the site owner. The web browsers or non-browser applications (e.g. online banking apps) must then verify the presented certificates and communicate the results to the end users in an appropriate form. Upon receiving the results (e.g. through the lock icon displayed next to the address bar of a browser), the end users should have sufficient information to make an informed decision on whether they should proceed to their online visits. Currently use of valid certificate by adversaries is infrequent; however, attackers are part of the long term trend towards more ubiquitous https. Both legitimate and attacking sites increasing use cryptographically valid certificates.

There are two components to defeating phishing. One is the detection of phishing sites. The second is the effective communication to end users that a site is indeed a phishing attack. Such effective communication would result in end users refusing to enter data in phishing sites or download the malicious payloads. We focus on the first component, that is the identification of phishing sites.

An obvious attack is typo-squatting. As anyone can register for virtually any unoccupied web domains, an attacker may also sign up for a misleading domain that is similar to the legitimate one he wishes to impersonate. For example, instead of bankofamerica.com, the attacker may get bank-of-american.com or bankofamerica.trustedbank.com. Without the verification of the legitimacy of the website owner, a technically valid certificate may be issued and presented to the end user. After the client-side verification, which primarily focuses on the validity of the certificate itself, a lock icon can even display in the browser window.

Public key certificates remain a potentially valuable source of information about the site owners. We argue that information embedded in the certificates is far more useful than the traditional client-side verification. In fact, due to the relatively dynamic nature of phishing domains, some underlying characteristics of their certificates can become a strong indicator of those websites. Through an analysis of the

978-1-4799-8909-6/15/$31.00

standard structure of the X.509 certificate and confirmed phishing websites, we have identified 42 machine-learning features and built classification models using a variety of algorithms. By ranking the influence of the identified machine-learning features, we discovered that the most effective features for our classifications were "relationship between web domain and the subject's common name listed in the certificate" and "length of the validity period". Yet other machine-learning features (e.g., "trusted CAs by major web browsers", "blacklist", "hash algorithm") further enhance the classification accuracy. We examined six machine-learning algorithms. Other than the five consistently important features listed here, the features had quite different ranks in models built on different algorithms. The results showed that when ten-fold cross validation was applied, the precision of our classification achieved 95% and the recall rate was above 93%.

Our primary research contribution is the proposal of a more effective mechanism to detect phishing websites. This certificate-based anti-phishing solution has several advantages over the existing mechanisms, in terms of correctness, completeness and timeliness. Applying the mechanism to our experimental dataset also demonstrated that our approach would also work well in phishing URLs using plain HTTP, as long as a certificate can be retrieved from the web server. Closely related to our work, Mishari et al. previously investigated the feasibility of using public key certificates to identify phishing websites from regular sites [2]. We differentiate our work from this (and other approaches for phishing identification) from the following perspectives: a larger dataset, a complete view of the X.509 certificate structure, higher classification performance in both categories, and more tested algorithms. We discuss them in detail in Section II-B.

This paper is organized as follows. Section II reviews previous research on anti-phishing techniques. Section III analyzes the threat model and describes our certificate-based approach. Section IV enumerates the set of features selected for use in our machine-learning models, and explains the basis for choosing that set. Section V overviews the related machine-learning algorithms and evaluation measurements for the performance of each model. Section VI reports the performance of the six trained models that resulted from our combination of feature and algorithm selection. Section VII discusses the limitations of the work. Specifically, we address miscategorizations and potential attacks. Section VIII summarizes and concludes with our plans for future work.

II. Related Work

There are two research areas with the greatest impact on this research. The first is the general battle against phishing. In terms of anti-phishing we discuss blacklisting and then move to more recent approaches. We characterize these according to correctness, timeliness, and completeness.

The second research area is the use of machine learning in security. As machine learning has been widely used in intrusion detection systems (IDS), malware identification, spam identification and other arenas, we focus specifically on the use of machine learning to detect phishing attacks.

A. Blacklisting

As an existing security solution to phishing, blacklists have been compiled by browser manufacturers, trusted third parties, and social networks of friends. Blacklists exist for different features of malicious sites: IP addresses (e.g. Spamhaus), domain names (e.g. PhishTank), and certificates (e.g. CRLs). An effective blacklist needs to satisfy three requirements simultaneously: correctness, completeness, and timeliness.

Correctness is the accuracy of the blacklist in distinguishing between malicious and non-malicious sites. Correctness directly affects the user experience. Previous empirical studies investigated the time required for updating phishing blacklists. In 2006, Ludl et al. tested 10,000 phishing URLs against the Microsoft and Google blacklists and received true positive rates of 65% for Google and 56% for Microsoft [3].

Timeliness is the delay between the publication of a phishing site and the time when the phishing site is included in the blacklist. The delay includes the identification of the site, the report of the site, the verification of the nature of the site, and then the client-side updates. This is especially difficult for spear-phishing websites. Since these websites normally target a very small portion of the online population, centralized blacklists may not receive a report after the attack has been successful. In 2009, Sheng et al. examined the exact time required for popular web browsers to blacklist a phishing URL [4]. Blacklists rarely identify fresh phishing pages within the first hour. After a site has been up for 12 hours, the identification rates could go up to 47%-83%, according to the study.

Completeness is the degree to which a blacklist accurately reflects the state of phishing online. It is a function of timeliness and correctness, as well as scope. Completeness is a challenge for blacklists due to the cycle of attack, identification, publication of the blacklist, then removal by defenders and reallocation by attackers of the hosting site. For example, in response to blacklisting phishing scams can frequently change their URLs, which leads to a short lifetime per phishing URL [5], [6]. The question then becomes how to create an effective defense mechanism against fresh and dynamic phishing sites before third-party blacklists are updated.

To meet all three requirements, we constructed a machine learning-based approach that can be executed by the end users in real time with high accuracy. Other user-centered and decentralized approaches have been proposed to identify phishing sites. NetTrust used temporal signals and social networks [7]. Moore examined crowdsourcing as a method to identify phishing sites [8].

Our proposed approach augments the existing blacklist approaches. First, the system described here does not require frequent updates from central servers. Due to the inherent lag between the first appearance of a phishing website and when the website is blacklisted, the users of traditional blacklists have a window of vulnerability to new phishing (especially

spear-phishing) attacks. In addition to being suitable for client as well as server deployment, our machine learning-based approach is able to identify websites that share a similar pattern of the known malicious websites but are not yet on the blacklist. Critically, machine-learning approaches can identify phishing sites as soon as they are encountered. Once identified these sites can be reported, thus augmenting blacklists.

