EFFECTIVE PHISHING DETECTION USING MACHINE LEARNING APPROACH

by

YAOKAI YANG

Submitted in partial fulfillment of the requirements

For the degree of Master of Science

Department of Electrical Engineering and Computer Science

CASE WESTERN RESERVE UNIVERSITY

January, 2019

Effective Phishing Detection using Machine Learning Approach

Case Western Reserve University

Case School of Graduate Studies

We hereby approve the thesis[1] of

YAOKAI YANG

for the degree of

Master of Science

Michael Rabinovich

Committee Chair, Adviser 11/16/2018
Department of Electrical Engineering and Computer Science

Soumya Ray

Committee Member 11/16/2018
Department of Electrical Engineering and Computer Science

Andy Podgurski

Committee Member 11/16/2018
Department of Electrical Engineering and Computer Science

[1]We certify that written approval has been obtained for any proprietary material contained therein.

Table of Contents

List of Tables

List of Figures

Acknowledgements

0.1 Acknowledgements

Abstract

Effective Phishing Detection using Machine Learning Approach

Abstract

by

YAOKAI YANG

0.2 Abstract

Online phishing is one of the most epidemic crime schemes of the modern Internet. A common countermeasure involves checking URLs against blacklists of known phishing websites, which are traditionally compiled based on manual verification, and is inefficient. Thus, as the Internet scale grows, automatic URL detection is increasingly important to provide timely protection to end users. In this thesis, we propose an effective and flexible malicious URL detection system with a rich set of features reflecting diverse characteristics of phishing webpages and their hosting platforms, including features that are hard to forge by a miscreant. Using Random Forests algorithm, our system enjoys the benefit of both high detection power and low error rates. Based on our knowledge, this is the first study to conduct such a large-scale websites/URLs scanning and classification experiments taking advantage of distributed vantage points for feature collection. Experiment results demonstrate that our system can be utilized for automatic construction of blacklists by a blacklist provider.

1 Introduction

1.1 Current Security: TLS, X.509 PKI and HTTPS

X.509 public key infrastructure[1] (referred to as X.509 PKI later) is a top-down tree-like structure, in which the root node is root certificate authorities (referred to as CA later) trusted by both clients and servers. Intermediate nodes of the tree are intermediate CAs, and the leaf nodes are certificates vouching for the identity of given websites. Based on this structure, X.509 PKI offers a chain of trust, from leaf to root, which is used by modern browsers to verify the identity of websites and their public key.

Transport layer security (TLS) protocol[2], designed to ensure data confidentiality, data integrity and end-to-end authentication for communication between two hosts over untrusted links, operates below the application layer and works as the cornerstone of a significant number of prevalent application layer protocols such as HTTPS, IMAPS, and SMTP. As one of the most critical security protocols that are widely used to protect end users of online services, TLS relies on X.509 PKI to guarantee the identity of the remote server and the genuineness of the public key associated with it.

HTTPS[3], one of the most widely-used application layer protocols that utilize TLS protocol, is the foundation of omnifarious Internet activities including online shipping, blogging, social networks, and so on. HTTPS leans upon TLS to provide encrypted HTTP connection between the client and the server. Based on the large-scale Internet-wide measurement performed by Durumeric et al. in[4], the number of hosts serving HTTPS, by the end of their scan in August 2013, has grown to 24,442,824.

When a client tries to connect with a server through HTTPS, a TLS handshake is performed. As the first step, the server needs to send over an X.509 certificate signed by a CA. Afterward, this certificate is used by the client to identify and authenticate the server against the X.509 trust chain until the root CA of the certificate is found in the so-called "root certificate store" of the client's computer.

The certificate typically includes the issuer (who signed this certificate), the subject (who is being authenticated by this certificate), the public key corresponding to the subject, the digital signature provided by the issuer, signature algorithm of the certificate, not before and not after fields specifying the valid period of the certificate, certificate version, serial number, and extension fields. The clients are supposed to verify the certificate based on the following criteria. First, the subject presented in the certificate should be consistent with the website to which the client is talking. Second, the certificate should be within its valid period specified by NotBefore and NotAfter field. Third, the trust chain from the certificate to a trusted root CA should be established. Once the certificate is verified, the public key linked in the certificate is used to create a temporary credential between the client and server, which is then used to establish an end-to-end encrypted communication channel.

1.2 Phishing Definition

Phishing is an online crime that tries to trick unsuspected users to expose their sensitive (and valuable) personal information, for example, usernames, passwords, financial account details, personal addresses, SSN, and social relationships, to the miscreant, often for malicious reasons. Phishing is normally perpetrated by disguising as a trustworthy entity in Internet communication which is achieved by combining both social engineering and technical tricks. The instruments frequently exploited by attackers includes sending out spoofing emails and putting up deceiving websites to entice users to expose information. The spoofing emails usually purport to be from legal businesses, intended to lead users to counterfeit websites that lure the user to input sensitive information.

1.3 Current Phishing Landscape

The Internet has become an increasingly attractive place for miscreants. As is pointed out by 2013 Microsoft Computing Safety Index[5], the annual impact of phishing worldwide could be as high as 2.4 billion USD. Based on APWG's[6] Phishing Activity Trends Report on the 4th quarter of 2017[7](referred to as "the report" later), there were 233,613 unique reports of phishing attacks submitted to APWG during Q4, 2017.

Meanwhile, the report also mentioned an increasing trend in utilizing HTTPS infrastructure on phishing websites, indicating that Internet miscreants are deceiving end users more efficiently by turning existing security tools against end users. The report stated that by the fourth quarter of 2017, more than 30% of phishing attacks were hosted on websites that had HTTPS certificates. Furthermore, according to a PhishLabs report[8], over 80% of the users believe the green lock icon on the browser URL bar means the webpage is legitimate and safe. The confusing "Secure" label given by modern browsers to HTTPS verifiable websites, even if it is a fake one, is only making the situations worse, as shown in Figure 1.1. For comparison, Figure 1.2 gives a screenshot of the real PayPal login page, which is almost the same as the fake one visually.

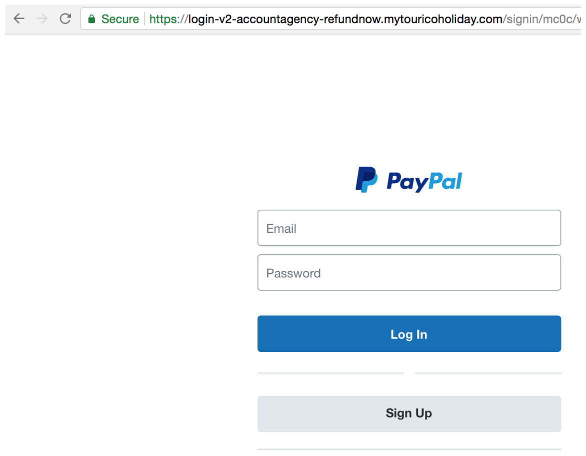

Secure
https://login-v2-accountagency-refundnow.mytouricoholiday.com/signin/mc0c/v
PayPal
Email
Password
Log In
Sign Up