B. Machine Learning and Phishing Detection

Machine learning has been increasingly utilized in security. Both unsupervised and supervised learning approaches have been explored for identifying malicious websites. Here we use supervised learning, i.e., there are pre-existing identified categories and some ground-truth knowledge (known as "training data"). Supervised learning models are built from this training data, which take advantage of the ground-truth knowledge. The data that are not used in training comprise a test set, which is then used to evaluate the performance of the trained models. After training and testing, these models can be used to classify new instances into the pre-defined categories. In terms of phishing detection, the machine-learning approaches attempt to distinguish between phishing websites and non-phishing websites based on the results of training using all machine-learning features.

Previous research has used both clustering and classification (i.e., unsupervised and supervised learning) to identify phishing, focusing on both phishing emails and the phishing websites. As an early proposal for email-based phishing detection, Basnet et al. tested features extracted from the format and contents of phishing emails. Their resulting machine-learning model was able to detect the majority of the 500 phishing email samples [9]. However, several identified features from 2006 now seem outdated. For example, the utilization of HTML objects and URL-based image source are now common in legitimate emails. Most importantly, the textual pattern and formats of phishing emails are easily manipulated by the adversary should this detection approach be widely deployed. Like phishing websites, the content of phishing emails is designed to be identical to legitimate emails, excluding the embedded URLs.

In contrast our approach is based on features extracted from the certificates of phishing websites. Previous similar approaches were primarily focused on the analysis of page content and classification of web URLs. Rosiello et al. constructed the HTML Document Object Model (DOM) trees for phishing and benign webpages, and used measurements such as layout similarity for the detection of phishing pages [10]. *CANTINA+* [11] is a multi-layer approach to extract features automatically from the DOM of a page. These features are then used to categorize the websites. Xiang et al. proposed a phishing-detection approach that extracts the claimed domain name from the page content, combined with a DNS lookup for the legitimate domain. A comparison of the two fields is then used to detect phishing [12]. A textual analysis of URLs using machine learning focused on the extraction of instances of obfuscation (e.g. www.bbc.unknown.com) and misspellings of the URL [13]. In an expansion of this work, Ma et al. combined the textual features of the URL (e.g. special characters, word appearance) and host information [14]. McGrath et al. examined the length of URL, the lifetime of a web domain, and other host information [15]. Based on Google's blacklist, Whittaker et al. built a phishing detection system using features extracted from the page URL, and page content [16].

These approaches utilized online resources that can be easily modified or forged by an adversary than cryptographic signatures or certificates. In contrast, the adversary would have to invest a time and money in order to obtain a valid end-entity certificate, particularly from a reputable CA. Given the low rate of return on phishing attacks, this may be economically significant. This is particularly applicable to fast flux attacks, for example with rock-phish sites cycling through up to eighty domains a day [17]. In addition, they would have to get these certificates significantly prior to a planned attack, which is another kind of cost increase. With an appropriate set of features, our proposed approach can be difficult to circumvent.

Ideally every time an end user provides personal identifiable information, the traffic would be encrypted and the host certificate would be validated. Unfortunately, this best-case scenario is not common practice. CAs may request proof of ownership of a web domain, but not perform additional verification on the legitimacy of the site owner's identity except for the Extended Verification (EV) certificates [18]. Site owners, especially smaller ones rarely obtain EV certificates, perhaps due to the higher costs and unproven benefit (e.g., from end users being unable to distinguish EV and non-EV certificates [19]). Pan et al. built machine-learning models based on features including page content, DNS records, etc. They also tested public key certificates. However the only feature utilized was the match between the subject field of the certificate and the web domain [20].

Closely related to our work, Mishari et al. previously investigated the feasibility of using public key certificates to identify phishing websites from regular sites [2]. Our approach differs in the following aspects.

First, in terms of the dataset used, we collected and tested certificates from all confirmed phishing websites associated with PhishTank entries, regardless of whether HTTPS was included in the listed URL. We found that connecting to TCP port 443 yielded many more certificates, even when the blacklisted phishing URLs were plain HTTP URLs. This approach substantially increased the size of our training and testing datasets, which consequently improved the value of the classification performance evaluation.

Second, by including a more complete view of the certificate structure in our analysis we achieved a significantly higher performance for both categories ('phishing' and 'non-phishing'). We have significantly expanded the examination and utilization of the feature set by including all standard X.509 certificate fields and optional certificate extensions. In addition to the certificate field values, relationships between certificate fields are utilized as features (e.g., between subject name and the

domain name; between initial valid date and date encountered).

Third, our constructed models achieved higher precision and recall for both the 'phishing' and 'non-phishing' categories. The importance of identifying non-phishing instance is equally important as that of phishing instances. A high false alarm rate will lead to low overall classification performance and therefore suboptimal user experience. We report the results in both categories in Section VI.

Fourth, we tested a larger number of classification algorithms. By analyzing the results from these algorithms, our proposed mechanism achieved better identification rates for both categories (again, phishing and non-phishing). We make recommendations on algorithms efficacy for different cases based on our experimental dataset.

The difficulty of obtaining a valid public key certificate that would comply with an optimized machine-learning model from a trustworthy CA is arguably increased as the constraints on the certificate are increased. For example, obtaining a domain name that passes a check grounded in the comparison between subject name and domain name is not trivial, particularly when the domain name is well-known. The more constrained a phishing attack must be to avoid detection, the greater difficulty and the greater the cost of an attack. Detection would be extremely difficult to circumvent with a distributed model using machine-learning mechanisms due to subtle differences in various institution instantiations, as the models can be continually updated.

III. System Design

In this section, we discuss the security threat of phishing. We then introduce the design of our certificate-based phishing detection mechanism.

The primary component which defines a phishing attack is an adversary who creates a malicious website which masquerades as a legitimate website. The goal is to trick victims into entering their private information on the phishing website, particularly authenticating credentials and financial information. The adversary can easily manipulate the content of a webpage, use misleading URLs that are similar to their legitimate counterparts, or even utilize HTTPS. Our proposed approach targets a subset of phishing attacks in which the adversary enables HTTPS connections or uses a hosting site with an active TCP port 443. We examined all identified phishing websites with an X.509 public key certificate, either self-signed, or obtained from a CA. Particular cases of interest include those where phishers have subverted a legitimate host, leveraging their control to host malicious content.

We make the following assumptions about the adversary.

1) The adversary has the ability to create a self-signed certificate for any arbitrary website.
2) The adversary may host the phishing content on a website with a valid server certificate issued by a CA. For example, the adversary may either have subverted a legitimate web server.
3) The adversary may purchase a valid server certificate (corresponding to the phishing domain name) from a CA.
4) The adversary does not have the ability to compromise a CA that is trusted by major web browsers.
5) The adversary has access to a potentially large number of certificates that are classified as 'non-phishing' by our system. That is, the multi-year collection of valid certificates could be repeated over time by adversaries.
6) The adversary has access to our proposed phishing detection system.

The design goals of the system are as follows.

1) Detect phishing websites with high precision and recall. This is the primary objective of all phishing detection techniques.
   We assume that the web domain and corresponding host server certificate are available to the machine-learning mechanism on the client. Following the classification, our mechanisms generates the predicted category, i.e., 'phishing' or 'non-phishing'.
2) Achieve superior classification performance in the non-phishing category. Effectively this requires low false positives with no false negatives. While categorizing non-phishing instances sometimes is sometimes overlooked, they are in fact of the same importance of the phishing category. A system making too many false classifications are not useful to support an informed decision by the end user. While such detection must be complemented by appropriate *risk communication* [21]. Without reliable detection such communications, nudges or warnings are not feasible. Too many warnings will result in people ignoring these profligate warnings. While warning design and usable security are not a focus of this paper, either requires an underlying detection system with few false alarms.
3) Prevent two adversarial learning attacks to circumvent the phishing detection system: evasion and poisoning. circumvention of the phishing detection mechanism. Defeating evasion requires a set of features that are, as a whole, difficult to forge or manipulate by the adversary. Prevent poisoning attack requires additional verification. In our case, a human analyst to filter out suspicious non-phishing instances from the training set. We discuss this in Section VII.

The structure and component interaction of our phishing-detection system is illustrated in Figure 1. The proposed system consists of the following four components.

Certificate downloader: The certificate downloader obtains a certificate from a server in two ways. First, through HTTPS the certificate may be provided by default to the browser during the connection. If no certificate is provided by default when connecting to the specific URL, the certificate downloader makes a separate call to TCP port 443 of the server. The downloaded certificate is stored in Base-64 format (PEM), with the corresponding domain name and the time of download creating a single record.

Feature extractor: The certificate downloader provides the

record (including the certificate and corresponding connection meta-data) to the feature extractor. The feature extractor parses the downloaded records into a set of actionable features. This requires transforming the data type of many of the certificate fields based on pre-processing rules, for example into boolean variables. The feature extractor also calculates the values for features which are a function of two or more fields in the record. The comparison of subject name and domain name is an obvious example. There are other internal evaluations, such as comparisons of dates. Some operations are simplifications; for example, the content of extensions is stripped and the features are number of extensions and existence of some classes of extensions. The values for all resulting features are stored in a vector (V) and are forwarded to the classification executor.

Creating a permutation of all possible fields and combinations is certainly possible, but not optimal. We selected meaningful features through examination of patterns of phishing, leveraging our understanding of the underlying problem using both personal observations and previous research. We used the resulting set of candidate features and then iterated on these sets as described in Section IV.

Classification executor: The feature extractor provides a vector of all features in the vector form required for the classification executor. The executor component comprises six previously trained machine-learning models. The algorithms used in these models are described in Section V-B. The specific models resulting from the application of these algorithms are described in Section VI. This component applies the six machine-learning models to the vector generated by the feature extractors. and the classification results of each model. Classification results are stored in another vector (R) to support final decision making. These are not purely Boolean data but can include probabilities and other details.

We evaluated a range of algorithms for the challenge of phishing detections. The classification executor utilizes multiple algorithms and instantiations. For example, Random Forest is constructed by the use of multiple decisions trees. The six algorithms are Random Forrest, K-Nearest Neighbors, C4.5, Decision Table, Naive Bayes Tree, and Simple Logistic. These are discussed in detail in Section V-B.

Decision maker: The decision maker calculates the single final probability after being provided with the detailed classification results. The decision making component is the final determination and thus recommendations of the website category. Depending on the personal or organization risk tolerances, end users may customize the policy to support their own decision-making. The decision maker used in our current instantiation was optimized for the highest true positive rate and the lowest false positive rate. Some organizations, for example, may choose to tolerate many false positives to avoid any risk of a false negative. The decision maker allows any organization to make this choice without requiring any understanding of the underlain machine learning mechanisms.

The decision maker component provides two options for the evaluation of the classification executor output: *Random Forest* and *Average Probability*. These two algorithms are complementary in that they provide different false negatives. Random Forest alone has good performance and it has a slight timing advantage over Average Probability while providing very good ROC curves. Average Probability results in fewer false negatives.

Thus the choice between these options is one between an increase in speed and a certainty in avoiding false positive. The choice would be determined both by risk tolerance of the user and other constraints (i.e., on a mobile phone Average Probability may be result in user-detectable delay). Thus the individual organization (or, less likely, user) can choose a different threshold for Random Forest , Average Probability, or a different manner of combining the outputs for their own risk posture. In our results we use majority likelihood, i.e., our decisions threshold is 0.5.

IV. Feature Selection

Effective machine learning requires identification of the salient features. More or less salient features can be identified iteratively through, for example, examining the varying weights created by different machine-learning algorithms. However, such an approach is insufficient and risks missing features that are important for structural reasons of both the dataset and the context of use.

If two features are highly correlated, such a process also risks selecting the dependent feature or variable. For example foot size is the greatest indicator is changes in ability to read; however the actual variable of interest is years of education. Changes in reading ability tend to plateau at adulthood, as does foot size. However the ability to predict changes is a function of continuing education, despite the fact that few obtain doctorates.

In contrast, our feature selection began with observations of previous work in phishing detection, selecting from the subject name and domain name match, and the date of initial validity. We also build on our previous work in domain reputation systems [7]. Self-signed certificates are obviously an indicator but not a determinant of a certificate being used for phishing. We identified further features, including contents of some individual fields, existence (or not) of other fields, characteristics of fields (e.g., length), and the relationship between two or more of these. The end result was a set of 42 features based on the structure and content of the X.509 certificate and the data associated with the context of its observation (e.g., date, domain name). Some fields have been treated as indicators but were not included here both due to their widespread use in legitimate certificates and timelines of verification for the user. For example, trust chain verification would require validation of multiple certificates for each individual certificate classified, including connection time and signature validation. Here we enumerate the features selected for in the final classifiers.