Figure 1.1. Secure Label Shown in Browser Address Bar for a Fake PayPal Page

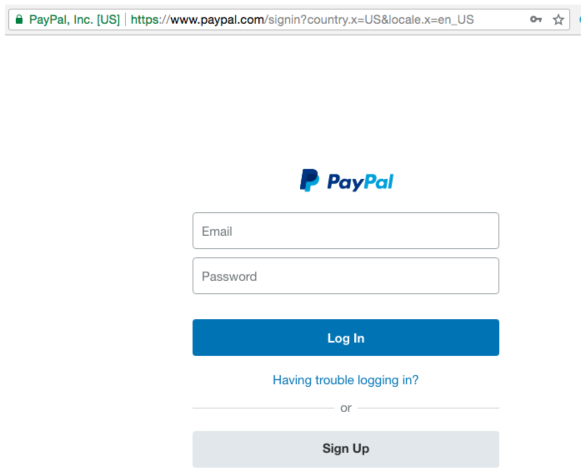

PayPal, Inc. [US]
https://www.paypal.com/signin?country.x=US&locale.x=en_US
PayPal
Email
Password
Log In
Having trouble logging in?
or
Sign Up

Figure 1.2. Secure Label Shown in Browser Address Bar for the True Paypal Page

In summary, phishing has become an important issue regarding Internet security whose efficient countermeasure is yet under discussion.

1.4 Our Work

Currently, a lot of existing tools, encapsulated in browsers, search engines or applications, such as SafeBrowsing from Google[9] and SmartScreen[10] from Microsoft, try to inform a user that a specific URL the user is about to visit has been identified as unsafe or malicious. This is realized by matching the URL being visited with blacklists constructed by the security community. Those blacklists are accumulated using various techniques, ranging from user reporting to web crawlers with site content analysis to automatic classification based on heuristics or machine learning classifiers. However, many malicious websites can still sneak through such protection systems, which can be the consequence of a number of reasons:

(1) The website is too new and thus has not been scanned or analyzed by any mechanisms yet.

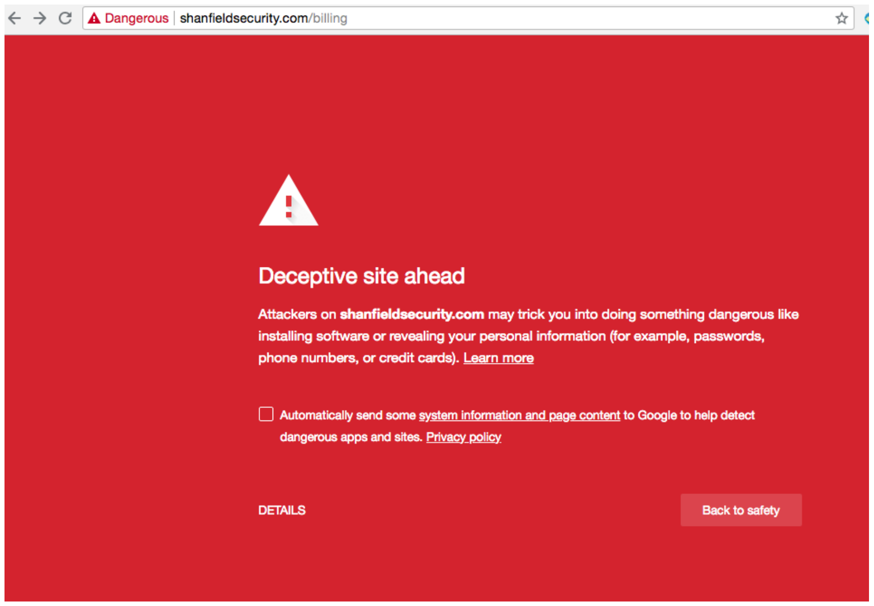
(2) The website has been incorrectly analyzed, either due to the imperfection of mechanisms or the countermeasures against detection taken by the attackers, e.g. "cloaking"[11], or abusing the legal short URL services[12].

There do exist some systems aiming at addressing the issue of incomplete blacklists by using real-time client-side evaluation against the content or behavior of the website when an end user visits it. However, those systems suffer from run-time overhead. Besides, depending on the nature of the attack, clients may have already been exposed to the threats of such malicious websites since the contents have been downloaded before the analysis begins.

The most critical component of defense against phishing is accurate detection of phishing websites in a timely manner. Successful recognition and blacklisting of phishing URLs would result in end users receiving a warning while being deceived to visit the phishing site. Once a striking warning such as the one presented in Figure 1.3 is displayed, it is highly likely that users would decline the login/data-input requests or malicious payload-downloading popups in phishing sites.

Hence, in this thesis, we propose an effective detection system that crawls websites and automatically discovers malicious pages. We intend our system to be used by a blacklist provider who can automatically compile and maintain an up-to-date blacklist of malicious URLs. Our system is equipped with a plentiful set of features that reflect various types of essential characteristics of the webpage content or behavior, which are impossible or difficult to be camouflaged by the miscreants. This system can proactively crawl and evaluate a given URL, labeling it to be phishing/malicious or legitimate, based on a trained classifier. Further, crawling is done from distributed vantage points, which allows the system to collect novel features and achieve higher accuracy and quicker recognition speed. By avoiding manual analysis, the blacklist can achieve better coverage and timeliness. Also, client-side analysis is avoided, thus removing both the runtime overhead and risk of downloading suspicious content to personal device.

Specifically, in this thesis, we propose a novel phishing detection system viewing the phishing detection task as a binary classification problem where the positive class is phishing and negative class is legitimate, which utilizes supervised learning algorithm employing a rich set of features, some of which are hard or impossible to collect from



Deceptive site ahead

Attackers on **shanfieldsecurity.com** may trick you into doing something dangerous like installing software or revealing your personal information (for example, passwords, phone numbers, or credit cards). Learn more

Automatically send some system information and page content to Google to help detect dangerous apps and sites. Privacy policy

Figure 1.3. Deceptive Site Warning of Chrome

the client side. Our end-to-end system empowers the blacklist provider to proactively detect and update phishing URL blacklist promptly with high accuracy. The set of features utilized by the machine learning algorithm is derived from a broad scope of information crawled by a distributed system of widely dispersed vantage points. Those features are extracted from raw probing data including the URL string, HTTP headers, DNS records, server status, network traces, WHOIS information, and redirection process. We demonstrate that by taking advantage of carefully selected features that indicate diverse aspects of the phishing content distributor, including physical characteristics, network characteristics and programming implementation characteristics, our system is robust to evasion attempts of miscreants.

Although some of the features used by our classifier have been discussed in previous works, based on our knowledge, this is the first study that combines so many diverse characteristics and collects data from more than a million websites, achieving highly accurate phishing detection. More importantly, our system employs many features that reflect the physical aspects of the servers hosting malicious contents, which are hard to forge, drastically raising the bar for attackers to sneak through the detection system.

We evaluate the proposed mechanism with a dataset containing more than 100,000 phishing URLs and 1,000,000 legitimate URLs. The phishing URLs are downloaded from PhishTank[13], which is a free blacklist provider operated by OpenDNS[14] while the legitimate URLs are a snapshot of Majestic top one million list[15]. According to the experiment results, the proposed system can obtain an overall prediction accuracy of 98% with very low false positives, while still enjoying the benefits of high classification power for both phishing and legitimate classes.

The remainder of this thesis is organized as follows. In Chapter 2, we discuss related work regarding phishing and spam detection with a similar purpose as ours. In Chapter 3, we describe the features used by our classifier and the intuition behind them. In Chapter 4, we present the complete system design as well as the experiment setup. In Chapter 5, we analyze the results of our experiments, explaining the implications of the performance and important features. Besides, we compare our methodology with a prior similar study, showing that our technique produces significantly higher classification accuracy. In Chapter 6, we describe the conclusions along with some future research directions.

2 Related Work

Our work in this thesis focuses mainly on detecting phishing websites with machine learning. There has been quite some effort regarding similar topics such as malicious domain blacklisting and email spam filtering. Furthermore, it is increasingly popular to utilize machine learning in these areas. Existing malicious websites detection approaches[16–28] can be mainly divided into two categories based on the features leveraged: static feature based approaches[16–23] and dynamic feature based approaches[24–28]. Static feature based approaches[16–23] rely on features extracted from the URL, page content, HTML DOM structure, domain-based information (such as WHOIS and DNS records) and so on. Alternatively, dynamic feature based solutions[24–28] primarily focus on analyzing behaviors captured when the page is loaded and rendered, or investigating system logs when some scripts are executed. In this thesis, we concentrate on exploiting static features.

2.1 Heuristic Based Methods in Proactive Blacklisting

Fukushima et al.[16] discussed the idea of evaluating reputations of IP address blocks and registrars used by attackers to perform proactive blacklisting of unknown malicious websites. They measured the prevalence of IP address blocks and registrars intensively used by attackers based on around 4,000 examples gathered from the Malware domain list[29]. IP address blocks are then combined together with registrars to measure the reputation level of websites, after which those with low reputations are reported as suspicious. However, their method potentially suffers from a high false positive rate, i.e., classifying benign domains as malicious ones. A lot of legitimate websites will be treated as malicious ones only because they share the same registrar or IP blocks as some vicious websites.

Felegyhazi et al.[17] proposed a proactive blacklisting method based on inferring the domains registered by the same miscreants whose domains have shown up in confirmed blacklists. This inference is achieved with the help of WHOIS and DNS zone file information. This method considers domains that are registered at the same time, or change to the same name server at the same time, as a known illegal domain. Based on their experiment, 73% of the inferred domains showed up in blacklists later on. However, it can only infer attacks based on existing blacklisted domains as seed, meaning it is insufficient when coping with new attacks that are not related to existing malicious domains. In contrast, our methods can be used for capturing any malicious domains as soon as the classifier is trained, regardless of the relationship between domains.

2.2 Machine Learning Based Methods

2.2.1 Malicious Domain Detection

Dong et al.[18] have investigated a comprehensive list of features that can be extracted from X.509 certificates. With using top 100,000 websites from Alexa top one million websites[30] as their legitimate examples and phishing URLs downloaded from PhishTank[13] as phishing examples, precision and recall of 95.5% and 93.7% are attained for the phishing category with a Random Forests classifier. However, these statistics are doubtful when it comes to the current Internet environment, as will be discussed in detail in Section 5.6. Our feature list, in contrast to their work, leverages information from not only certificates, but also several other dimensions of a given website, including server characteristics, DNS responses, network performance and so on.

Xiang et al.[19] proposed a multi-layer machine learning framework called CANTINA+, in which features exploited from URL, HTML DOM, search engines and third-party services such as PageRank are used for phishing websites detection. In total, 10 out of 15 features in their framework are obtained from textual patterns and formats of HTML or URL. Rosiello et al.[20] proposed another HTML DOM layout similarity based approach for phishing detection used by browsers. However, there are some severe drawbacks for the methods relying on DOM or lexical information. First of all, lexical characteristics of URL or HTML content can be easily manipulated by the attacker to deceive the classification system. Meanwhile, in order to perform the classification, the page content needs to be downloaded, during which the malicious component might already be introduced and downloaded to the user machine if those methods are used within the user's browser. Additionally, as has been shown by several studies[11,31–33], cloaking technology (cloaking means the website serves different content to different visitors based on pre-defined criteria for the seek of bypassing the security system or deceiving the crawler) is used by a non-negligible portion of malicious websites for the purpose of misleading crawler services like Google to put them on higher rank or getting better exposure in the search result. This phenomenon makes the PageRank or keyword search results which CANTINA+ depends on less reliable.

Ma et al.[21] attempted to circumvent this issue by not including any page content related features for classification. Beyond that, they also took advantage of tens of thousands of features generated from IP address, WHOIS records and domain names, with a majority of features being lexical features constructed using “bag of words” technique. One drawback of their method is that the resulting model is hard to interpret due to the massive number of algorithmically generated lexical features. On the contrary, features within our system are all comprehensible, making the resulting classifier explainable.

2.2.2 Email Spam Filtering

Ouyang et al.[22] proposed a multi-stage pipelined spam email detection system using machine learning approach, which contains an extensive list of network features. They analyzed their methodology with email data collected in over two years, consisting of over 1.4 million messages, and reported a true positive rate between 12% to 77% using the Decision Tree algorithm. Our work in this thesis differs from their work in several aspects. Firstly, we use an advanced machine learning algorithm, i.e., Random Forests, for the classification purpose. Secondly, we have used a lot of features other than network features for the classification purpose, which makes classification potentially more stable considering that network quality might have a significant impact on the collected network features. Thirdly, our work is focused on phishing URL classification, which has a broader range of application scenarios.

Basnet et al.[23] focused on leveraging features extracted from email content for spam email detection and examined several different machine learning algorithms. However, their method also suffers from the same issue as mentioned for Xiang et al.[19] and Rosiello et al.[20] above. Besides, their investigated samples are limited, including only

973 phishing emails and 3027 legitimate emails, making their classification accuracy number less trustworthy.

In conclusion, our work offers different point of views than existing studies in the subjacent ways.

First, in terms of the dataset, we collect and test the system against a large number of example URLs, in which the phishing URLs are downloaded from PhishTank[13] and the legitimate URLs are downloaded from Majestic top one million list[15]. This improves the value of the performance evaluation and corresponding analysis.

Second, we significantly increase the richness and diversity of features examined by the machine learning algorithm. Via introducing features reflecting different aspects of the target URLs, a comprehensive view of the target URLs as well as the underlying hosting infrastructures is observed, which significantly improves the performance of the proposed system and raises the bar for a miscreant to circumvent such a detection system. Furthermore, synthetical understanding of the current phishing landscape is gained by analyzing the learned models.

3 Feature Selection

In this section, we present the features we selected for the machine learning algorithm. Effective machine learning highly relies on the discerning of an informative set of features. Our feature selection starts with observations of the natural behavioral differences between phishing websites and legitimate websites, selected carefully from both application and infrastructure level. We also absorb useful features and concepts from previous works on related topics such as phishing detection, proactive blacklist generation and email spam detection. Eventually, our feature set can be divided into eight groups based on their origin. The intuition and meaning behind each feature will be discussed in this chapter.