A. Temporal Features

Features: *Diff(NotBefore, NotAfter), Diff(NotBefore, DateDownloaded), Diff(DateDownloaded, NotAfter).*

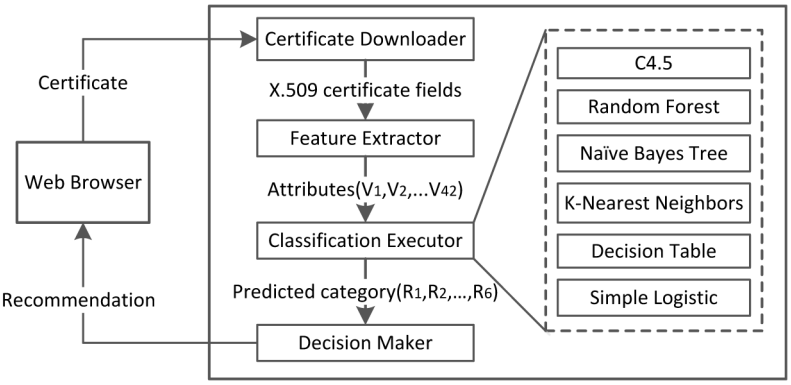

Fig. 1. Certificate-based Phishing Detection System

The expiration date of a certificate is an important security indicator; however, we also calculated the number of days between certificate expiration and the date we downloaded the certificate. In addition we examined the entire validity period by a simple subtraction of *NotBefore* from *NotAfter*. We found these differences to be important features in some cases. While it is crucial to possess a valid certificate, it is not a secure practice for a CA to issue a server certificate that is valid for an extremely long period of time (e.g. a decade or more). Further, certificates that are newly issued are more suspicious than a certificate that has been used for a relatively long period (e.g. one year). We therefore identified the number of days since a given certificate was issued as a machine-learning feature. A large time between initial issuance and observance was also suspicious.

B. Issuer, Subject and Domain Name

Issuer and Subject are two special fields of the X.509 certificate. Instead of assigning one string value to each field, the CA can provide a richer set of information using 'Distinguished Names' (DNs). Detailed information such as country, state, city, email, common name, organization name, and organizational unit can be included as sub-fields, known as 'Relative Distinguished Names' (RDNs). We developed three groups of machine-learning features based on the sub-fields of Issuer and Subject.

1) Relationship between Sub-fields: Common Name (*CN*), Organization (*O*), and Organizational Unit (*OU*) are three sub-fields that are closely related to the identity of the issuer or subject. We examined the relationships between the following three data tuples to create five additional features.

a) *Match(Issuer_CN, Issuer_O), Match(Issuer_O, Issuer_OU).*
These two features can reveal the structure and naming conventions of the CA. Patterns extracted from these two features are useful for detecting sudden but cryptographically valid changes in the issuer field. They could indicate different CA policies and may even be used to detect rogue certificates that are technically valid. Insecure certificates have become a particular focus of large online entities such as Microsoft [22].

b) *Match(domain, Subject_CN).*
Following the IETF RFC 6125 [23] and based on current common practice, we determine whether the subject's common name matches the present domain name. Note that we consider this an indicator and not a determinant of phishing, as the domain name may also be listed in the Subject Alternative Name extension.

c) *Match(Issuer_CN, Subject_CN), Match(Issuer_CN, domain).*
These two attributes are designed primarily for detection of self-signed certificates. They are not used as deterministic flags for phishing websites. Larger companies such as Google may operate their own CAs and issue certificates to their sites. Similarly, universities and other public institutions may run CAs and sign their own certificates.

2) White and Blacklists for Issuer and Subject.:

a) *isTrustedCA (Issuer), isHighlyTrustedCA(Issuer).*
These two features are indicators of the CAs' reputations. Since there is no limitation on the jurisdiction of a CA, it is technically feasible for a rogue CA to issue a valid certificate for any website. We therefore created a "Trusted CA list", consisting of a subset of the certificate authorities that are trusted by major web browser manufacturers (Microsoft, Apple, and Mozilla). Google Chrome relies on the underlying operating system to maintain root certificates, so it is also covered by this list [24]. We maintain a second trust list that only contains VeriSign and Thawte. VeriSign is a highly reputable CA that has issued certificates for many large organizations. Thawte, which is also owned by Symantec, provides low-cost solutions to smaller websites. This list provides another candidate feature for reliable CAs. Again, the CA itself is not the sole indicator. Suspicious certificates may come from reputable CAs For example, Flame illustrates a cryptographic attack that can be used against certificates from highly trusted CAs (particularly those with weak algorithms, as described in Section IV-C).

b) *isProhibited (Issuer), isProhibited (Subject).*

Due to the transient nature of phishing sites, a complete blacklist cannot be created a priori. Yet, certain values should never appear on the issuer and subject fields, for example, local network addresses: 192.168.x.x, 10.x.x.x, 127.0.0.1, or localhost. These prohibited entries alone do not prove that a certificate is used for phishing, however they can serve as an important indicator. As another widely observed example, default self-signed certificates in website hosting software by Parallels and VMWare are not secure in practice. Further, although wildcard certificates are technically permitted, having only the star character ('*') in the issuer or subject creates a certificate that can be easily exploited by an adversary.

3) Other Sub-fields: Email, Organization, Organizational Unit, City, State, Country of Issuer and Subject, respectively.

Although we do not try to match other optional sub-fields of the issuer or subject, the length of each field (besides country) is examined. The information is used by different machine-learning algorithms to extract the underlying pattern of a CA. For example, VeriSign and GeoTrust provide only the country of the issuer, not state or city. Several self-signed certificates, however, fill in every optional field. Similarly, some optional sub-fields may be omitted by certain CAs. Finally, for the country of issuer or subject, we recorded whether it is located in the US. Country codes can serve as important additional information for the classification, as the rates of malicious websites in different countries vary significantly [25], [26].

C. Other Required Fields

Features: *Cryptographic algorithms, CertVersion, len(CertSerial).*

The first feature in this group consists of the cryptographic algorithm and hash function that are used for generating the signature of the certificate. We consider this feature as an important indicator for insecure certificates, as MD5 is still among the candidate hash functions. This may also reveal conventions for certificates issued by a specific CA. Certificate version indicates the structure of the certificate and implies level of cryptographic strength, since later certificate versions have the choice of stronger algorithms. Extensions also vary across versions.

We also use the length of the certificate serial number as a candidate indicator of the pattern for each CA. While the actual serial numbers are assigned randomly the length of the number rarely changes.

D. Certificate Extensions

Features: *Extension count, existence of common extensions and whether they are critical, other extension count.*

The certificate extensions can serve as an important indicator of a fraudulent certificate (e.g. the Flame malware certificate) or an insecure certificate (e.g. self-signed certificates by Plesk). For the optional certificate extensions, we first count the total number of extensions. We then check on whether each of the 12 most frequently used extensions is defined in the certificate. Although none of these extensions is required for an X.509 certificate, important information may be embedded by the CAs. This information includes the identity of the issuer and subject, constraints and policies on the usage of public keys, and Certificate Revocation List (CRL). There is also a feature that indicates whether the certificate is an Extended Validation (EV) certificate, implying a greater investment in certificate issuance. In addition to the existence of an extension, we also examine whether each of the extensions has been labeled as 'critical' by the issuer.