3.1 Certificate Features

As mentioned above, X.509 digital certificates are extensively used in modern Internet communication to provide end-to-end identity authentication. However, except for the traditional client-side verification of the certificate itself, the information embedded in the certificate can provide a large number of useful features to deduce the nature of the server. An extensive list of features that can be extracted solely from the X.509 digital certificate has already been proposed by Dong et al.[18]. Our certificate feature set is gathered by reusing most of their features and logical groups, while adding some new ones. Meanwhile, we have also removed some of their features due to their deficiencies in practice. For example, *isHighlyTrustedCA(Issuer)*, which checks if the issuer of the certificate is VeriSign or Thawte, is removed since it is hard to judge which CA can be indeed more highly trusted than others in the root store, and also because this feature would throw off the classifier if one of these highly trusted CAs is compromised.

Furthermore, our certificates are collected around the world using eight distributed AWS EC2 instances[34], with the purpose of getting an approximate total number of different certificates used by the domain, which could be an indicator of the level of distributivity of the underlying hosting infrastructure.

Here we enumerate through the features extracted from certificates in groups.

3.1.1 Certificate Extensions

Certificate extensions are a list of fields that are optional for a valid certificate. Although a clear standard does not exist, extensions can still potentially reveal critical underlying patterns of fraudulent websites potentially. Certificate extensions could include extra information such as alternative names for the issuer and subject, certificate and public key usage policies, Certificate Revocation List (CRL), CRL distribution points, extended validated certificate, and so on. The features are retrieved in the following way. Firstly, the total number of extensions is recorded as a numerical feature. Then the existence of 13 most frequently used extensions is checked, each kept as a boolean feature. The following is the list of features retrieved from certificate extensions:

- *authority_info_access*: If the authorityInfoAccess field exists in the certificate's extension.

- *authority_key_identifier*: If the authorityKeyIdentifier field exists in the certificate's extension.
- *subject_key_identifier*: If the subjectKeyIdentifier field exists in the certificate's extension.
- *basic_constraints*: If the basicConstraints field exists in the certificate's extension.
- *certificate_policies*: If the certificatePolicies field exists in the certificate's extension.
- *extended_key_usage*: If the extendedKeyUsage field exists in the certificate's extension.
- *CRL_distribution_points*: If the crlDistributionPoints field exists in the certificate's extension.
- *freshest_CRL*: If the freshestCRL field exists in the certificate's extension.
- *is_extended_validation*: If the isExtendedValidation field exists in the certificate's extension.
- *key_usage*: If the keyUsage field exists in the certificate's extension.
- *issuer_alt_name*: If the issuerAltName field exists in the certificate's extension.
- *subject_alt_name*: If the subjectAltName field exists in the certificate's extension.
- *subject_directory_attributes*: If the subjectDirectoryAttributes field exists in the certificate's extension.
- *extension_count*: The total number of extension fields contained in this certificate.

3.1.2 Issuer and Subject/Domain Information

The two most important fields within a certificate are the Issuer and Subject. Issuer refers to the CA who signs and certifies the identity of the current domain using the certificate. Subject refers to the domain or organization who is being certified by the digital certificate. Both Issuer and Subject share the same schema, known as 'Distinguished Names' (DNs), containing the detailed name, location, and contact information of the issuer and the subject, respectively. They each can provide a set of subfields including country, state, city, common name, organization name, organization unit, and email. Three groups of features are developed from the subfields of Issuer and Subject.

Whitelisting and Blacklisting for Issuer and Subject. This group contains three features, *is_trusted_issuer*, *is_prohibited_issuer* and *is_prohibited_subject*.

is_trusted_issuer indicates whether the issuer of this certificate is one of the CAs trusted by Mozilla certificate store[35]. As mentioned above, in X.509 PKI, all CAs are trusted equally, allowing any CA to issue a digital certificate to any website. Hence including clues of the certificate issuer's reputation could be useful for phishing detection. A list of CAs trusted by Mozilla, one of the primary browser providers, is used as the set of trusted issuers, considering that it represents the state of the art certificate validation experience. We argue that *is_trusted_issuer* is more feasible than *isHighlyTrustedCA(Issuer)* because, to the best of our knowledge, there is no previous work showing that the two "highly trusted" CAs (i.e., VeriSign and Thawte) chosen by Dong et al. are superior to other CAs existed in Mozilla trust store. Moreover, if any accident lead to those highly trusted CA being compromised, such as the DigiNotar[36] and Comodo[37] case, *isHighlyTrustedCA(Issuer)* would be harmful to the classification. On the other hand, checking the issuer against Mozilla trusted CAs illustrates if the issuer is trusted by one of the most critical contributors to the Internet community whose product serves millions of users, indicating the public consensus of the reliability of the issuer to some extent.

is_prohibited_issuer and *is_prohibited_subject* checks if the issuer or subject of this certificate belongs to one of the following categories.

(1) The name is a local network address: 192.168.*.*, 10.*.*.*, 127.0.0.1 and localhost.

(2) The name is the wildcard character "*".

(3) The name is "default" or "none".

Although there is no definite list of strings that should be avoided in the Issuer and Subject field, checking the existence of the limited set of values listed above still exposes some valuable information about the certificate. For example, appearance of using a local network address or "default" as issuer or subject in a certificate suggests that this could be a self-generated one. And the usage of a wildcard character in issuer and subject field obviously demonstrates a poor practice in certificate signing, implying the issuer and subject is less reputable or even suspicious. Similar to *is_trusted_issuer*, *is_prohibited_issuer* and *is_prohibited_subject* could evoke some potential pattern differences between phishing and legitimate websites.

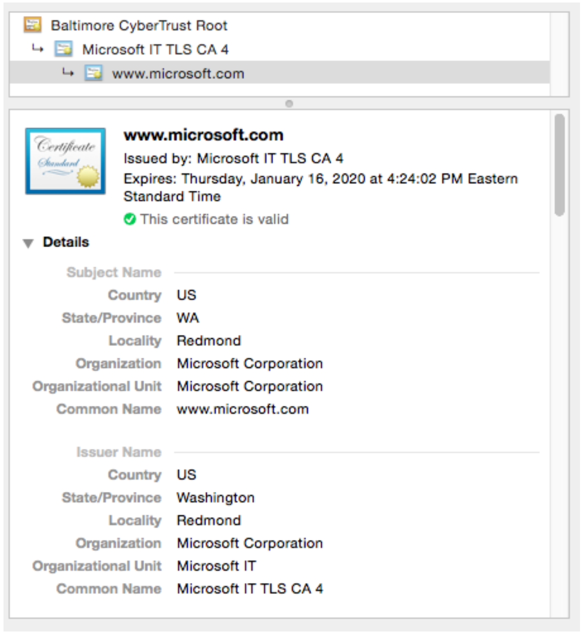
The list of features extracted is presented as follows.

- *is_trusted_issuer*: If the issuer of this certificate is trusted based on Mozilla trust store.
- *is_prohibited_issuer*: If the issuer of this certificate contains prohibited strings listed above.
- *is_prohibited_subject*: If the subject of this certificate contains prohibited strings listed above.

Relationship Between Subfields. There is no clear standard published regarding the content and relationship that a CA should follow for the subfields inside Issuer and Subject section of a certificate when signing a digital certificate. However, checking the patterns revealed by the relationships between subfields can still be helpful to discover different CA characteristics. Moreover, those patterns could even serve as evidence to detect technically valid certificate that is signed by unethical or compromised CAs, as is mentioned in a report from Microsoft in 2014[38].

For example, the similarity between the common name of Issuer and Subject and the similarity between the common name of Issuer and domain name of the website is a good indication for whether this is a self-signed certificate or not, even if the trust chain of the certificate could be validated. High similarities insinuate a self-signed certificate is being used. Nevertheless, we do need to note that large-scale organizations might operate their own CA. For example, Microsoft and Google both run intermediate CAs and issue themselves certificates. Figure 3.1 shows the trust chain of a certificate belonging to Microsoft, in which the Issuer and subject are both Microsoft, generated via the Chrome browser.

Another good differentiator between legitimate and suspicious sites is the similarity between the domain name and name of the certificate Subject. In accordance with RFC6125[39] and common practice, the certificate presented by a domain, in order to certify its identity, must include the corresponding domain name either in the Subject field or the Subject Alternative Name extension. Therefore if the domain name matches with the common name of the Subject or Subject Alternative Name in the extension could be inferred as a direct hint of trustworthiness.


Baltimore CyberTrust Root
↳ Microsoft IT TLS CA 4
↳ www.microsoft.com

www.microsoft.com
Issued by: Microsoft IT TLS CA 4
Expires: Thursday, January 16, 2020 at 4:24:02 PM Eastern Standard Time
This certificate is valid

Details

Subject Name	
Country	US
State/Province	WA
Locality	Redmond
Organization	Microsoft Corporation
Organizational Unit	Microsoft Corporation
Common Name	www.microsoft.com

Issuer Name	
Country	US
State/Province	Washington
Locality	Redmond
Organization	Microsoft Corporation
Organizational Unit	Microsoft IT
Common Name	Microsoft IT TLS CA 4

Figure 3.1. Microsoft's Own Certificate Authority

In addition, a high similarity between the organization unit and common name implies a legitimate site as well, since the two fields are supposed to refer to the same entity. We measure similarity between string-valued fields using matching ratios, which in this thesis are all calculated using the default sequence matching function provided by *difflib* of python 2.7[40].

In the following we present the list of features corresponding to the relationship between subfields.

- *match_issuer_o_cn*: Matching ratio of organization and common name of certificate issuer.
- *match_issuer_o_ou*: Matching ratio of organization and organization unit of certificate issuer.
- *match_subject_o_cn*: Matching ratio of the organization and common name of certificate subject.

- *match_subject_o_ou*: Match ratio of the organization and organization unit of certificate subject.
- *match_issuer_subject_cn*: Matching ratio of the common name of certificate issuer and subject.
- *match_website_issuer_cn*: Matching ratio of website domain with issuer's common name.
- *match_website_altname*: Highest matching ratio of website domain with the each of the alternative name in the list of alternative names field of certificate extension.
- *match_website_subject_cn*: Matching ratio of website domain and subject's common name.

Other Subfields. As the last step in extracting information from Issuer and Subject fields, the length of the state, city, common name, organization, organization unit and email is recorded. Presumably this information could serve as an indication of the content filling patterns of the CA who signs the certificate. For example, DigiCert does not provide state and organization unit in the Issuer field. However, our primary motivation to select these length features is to enable fair comparison with Dong's et al. study (see Section 5.6) which used these features. Note that the length of the country is not kept since the length of all country codes are the same. However, the country code itself is kept as a feature describing from which country this certificate comes. The intuition behind those features is that different CAs might have different behaviors in filling those subfields. Therefore the underlying patterns could be used by our machine learning algorithm for classification. The list of features extracted is presented as follows.

- *len_issuer_st*: Length of issuer's state.

- *len_issuer_city*: Length of issuer's city.
- *len_issuer_cn*: Length of issuer's common name.
- *len_issuer_o*: Length of issuer's organization.
- *len_issuer_ou*: Length of issuer's organization unit.
- *len_issuer_email*: Length of issuer's email.
- *len_subject_st*: Length of subject's state.
- *len_subject_city*: Length of subject's city.
- *len_subject_cn*: Length of subject's common name.
- *len_subject_o*: Length of subject's organization.
- *len_subject_ou*: Length of subject's organization unit.
- *len_subject_email*: Length of subject's email.
- *cert_issuer_c*: Country of the certificate issuer.
- *cert_subject_c*: Country of the certificate subject.

3.1.3 Chronological Information

NotBefore and NotAfter are two fields in a certificate specifying its validity period. As a common practice, a certificate with a valid period that is too long is considered as less secure. Hence the time difference between NotBefore and NotAfter is used as a feature. In addition to that, the time differences between the time when we obtain the certificate (referred to as "downloading time" of the certificate below) and those two fields are also calculated. The difference between the downloading time and NotBefore, to some extent, could serve as an indicator for suspiciousness of the domain, with the intuition that certificates which have been used for a while are more credible than a newly issued one. We also check if the certificate has expired, that is, being captured after the time specified in its NotAfter field, as this behavior is illegal and highly suspicious.

The list of chronological features is presented as follows.

- *diff_notbefore_timestamp*: Time difference in seconds between the timestamp the certificate is captured and the NotBefore field in the certificate.
- *diff_notafter_timestamp*: Time difference in seconds between the timestamp the certificate is captured and the NotAfter field in the certificate.
- *diff_notbefore_notafter*: Time difference in seconds between the NotBefore and NotAfter field in the certificate.
- *has_expired*: Has the certificate expired or not.

3.1.4 Other Certificate Features

Other certificate features can be extracted from fields including SerialNumber, SignatureAlgorithm, and CertificateVersion. The content in the SerialNumber is not suitable to be recruited as a feature directly as it is a randomly assigned unique number. The length of it, however, is considered as a feature since it could imply some underlying pattern of CAs. Along with each certificate, a digital signature is generated to prove the authenticity of the certificate and public key with a specific algorithm provided in SignatureAlgorithm field. The algorithm name is recruited as a feature for that the algorithm itself could be an important factor indicating the level of cryptographic security of the certificate. Reputable domains and CAs tend to avoid using insecure algorithms. The version of the certificate could also reveal some information about the structure of the certificate as different versions have different optional extensions and algorithm choices. So the certificate version is included as a feature.

In addition to the content of the certificate fields, we also recruited a new feature which, based on our knowledge, has not been proposed by any other research yet. We count the total number of distinct certificates being captured by eight AWS EC2 instances distributed around the world and use it as a feature. This is considered as an indication of the scale for the web servers. More than one different certificate being captured indicates a globally distributed server structure and thus potentially a more legitimate organization.

Finally, whether or not the target domain has an associated certificate is recorded as a binary feature. This feature is added to improve the training and classification efficiencies of machine learning classifiers since it could mark all websites without certificates explicitly with only one variable.

The features are presented as follows.

- *len_serialnum*: Length of the serial number of this certificate.
- *signature_algorithm*: The signature algorithm of this certificate.
- *cert_version*: Version of the certificate.
- *cert_count*: The total number of different certificates our crawler get from eight AWS EC2 instances around the world.
- *has_certificate*: Whether or not a certificate could be obtained for the targeted domain.