Here is a list of common extensions: *authorityKeyIdentifier, subjectKeyIdentifier, keyUsage, certificatePolicies, subjectAltName, issuerAltName, subjectDirectoryAttributes, basicConstraints, isExtendedValidation, extendedKeyUsage, crlDistributionPoints, freshestCRL, authorityInfoAccess*.

Additionally, we conduct another count on the number of extensions not in the list above.

V. Experimental Design

A. Data Collection

Our data collection began Dec. 2012 and continues. It uses PlanetLab [27] and initiates connections with servers from three continents. We also have servers in Eastern and Pacific time zones in the United States. Our data collection of non-phishing certificates began in December 2012. Our script has been downloading a list of the top 1 million websites daily from Alexa [28] since that time. On occasion for institutional reason collection was interrupted. Thus there are a few days with no data.

We connected to each targeted website via TCP port 443. A certificate was downloaded when an HTTPS connection request was successfully established, and when the certificate was different from the previous observation. For the purpose of choosing a representative 'non-phishing' category, we choose the certificates that were downloaded from the top 100,000 websites for classifier training and testing, as described in Section VI.

We also maintain a constantly updated database of confirmed phishing and phishing-related certificates. Our script checks with PhishTank [29] every six hours for the latest active phishing list, as phishing pages usually have short lifetimes. While phishing is normally linked with web pages rather than domains, we discovered that an increasing number of phishing pages reside in websites with HTTPS enabled. In addition to the HTTPS URLs that were listed, we attempted to connect to all websites associated with the PhishTank list including those with plain HTTP URLs listed. We then initiated a TCP connection to port 443, and downloaded any response that resulted in a certificate. This data collection began in Oct. 2013.

Once a certificate is downloaded from a website, it is parsed locally into a set of values according to the standard X.509 certificate fields. The 42 features are then calculated from the values of the required and optional fields. The instances are then saved in an ARFF format data file for further analysis. ARFF is the data format of Weka [30]. During the data collection, we have recorded 95,490 instances in the 'phishing'

category with an extremely large non-phishing category — 1.1 million unique certificates. To avoid possible negative effect of an imbalanced dataset, we have taken an *under-sampling* approach [31] to randomly select instances from the 'non-phishing' category. We ended up with 113,156 non-phishing instances used in the classification from the top 100,000 websites. These websites include the most frequently targeted (i.e., Paypal, Bank of America) and also follows common research practices for this area.

B. Machine-Learning Algorithms

In this subsection, we provide a brief overview of the machine-learning algorithms we used for classification.

To the best of our knowledge, there is no classification algorithm that performs universally better than other algorithms for all datasets. We therefore tested several well-known classification algorithms. We did this using the machine-learning tool Weka [30]. We ranked the algorithms based on their overall performance. In this section, we describe these algorithms with the best performance.

C4.5 (Weka implementation: J48) is a basic decision-tree algorithm. Before adding a node (feature) to the decision tree, the algorithm calculates the *information gain* for all features according to its entropy. It then creates a single decision based on the feature with the highest value. Then the feature selection is iterated on the remaining features. It divides on the feature with the highest value. One advantage of C4.5 over its predecessor ID3 is the reduction of *overfitting*. The algorithm goes through the decision tree once it is constructed and replaces unnecessary branches of the decision tree with leaf nodes. This process is known as *pruning*.

Different from C4.5, Random Forest adds randomness to the generation of decision trees. Instead of relying on one single decision tree to cover the entire dataset and features, this approach selects features and training data randomly from the given sets and constructs a series of decision trees based on these randomly selected inputs. The output of Random Forest is then calculated by the outputs of the contained decision trees.

Naive Bayes Tree is a hybrid algorithm that combines the strengths of decision trees and Naive Bayes classification. The algorithm builds a tree-like structure with a Naive Bayes classifier on every leaf node. Based on the comparison of the current node's *utility* versus the utility for splitting the current node, the algorithm can decide whether a leaf node (Naive Bayes classifier) or an intermediate node (decision-tree node) is needed for the model. With Naive Bayes the path through the decision tree does not determine but rather influences the classification. The final classification will be a result of the Bayesian function at the leaf.

Simple Logistic Regression is a regression analysis. It represents the underlying connections between the set of features and the categories using one logistic function per category.

Decision Table is another useful algorithm that starts with a search on the subset of features to find those with the best performance. Based on the selection of features, a number of rules are developed for the classification of the new instances.

K-Nearest Neighbors (Weka implementation: iBK) is an instance-based algorithm. For each unknown instance, its category is determined by a majority vote of the K training instances that are closest to that instance (based on the features). This algorithm is simpler than the other algorithms in terms of model building but requires loading the entire dataset when making each prediction. Based on the *Cross Validation Parameter Selection*, our model calculates three neighboring instances to generate the predicted class (K=3).

C. Evaluation Methods

In order to evaluate the average performance of our classifications, we applied *10-fold cross validation* for each algorithm in our experiment. Specifically, this approach partitions the entire dataset into 10 subsets of equal size. A subset is randomly chosen for validation, while the remaining nine subsets are used to build the machine-learning model. The process then repeats nine more times in which each subset is used for validation once.

Precision and recall are two important metrics that are commonly used to evaluate the classification performance of a particular category. Precision is the percentage of instances classified into a category that have been correctly classified. That is, if there are a large number of classification into a given category but many of them should not have been that category, then that model has low precision for the measured category. The formula to calculate precision is shown in Equation 1.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

In contrast recall measures how many instances have been missed, rather than how many have been incorrectly added, for a given category. The formula to calculate recall is shown in Equation 2.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The overall performance of the trained classifiers is evaluated by *Kappa Statistic*. The measure gives a numeric value between 0 (equivalent to a random classification) and 1 (perfect classification). The formula to calculate Kappa Statistic is listed in Equation 3.

$$Kappa = \frac{Prob(Classifier) - Prob(Random)}{1 - Prob(Random)} \tag{3}$$

For the 'phishing' category, we have also examined the *Receiver Operating Characteristic (ROC)* curve. The ROC curve can provide a direct illustration of how the classifier performs. The curve is plotted with a series of 'thresholds', indicating different combinations of True Positive Rate (TPR) and False Positive Rate (FPR). The TPR and FPR are calculated as shown in Equations 4 and 5, respectively.

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

$$FPR = \frac{FP}{FP + TN} \quad (5)$$

The two metrics are ranked and plotted in the form of (TPR, FPR) on the coordinate system, respectively. An ideal classifier can achieve the highest TPR with a small FPR, indicating high correct classification rates with low false alarm rates. From the chart, an ideal ROC curve should be close to the upper-left corner of the coordinate system. We have plotted the ROC curve of our best performing classifier in Figure 2, while performances of all classifiers are provided using the Area Under Curve (AUC). This (numeric) metric also gives a direct view of the performance similar to the actual ROC curve. The maximum value of AUC is 1, indicating an ROC curve that is close to the upper-left corner and therefore covers the entire 1x1 coordinate system.