3.2 Server Status Features

Whatever methodologies are used for phishing websites to deceive end users, the miscreant needs to host them on some physical servers. Those physical servers, which are used for short-live malicious purposes, are possibly botnets or hosts that are not maintained professionally. Thus features reflecting the status of the physical server could be extracted to detect malicious sites. These features are especially desirable because they are hard for a miscreant to circumvent.

3.2.1 Port Status

First, a port scan is performed against the domain of each URL in our example sets using nmap[41], targeting the top 50 most commonly used ports based on nmap data. The status of each port (i.e., whether it is open or closed) is recorded as a binary feature. Generally, legitimate servers could be expected to have well-defined pattern of which ports they open and a limited number of open ports overall for the sake of security, while malicious websites would be held on servers or botnets with inferior or no maintenance at all, thus leaving more ports open by default. Besides, those malicious servers could potentially host a lot of different malicious sites and applications at the same time, leaving more ports open for different services or domains, due to budget limitation or other reasons. On the other hand, we can speculate that legitimate servers are usually dedicated to a small number of tasks in production, restricting the total number of open ports. Thereby the total number of open ports is also kept as a feature.

The top 50 commonly used ports scanned by the crawler are: 21, 22, 25, 53, 67, 68, 69, 80, 110, 111, 123, 135, 136, 137, 138, 139, 143, 161, 162, 443, 445, 514, 518, 520, 593, 631, 993, 995, 999, 1025, 1026, 1433, 1434, 1645, 1646, 1723, 1812, 1900, 2049, 2222, 3283, 3389, 3456, 4500, 5060, 5353, 5900, 8080, 20031, 32768.

Port features are presented as follows.

- *port:{port_number}*: Open/Close status of each of the top 50 commonly used ports based on nmap scan results.
- *open_ports_count*: Total number of open port based on nmap scan results.

3.2.2 Server OS

Except for port scanning, nmap is able to give a server operating system fingerprinting as well. Tu et al. has presented the effectiveness of using OS as a feature for anti-spam purpose[22,42]. Thus we extract the server operating system and use it directly as a feature. The operating system is classified as Linux, Windows, OpenBSD, FreeBSD, HP-UX, Mac, Solaris, Others and Unknown, in which UNKNOWN means nmap does not return any server OS name in its scan result.

- *server_os*: Operating system of the server based on the nmap scan results.

3.2.3 Average Distance Measurement

As modern websites are paying more and more attention to high availability and scalability, domain administrators have greater motivation on migrating to a distributed hosting infrastructure, e.g., CDN or Cloud Service. This trend, as a consequence, could be used for distinguishing malicious websites, as miscreants may not have much motivation to set up distributed hosting infrastructure for a short-lived phishing website, because of both the effort and costs. We mainly recruit three features with the purpose of measuring both network and physical distances between our vantage points and the servers. The features included are average geographical distance, average RTT and average hop distance.

Average Geographical Distance. First of all, average geographical distance from our distributed vantage points to the servers responding to our requests is calculated. The intuition is that popular websites would have servers distributed around the world and perform load-balancing based on user location for the sake of better user experience, while malicious websites may not care about those aspects. The geographical distance is calculated using latitude and longitude of the source and destination IP address based on GeoLite2[43] data. Hence for legitimate websites, this average distance should be smaller. The source and target IP addresses are obtained from *traceroute* recorded from a given vantage point to the target domain. The reason for using traceroute is that public IP addresses assigned to EC2 instances are not everlasting and can be changed in cases such as instance reboot. Note that the first few hops in the traceroute results are always unreachable, indicated by a sequence of *****, when *traceroute* command is executed from our EC2 instances. This is due to that machines within the AWS infrastructure do not respond to any ICMP messages by default. Thus, based on the traceroute results, we utilize Algorithm 1 to calculate the best estimate of the geographical distance by using the very first address that is reachable as the source IP. Our intuition here is to get the border routers of the AWS network as the source addresses. Presumably a border router of the AWS network should have an IP address pertaining to Amazon and expose short RTT to EC2 vantage points we used. In order to prove the intuition, we randomly picked 40 IP addresses selected by the algorithm from traceroute results and manually checked them through GeoIP2 City Database Demo page[44], which provides the organization a given IP belonging to. All of the 40 IPs picked belong to Amazon. Further, the RTTs from our vantage points to those IPs are within 0 to 3 milliseconds, indicating they are nearby to our vantage points.

The function used to calculate distance based on the latitude and longitude of source and target, *calculateSphericalDistance*, is implemented based on the great circle distance function[45], treating earth as a sphere whose radius is 6731 kilometers. Let ϕ_s, λ_s

Algorithm 1 Geographical Distance Calculation

Input: S is the source address, T is the destination address
Output: Spherical distance between S and T

```
1:  procedure CALCULATESPHERICALDISTANCE($S, T$)  ▷ Calculate the spherical distance between source and target
2:      $trace\_path \leftarrow traceroute(T)$
3:      $target\_ip \leftarrow getLastHopIP(trace\_path)$
4:      $target\_location \leftarrow geoip.resolveIP(target\_ip)$
5:      if $target\_location = null$ then      ▷ If the target IP cannot be resolved to a geographical point
6:          return null
7:      end if
8:      $source\_location \leftarrow getFirstResolvableIP(trace\_path)$
9:      if $source\_location = null$ then      ▷ If the source cannot be resolved to a geographical point
10:         return null
11:     end if
12:     $sla \leftarrow source\_location.coordinate.latitude$      ▷ latitude of the source
13:     $slo \leftarrow source\_location.coordinate.longitude$     ▷ longitude of the source
14:     $tla \leftarrow target\_location.coordinate.latitude$      ▷ latitude of the target
15:     $tlo \leftarrow target\_location.coordinate.longitude$     ▷ longitude of the target
16:     $disntace \leftarrow calculateSphericalDistance(sla, slo, tla, tlo)$
17: end procedure
```

Algorithm 2 getFirstResolvableIP(T)

Input: T is the traceroute output
Output: The first IP address in the hops of traceroute output that can be resolved by geoip

```
1: function GETFIRSTRESOLVABLEIP($T$)
2:     for $hop \in T.hops$ do:
3:         $location \leftarrow geoip.resolve(hop.ip)$
4:         if $location \neq null$ then
5:             return location
6:         end if
7:     end for
8:     return null
9: end function
```

and ϕ_t, λ_t be the latitude and longitude in radians of the source and target points. Also, let $\Delta\phi$, $\Delta\lambda$ be the absolute differences between the radians. The distance, or arc length, is calculated using Equation 3.1 with radius r estimated as 6731 kilometers, in which the central angle in radians between the two locations, labelled as $\Delta\sigma$, is calculated using Equation 3.2.

$$d = r \cdot \Delta\sigma \tag{3.1}$$

$$\Delta\sigma = \arccos\left(\sin\phi_s \cdot \sin\phi_t + \cos\phi_s \cdot \cos\phi_t \cdot (\cos\Delta\lambda)\right) \tag{3.2}$$

Average RTT (Round Trip Time). RTT is one of the most straightforward metrics for measuring connection quality because most of the users have a low tolerance for latency. Hence the average RTT calculated using RTTs collected from eight distributed vantage points using *traceroute* is used as a feature. Generally speaking, a low average RTT indicates a capable server system that responds quickly to user requests, which is intuitively more likely to be a legitimate server system. Since most of the legitimate websites care about user experience, attempting to optimize the performance using relatively powerful and professional machines, the average RTTs to legitimate websites from all the vantage points would intuitively be smaller and follow different distribution pattern than phishing websites.