VI. Results

In this section we provide more details on the classification performance of our trained models. As stated in the previous section, we utilized 10-fold cross validation to obtain an averaged measurement of the performance. Six algorithms were utilized to train the classification models: C4.5, Random Forest, K-Nearest Neighbors, Naive Bayes Tree, Decision Table, and Simple Logistic regression. We also examined *Average Probability*, which is an *ensemble* learning approach that simply calculates the classification based on the average probabilities of the six models.

A. Classification Accuracy

Table I demonstrates the classification performance, both overall and by category. As the primary goal of this paper is phishing detection, we ranked the algorithms based on the precision of the 'phishing' category. According to the table, Random Forest, K-Nearest Neighbors (iBK), and C4.5 (J48) all achieved great precision and recall rates, above 92.8%. The classification model built using Random Forest even achieved 95.5% precision with 93.7% recall for the phishing category, demonstrating a high accuracy not only among the websites that were classified as 'phishing', but also among websites that were actually phishing. When considering multiple algorithms, Average Probability reached a recall of 94.1% in the phishing category, with a precision of 93.5%. The precision and recall of the phishing category are comparable to the performance a large number of phishing detection proposals [10], [11], [15], [16].

An important strength of our proposed approach is that our classification models do not sacrifice precision or recall for the 'non-phishing' category for 'phishing'. The rates in the 'non-phishing' instances are of the same importance as the performance of the phishing category. The false positive and negative rates of the 'non-phishing' category would affect the false alarm rate of the phishing detection system, which directly links to user experience of the system. False positives create warning fatigue which in turn results in users ignoring alerts. Warning fatigue is a relevant (and well established) phenomena across the field of information security. Even those users aware of privacy and security risks generally click through EULAs and privacy policies to accomplish their desired tasks. [32] An in depth look at users' ability and SSL warnings found that in most cases, expert users were not different than non-experts in that most users were willing to ignore SSL warnings to get to a desired website. [33] Users may be experiencing warning fatigue and do not want to be bothered to read pop-up notifications if the notifications interrupt their desired task, such as app installation. [34] Even more explicit or graphical warnings have mixed results. [35] Warning habituation is also a problem that results in users ignoring alerts [36]. For Random Forest, the precision achieved 94.7%, and the recall rate was 96.3%. For Average Probability, the precision of the 'non-phishing' category was 95.1% with a recall of 94.6%. High precision and recall rates in both categories is also a direct indicator of the high quality of feature selection.

Classifications using Naive Bayes Tree, Decision Table, and Simple Logistic also yielded good precision and recall rates. NBTree demonstrated stable performance in terms of precision and recall for both categories, which were comparable to the other tree-based algorithms (C4.5 and Random Forest). Decision Table performed better in the detection of phishing, with 92.6% of precision and 85% of recall. Although the performance of Simple Logistic was not as good as other approaches, we still listed this as a candidate algorithm, since this algorithm generates models that are smaller and easy to understand. In addition, since miscategorized certificates varied across different algorithms, we observed that there were several certificates miscategorized by other algorithms but were correctly categorized by Simple Logistic. In Section VII we will discuss the miscategorizations of our top two performing approaches: Random Forest and Average Probability.

Finally, the Kappa Statistic column shows that the rates of correct classifications were high for every algorithm. Random Forest and K-Nearest Neighbors both achieved 0.9, which is an excellent classification result. Additionally, as described in the previous section, the area under the ROC curve is an important indicator of the classification performance. The AUC values for the first five algorithms were all above 0.94, while Simple Logistic also achieved 86%. The AUC of Average Probability has reached 0.99. As an example, we plotted the ROC curve of the Average Probability algorithm, shown in Figure 2. Close to the ideal case, the curve has a very short distance to the upper left corner of the coordinate system.

B. Effectiveness

As it is an important aspect of the performance evaluation, we tested the effectiveness of the system by recording the time for classification. This consists of the time required to parse a certificate string into machine-learning features, and then to conduct classification based on the preferred algorithm as well as Random Forest, and finally to make a decision based on the six algorithms. Should the end user choose to be warned based on Average Probability, the decision-making process will need

TABLE I
CLASSIFICATION PERFORMANCE

Algorithm	Phishing		Non-Phishing		Kappa	AUC
	Precision	Recall	Precision	Recall	Statistic	
Random Forest	95.5	93.7	94.7	96.3	0.9	0.98
K-Nearest Neighbors	95.3	93.6	94.7	96.1	0.9	0.97
C4.5	93	93.2	93	94.3	0.87	0.97
Decision Table	92.6	85	88.2	94.2	0.8	0.94
Naive Bayes Tree	90.9	90.5	92	92.4	0.83	0.96
Simple Logistic	73.7	87.3	87.3	73.8	0.6	0.86
Average Probability	93.6	94.2	95.1	94.6	0.89	**0.99**

TABLE II
EFFECTIVENESS EVALUATION OF CERTIFICATE CLASSIFICATION

Component	Average Time Per Certificate (ms)
Feature Extractor	2.28
Random Forest (first)	1614
Random Forest (remaining)	0.006
Average Probability	2721

to wait until the slowest algorithm, K-Nearest Neighbors, finishes.

The classification executor and decision maker were written in JavaScript to ensure better integration with the web browser. We implemented the feature extractor in Python, as it could parse the certificate strings more efficiently than JavaScript. We then partitioned a total of 208,600 certificates into 10 groups of equal size, and recorded average time for each component.

One interesting observation in our analysis of classification executor is the initialization time of this component was significantly longer than subsequent classifications in tree-based algorithms (e.g. Random Forest). Therefore, it would take a similar amount of time for classifying only one certificate compared to a group of 20,860 certificates. We attribute the improvement on the average classification time to the automatic optimization by the Just-In-Time (JIT) compiler on the 'if-else' statements in JavaScript. This phenomenon, however, did not occur in other algorithms (e.g. K-Nearest Neighbors) with fewer conditional statements. In Table II we summarized the time to classify the first certificate and the average time for classifying each of the remaining certificates in the group.

As shown in Table II, feature extraction can be completed very quickly, within 3 milliseconds. Average Probability required three seconds to complete, caused by the calculation of the distances between nearest neighbors for each certificate instance in the slowest algorithm. Our implementation of Random Forest required an initialization time of 1.6 seconds, while the subsequent certificates would take only 2.2 milliseconds per certificate including the feature extractor. (See Section VI-B for more details.) for details) According to the analysis of correctness and effectiveness, the end user may choose from Random Forest, which runs more quickly and with a superior precision, or Average Probability that runs relatively slower but may be more accurate in phishing detection.