Average Hop Distance. Similarly, average hop distance is also calculated based on hop distances captured from the vantage points around the world. Here hop distance refers to the number of router hops on the path from a current vantage point to a given target server, and is probed using *traceroute*. Average hop distance reveals network topological information of where the target servers reside inside the Internet world relative to our vantage points. The servers which utilize distributed structure and route user requests based on geographical information would achieve a low average hop distance. This type of request routing intuitively is an indication of legitimate websites.

In conclusion, the distance measurement features are presented as follows.

- *average_geo_distance*: The average geographical distance from our distributed vantage points to the server responding to requests.
- *average_rtt*: Average round trip time from our distributed vantage points to the website server.
- *average_hop_distance*: Average number of hops needed from our distributed vantage points to the remote host of the target domain.

3.2.4 Geographical Locations

The total number of different cities the target domain is located in is considered as a feature, given that this piece of information could shed light on the scale of the distributed server infrastructure. However, as mentioned in previous research[22], a large number of distinct locations could also be the result of large-scale botnet that is distributed around the world.

- *target_location_count*: The number of different cities where the server of the target domain locates based on our distributed crawling results.

3.3 HTTP Header Features

The HTTP header is returned by an HTTP server to the client along with the requested contents, which allows additional information corresponding to the requests and responses to be passed to the client. The HTTP header is a list of key-value pairs of which the possible keys and values are registered with IANA[46]. The content of the HTTP headers can be used to induce some clues regarding the underlying server implementation details, for example, cache control policy of the server, web framework used by the server, access control policy of the server, content type of the response. Out of the HTTP header content, three features that can be easily parsed are extracted.

Firstly, we counted the total number of different header fields in the server response. The intuition behind this feature is to acquire differences in underlying patterns in the HTTP response of malicious and legitimate websites. Considering that malicious websites are usually maintained with less effort, their HTTP header could be the default values of the web framework used by the miscreant. However, headers returned by a legitimate server could be developed with clear purposes.

Secondly, whether or not the cache-control field exists in the header is checked, with the intuition that legitimate websites may have fine-tuned cache-control logic.

Last but not least, the length of the server field inside the header is recorded. This feature might indicate some underlying pattern differences in the backend implementation of legitimate websites and malicious websites, respectively.

HTTP header features are presented as follows.

- *http_header_count*: Total number of header fields contained in the HTTP header returned.

- *exist_cache_control*: Whether the HTTP header contains a cache control field or not.
- *len_header_server*: Length of the server field within HTTP header.

3.4 Network Features

Network traffic and network-level characteristics between the vantage points and the target server can be a good indication of the underlying connection quality as well as the ability of the server to handle user requests. These features were investigated by Ouyang[22,42], my collaborator on our overall effort to detect malicious URLs. Thus we only present a brief description of the network features here. Readers who are interested in learning more about the network features can refer to Ouyang's previous work for details.

- *ttl*: TCP TTL value retrieved from the SYN/ACK packet accepted by the local vantage point.
- *advertised_window_size*: Size of the TCP advertised window field retrieved from SYN/ACK packet accepted by the local server.
- *3_way_handshake*: Time interval between sending out SYN packet and sending out ACK packet for the SYN/ACK packet received.
- *fins_local*: Total number of packets with TCP FIN flag set emitted by local vantage point.
- *fins_remote*: Total number of packets with TCP FIN flag set accepted by local vantage point.

- *max_idle_time*: The Longest time interval between two consecutive reception of packets on the local vantage point.
- *variance_packet_arrival_time*: The variance of time intervals captured between consecutive receptions of remote packets.
- *packets_received_to_packets_sent*: Total number of packets received by the local vantage point divided by the total number of packets emitted by the local vantage point.
- *rsts_local*: Total number of packets with TCP RST flag set emitted by local vantage point.
- *rsts_remote*: Total number of packets with TCP RST flag set accepted by local vantage point.
- *variance_rtt*: The variance of RTTs captured between consecutive two-way communications of local vantage point and remote server.
- *retransmission_local*: Total number of retransmitted packets emitted by local vantage point.
- *retransmission_remote*: Total number of retransmitted packets accepted by local vantage point.
- *packet_counts*: Total number of remote packets accepted by the local vantage point.
- *packet_rate*: Total number of remote packets accepted by the local vantage point divided by the persisted time of the connection.

3.5 URL Lexical Features

Recruiting URL lexical features for phishing and malicious website detection has been extensively discussed in many previous studies[19,21,47–49]. We reused those lexical features from previous studies since their feasibility in phishing detection has been proved.

Note that we only consider the features for domain names, since our negative examples (legitimate websites) only consist of top million domains.

3.5.1 Is IP Address Used as Domain

Sometimes the miscreants try to establish their malicious websites without being able to set up corresponding DNS entries, because of either the effort or the cost needed. The most straightforward solution, in this case, is using the IP address directly as the domain. On the contrary, a legitimate website would never want to do so because this would degrade the user experience, e.g., users would highly prefer visiting www.google.com rather than 172.217.6.14, which is much harder to remember. Thus, using IP address as the domain name is recorded as a feature.

3.5.2 Characters in Domain

Besides the IP address checking, we also calculate the number of dots, the number dashes, and the length of the domain as features since, according to previous studies, those numbers have different distribution patterns in phishing and legitimate domains.

URL lexical features are presented as follows.

- *IP_address_in_domain*: If the IP address is used directly as the domain name.
- *num_of_dot*: The number of dots (“.”) in the domain.
- *num_of_dash*: The number of dashes (“-”) in the domain.

- *len_domain*: Length of the domain.

3.6 WHOIS Features

WHOIS[50] records are the registration details of a given domain, including the creation date, updated date, expiry date, registrar of the domain, registrant of the domain, abuse contact of the domain, DNS servers of the domain and so on.

Previous studies[16,17,19,21,47,48,51] have demonstrated the practicality of using features extracted from WHOIS records in detecting phishing websites. We reuse features that have been proposed in previous works. However, as pointed out by previous studies, there are neither established standards on the format of WHOIS records nor consensual API for retrieving WHOIS records, making collecting and parsing WHOIS content burdensome.

3.6.1 WHOIS Chronological Information

First of all, the time difference in seconds between the downloading timestamp and domain creation time, the downloading timestamp and updated date, the downloading timestamp and expiry date is used as *domain_age*, *diff_update_time* and *diff_exipry_time*, respectively. Considering the fast-iteration nature of phishing websites, the patterns of those chronological values could be distinct from legitimate websites.

3.6.2 WHOIS Registrar

Meanwhile, previous research[16,17,21] has shown that attackers tend to abuse specific registrars intensively. Thus the registrar field of WHOIS record is also utilized as a feature.

3.6.3 Is Entry Locked

Lastly, whether the domain is locked or not is recorded as a feature. Locked domains are indicated by "Registrar lock" or "Client Transfer Prohibited" status in WHOIS entry. Domain names are normally locked to prevent from unauthorized changes or accidental transfers. A locked status indicates the domain is safer against misconduct than those that are not.