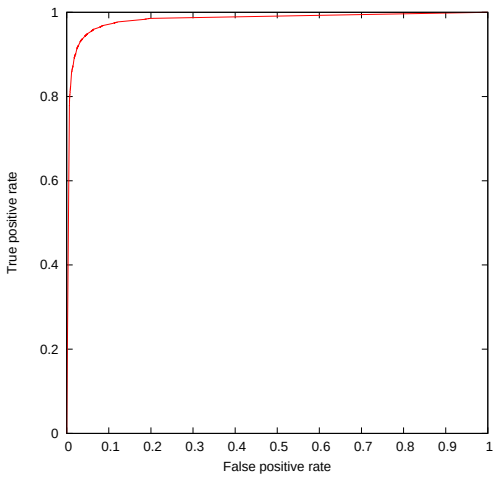

Fig. 2. ROC Curve for the Average Probability Classifier

VII. DISCUSSION

In this section we further analyze the results of classification, potential attacks against our proposed certificate-based classification, and offer some additional observations from our data analysis.

We demonstrate the classification results of Random Forest and Average Probability using a waterfall chart in Figure 3. The horizontal lines in the chart indicate the categories. The first row in this chart is the actual category of the certificates, and the lower lines indicate the classification results of our best performed classification algorithms, respectively. The chart uses different colors to denote the actual category of the instances so classification result is visible in each of the classification categories (non-phishing=green, phishing=dark blue, but is readable in black and white). A vertical line indicates an agreement in classification between two algorithms (i.e. the instances are classified as the same category); while an oblique line (from the left half to the right half, and vice versa) shows a disagreement between two algorithms.

We can observe that Random Forest and Average Probability are complementary in terms of misclassifications. Overall, the True Positive Rate of Average Probability in the phishing category is slightly higher than that of Random Forest (94.2% vs. 93.7%). However, Random Forest has a lower False Positive Rate (3.7% vs. 5.5%).

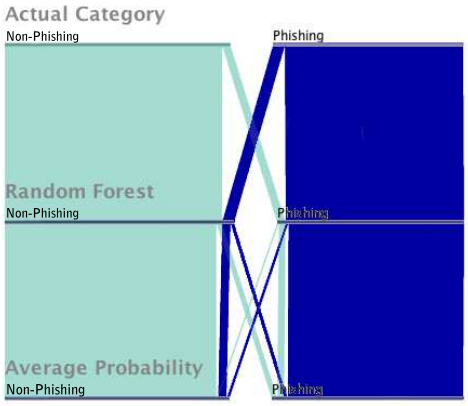

Fig. 3. Classification Results based on Random Forest and Average Probability

A. Limitation

A simple attack to circumvent our certificate-based phishing-detection system is disabling all HTTPS connections from the web server. However, rejecting all inbound HTTPS connections can be a significant indicator when sensitive information is supposed to be entered on the website. This case can be detected by an analysis of page DOMs, especially the password fields in HTML.

There are more complex and typical adversarial learning attacks: evasion and poisoning.

In evasion, the adversary attempt to modify the values of several certificate fields in order to obtain a certificate that looks benign by our classifiers. To increase the success rate, he can obtain a valid certificate from a highly reputable CA that is trusted mutually by the major web browsers, or even purchase an extended validation certificate for his website. This method involves substantial financial investment and can fail if the reputable CA performs regular checks on the page content and/or domain matches.

Such an increase in cost would possibly defeat fast-flux attacks, as these would no longer be financially viable. Instead of relying on a reputable CA, the adversary may create his own self-signed certificate, which can be captured by multiple machine learning features (e.g. Match(subject.CN,issuer.CN)).

Further, features such as 'hash algorithm' are potential indicators for hash-collision attacks, which rely on MD5 (although forged certificates are not the focus of this paper).

Lastly, we expect our set of features to evolve over time, based on new threats. Our certificate-based approach can also be combined with other indicators to achieve optimal detection.

Poisoning is the second typical type of adversarial learning attack. The attacker can intentionally make some phishing certificates and applies them to the top-visited sites, but without performing the attack right away on these websites. This requires long-term commitment with an uncertainty of not being detected during the poisoning period. Popular websites are likely to identify these certificates (e.g. Google).

Generally, our results indicate that the more popular a website is, the easier it is to distinguish from a phishing website. Notice that the economics of phishing require very large-scale spam campaigns, so that less popular websites are less profitable to phish due to size of the target population.

One interesting observation we made was that, besides pages on the same website, certificates may be reused across multiple domains. To make the performance evaluation of our approach comparable to other local anti-phishing approaches discussed in related work, we allowed duplicate entries of certificate strings in our dataset if they belonged to the same category. We have reported separately on the current status of banking certificate in another paper [37].

VIII. Conclusion

We have shown that automated analysis of TLS certificates using machine learning is a promising approach for identifying phishing websites. We conclude that it is feasible to classify websites into 'phishing' and 'non-phishing' categories with reasonable accuracy using the set of classifiers we have developed. Public key certificate classification has an advantage in that it is less dependent on a central server than blacklisting.

This approach changes the economics of phishing. Sensitive information should only be entered if a connection is protected by SSL/TLS. The presence or absence of such protection can easily be detected. Should phishing be limited to subverted but legitimate websites, this changes the economics of phishing. Legitimate websites have the incentive and the competence to recover quickly. (Previous work has shown that recovery time is the most critical variable in preventing epidemics of subversions [38].) Fastflux would be more difficult. It is exactly this kind of attack that is most effective against blacklists. Thus this approach is complementary to blacklisting and other techniques, such as those examining temporal signals or performing content-based analysis of websites.

Our findings motivate several avenues for further investigation. While this TLS-based identification is more resilient to manipulation than URLs and page content, the features we have selected are subject to different degrees of control by attackers. We would like to investigate how phishing attacks would evolve to such defenses. First, we are currently studying the rate of change in values for various certificate fields. We see this as a potential feature for phishing detection, as policies of renewing certificates can vary significantly among websites. Well-known websites usually present a more stable pattern of certificate renewal compared to phishing websites. We would also like to expand the classifier to include the top million websites.

Additional testing of the classifiers against different certificate compilations is an important next step. We are seeking to identify an international partner with a larger certificate compilation, one that focuses on markets with dominant languages other than English. We are also seeking a multinational partner for real-time classification into the categories of legitimate work-related certificates and all other certificates. This approach has the potential to prevent entering workplace

credentials into any non-employer website, thus preventing spear-phishing. These experiment could provide verification of the efficacy of our certificate-based phishing detection approach in the business world. For this reason, our classifiers will be public upon publication of these initial results.

ACKNOWLEDGMENTS

Research was sponsored by the Army Research Laboratory and was accomplished in part under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). This material is based upon work supported, in part, by the National Science Foundation under Grant CNS 1250367; DHS BAA 11-02-TTA 03-0107, and Google. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, Department of Homeland Security, Google, NSF, Indiana University or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] M. Langberg, "AOL acts to thwart hackers," *San Jose Mercury News*, Sep. 1995.

[2] M. A. Mishari, E. De Cristofaro, K. E. Defrawy, and G. Tsudik, "Harvesting SSL certificate data to identify web-fraud," *arXiv preprint arXiv:0909.3688*, 2009.

[3] C. Ludl, S. Mcallister, E. Kirda, and C. Kruegel, "On the effectiveness of techniques to detect phishing sites," in *Proc. of DIMVA '07*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 20–39.

[4] S. Sheng, L. F. Cranor, J. Hong, B. Wardman, G. Warner, and C. Zhang, "An Empirical Analysis of Phishing Blacklists," in *Proc. of CEAS '09*, Jul. 2009.

[5] T. Moore and R. Clayton, "Examining the impact of website take-down on phishing," in *Proc. of eCrime '07*, 2007, pp. 1–13.

[6] R. McMillan, "'rock phish' blamed for surge in phishing," *InfoWorld*, vol. 12, 2006.

[7] L. J. Camp, "Reliable, usable signaling to defeat masquerade attacks. the," in *Proc. of WEIS '06*, 2006, pp. 26–28.

[8] T. Moore and R. Clayton, "Financial cryptography and data security," G. Tsudik, Ed. Berlin, Heidelberg: Springer-Verlag, 2008, ch. Evaluating the Wisdom of Crowds in Assessing Phishing Websites, pp. 16–30. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85230-8_2

[9] R. Basnet, S. Mukkamala, and A. Sung, "Detection of phishing attacks: A machine learning approach," in *Soft Computing Applications in Industry*, ser. Studies in Fuzziness and Soft Computing, B. Prasad, Ed. Springer Berlin Heidelberg, 2008, vol. 226, pp. 373–383. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77465-5_19

[10] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi, "A layout-similarity-based approach for detecting phishing pages," in *Proc. of SecureComm '07*, Sept 2007, pp. 454–463.

[11] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, pp. 21:1–21:28, Sep. 2011.

[12] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proc. of WWW '09*, 2009, pp. 571–580.

[13] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proc. of WORM '07*. ACM, 2007, pp. 1–8.

[14] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URL's: An application of large-scale online learning," in *Proc. of ICML '09*, 2009, pp. 681–688.

[15] D. K. McGrath and M. Gupta, "Behind phishing: An examination of phisher modi operandi," in *Proc. of LEET '08*, 2008, pp. 4:1–4:8.

[16] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in *Proc. of NDSS '10*, 2010. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/10/pdf/08.pdf

[17] T. Moore and R. Clayton, "Evil searching: Compromise and recompromise of Internet hosts for phishing," in *Financial Cryptography and Data Security*. Springer, 2009, pp. 256–272.

[18] C. B. Forum, "About EV SSL," https://cabforum.org/about-ev-{SSL}/, Last Checked: 12/14/2014.

[19] R. Biddle, P. C. van Oorschot, A. S. Patrick, J. Sobey, and T. Whalen, "Browser interfaces and extended validation SSL certificates: An empirical study," in *Proc. of CCSW '09*. ACM, 2009, pp. 19–30.

[20] Y. Pan and X. Ding, "Anomaly based web phishing page detection," in *Proc of ACSAC '06*, 2006, pp. 381–392.

[21] J. Blythe, J. Camp, and V. Garg, "Targeted risk communication for computer security," in *Proc. of IUI '11*. ACM, 2011, pp. 295–298.

[22] S. Anoosh. A novel method in IE11 for dealing with fraudulent digital certificates. [Online]. Available: http://blogs.technet.com/b/{PKI}/archive/2014/02/22/a-novel-method-in-ie11-for-dealing-with-fraudulent-digital-certificates.aspx

[23] IETF, "Representation and verification of domain-based application service identity within Internet Public Key Infrastructure using X.509 (PKIX) certificates in the context of transport layer security (TLS)," http://tools.ietf.org/html/rfc6125, 2011.

[24] T. C. Projects, "Root certificate policy," http://www.chromium.org/Home/chromium-security/root-ca-policy, 2013.

[25] M. van Eeten, J. Bauer, H. Asghari, S. Tabatabaie, and D. Rand, "The Role of Internet Service Providers in Botnet Mitigation An Empirical Analysis Based on Spam Data," *Proc. of WEIS '10*, 2010.

[26] V. Garg, L. J. Camp, and C. Kanich, "Analysis of ecrime in crowd-sourced labor markets: Mechanical turk vs. freelancer," in *The Economics of Information Security and Privacy*. Springer, 2013, pp. 301–321.

[27] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

[28] Alexa, "Alexa top sites," http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[29] PhishTank, "Developer information," http://www.phishtank.com/developer_info.php.

[30] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[31] C. Drummond, R. C. Holte *et al.*, "C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling," in *Workshop on Learning from Imbalanced Datasets II*, vol. 11. Citeseer, 2003.

[32] R. Böhme and S. Köpsell, "Trained to accept?" in *Proc. of CHI'10*. New York, New York, USA: ACM Press, 2010, p. 2403.

[33] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, "Crying Wolf: An empirical study of SSL warning effectiveness," in *Proc. of USENIX Security '09*, 2009.

[34] A. Felt, S. Egelman, M. Finifter, D. Akhawe, and D. Wagner, "How to ask for permission," in *Proc. of HotSec '12*. USENIX Association, 2012.

[35] K. Benton, L. J. Camp, and V. Garg, "Studying the effectiveness of android application permissions requests," in *2013 IEEE Int. Conf. Pervasive Comput. Commun. Work. (PERCOM Work.*, no. March. IEEE, Mar. 2013, pp. 291–296.

[36] B. B. Anderson, B. Kirwan, J. Jenkins, D. Eargle, S. Howard, and A. Vance, "How Polymorphic Warnings Reduce Habituation in the Brain: Insights from an fMRI Study," in *Proc. of CHI'15*. ACM, 2013.

[37] Z. Dong, K. Kane, and L. J. Camp, "Phishing in smooth waters: The state of banking certificates in the us," in *2014 TPRC Conference Paper*, 2014.

[38] T. Kelley and L. J. Camp, "Online promiscuity: Prophylactic patching and the spread of computer transmitted infections," in *The Economics of Information Security and Privacy*. Springer, 2013, pp. 157–180.