WHOIS features are presented in the following list.

- *domain_age*: Time difference in seconds between the domain creation time and the time when the WHOIS entry is downloaded.
- *diff_update_time*: Time difference in seconds between the update time and the time when the WHOIS entry is downloaded.
- *diff_expiry_time*: Time difference between the domain expiry time and the time when the WHOIS entry is downloaded.
- *is_entry_locked*: Whether or not the WHOIS entry for this domain is locked.
- *who_is_registrar*: The Registrar name specified in the WHOIS entry.

3.7 DNS Features

DNS plays a vital role in the Internet architecture. To deceive end users to visit phishing websites, a miscreant normally attempt to set up a page that looks like a legitimate website which then asks for sensitive personal information such as username and password, as is presented in Figure 1.1. In this case, the attacker needs to configure assorted DNS entries, e.g., NS and A records, for the phishing website. Otherwise, the attacker need to use IP address as the domain name which has been discussed in Section 3.5. If the corresponding DNS structure is set up for phishing websites, then the latent pattern difference between phishing websites and legitimate websites regarding the number of DNS records, type of records and TTL of records could be discovered through analysis of DNS content. Previous attempts of extracting features from DNS records have been discussed in prior works[21,51]. We reuse some of the previously proposed features while adding new features at the same time.

Some of the recursive name servers are known to strip off additional information of a DNS response. Thus, for each URL, we first perform a DNS NS query to get the domain's authoritative name server. Then we try to retrieve A records, NS records, A records for name server addresses in NS records, AAAA records, AAAA records for name servers, and PTR records. In this way the additional information in the DNS responses for a domain is retained. Furthermore, since the TTL value is used as features, querying directly from authoritative name servers gives back the initial TTL set by the domain administrators.

Note that we perform the DNS resolution against each domain from eight different AWS EC2 instances around the world. Querying from more than one places empowers the crawler with a comprehensive dataset for each domain.

3.7.1 A Records

We begin with examining the number of different type A records. A record stores the mapping relationship between a domain and its IP address. More than one unique type A entry indicates a complex hosting infrastructure for the domain whose contents could be load-balanced and served via multiple servers around different regions, such as CDN networks or cloud services. However, fast-flux network, which serves malicious contents through tens of thousands of hacked computers also has the possibility of serving several IP addresses for a single domain. Because hacked computers are unreliable, botmasters need to keep more than one A records for a single domain to make sure it is reachable. In this case, other features would need to provide extra information for distinguishing legitimate and phishing domains.

Next, the maximum TTL value of all the type A entries collected is retained. TTL indicates how long should an entry stay in the cache before it is deleted. This indicates the dynamics of the server structure. Smaller TTL values give the hosting infrastructure ability to change servers faster to guarantee the availability of the domain.

The list of A records related features is presented as follows.

- *num_unique_A_record*: The number of unique A records in all DNS responses.
- *max_dns_a_ttl*: Maximum TTL value among all A records.

3.7.2 NS Records

NS records keep the mapping between a domain and its name servers. This is also an indication of the hosting infrastructure for the domain. Similar to A records, we also examine the number of unique NS records and the maximum TTL among all NS records.

The list of NS records related features is presented as follows.

- *num_unique_ns_record*: Number of the unique NS records in all DNS responses.
- *max_dns_ns_ttl*: Maximum TTL value among all NS records.

3.7.3 Glue Records

A records for nameservers are referred to as glue records. Since there might be more than one name server for a domain, the average number of A records for name servers is calculated, based on our distributed crawling results, as a feature. Besides, the maximum TTL value among all the glue records is recorded as a feature as well. Similarly, these numbers could also serve as an implication of the underlying server infrastructure pattern.

The list of features is presented as follows.

- *average_num_A_records_for_name_servers*: Total number of nameservers divide by total number of A records for name servers.
- *max_dns_nsa_ttl*: Maximum TTL value among all glue records.

3.7.4 PTR Records

PTR record is used for reverse DNS lookup, that is, finding the associated domain for a given IP address.

Because the reverse lookup can reveal the hostname information associated with a given IP address, the matching ratio of the original domain name and the domain in the reverse lookup record is calculated. High matching ratio indicates the IP address is dedicated to this domain , which is a credible behavior, and thus indicates a legitimate domain. Note the highest matching ratio between the target domain and all PTR records crawled from eight vantage points is kept as a feature. The matching ratio here is calculated in the same way as described in Section 3.1.2.

Beyond the matching ratio, whether there exists at least one corresponding PTR record to the IP addresses in all A records is stored as a binary feature.

The list of PTR features is presented as follows.

- *reverse_dns_look_up_matching*: The highest matching ratio of all the reverse lookup domains in the PTR records and the target domain name.

- *exist_PTR_record*: Is there at least one PTR record exists for the all the IP addresses of a domain.

3.7.5 AAAA Records

AAAA records specify the mapping relationship between a domain and its IPv6[52] addresses. Thus it could be an indication of the willingness for a domain to adopt modern or next-generation Internet architecture. A domain that is willing to embrace modern designs to ensure long-term benefits is intuitively less likely to be a phishing website that generally serves for a short-term deceptive purpose.

We examine the existence of AAAA records for both the domain itself and its name servers as binary features, which is presented as follows.

- *whether_AAAA_record_exist_for_domain*: whethe an IPv6 address exists for the domain.
- *whether_AAAA_record_exist_for_name_servers*: whether an IPv6 address exists for the domain's name servers.

3.8 HTTP Redirection Features

While both legitimate domains and phishing domains utilize redirection in practice, different purposes and patterns lie behind it. For example, a legitimate domain, say, a.com, could redirect a user who is visiting http://a.com to https://a.com to ensure the security of the communication. On the contrary, a phishing website might ensconce itself behind an online redirection service to avoid being detected, as pointed out by Chhabra et al.[12]. Hence for each URL or domain, we first perform a redirection checking via

Chromium browser[53] running in headless mode driven by Selenium[54], a browser automation framework.

During the redirection, the program remembers all the intermediate URLs passed and the final landing URL, from which the length of the redirection chain and the number of different domains crossed are calculated as features. These two features could be recruited to reflect underlying redirection pattern differences between legitimate domains and phishing domains. For example, a long redirection chain with multiple different domains crossed could be suspicious. We use both redirection features because one could imagine that a legitimate website might be more likely to use redirection within the same domain.

The list of features extracted from redirection is presented as follows.

- *len_redirection_chain*: Length of the redirection chain from the initial URL/domain to the final destination.
- *different_domains_crossed*: The number of different domains crossed during the redirection.

4 System and Experiment Design

In this section, we first introduce the design of the data collection and classifier training system, going through each of the components. Next, we discuss the setup of the experiment, including the data collection process, legitimate and phishing website data source, machine learning algorithms used, and evaluation methods for the classification results.

4.1 Classification System Design

The design of our system for crawling data and generating classifier complies with the following criteria:

(1) The classifier should work with low false positives, since the misclassification of a legitimate website might lead to serious consequences such as lawsuits and financial loss. In a real commercial application of such systems, the administrator might even be willing to sacrifice the true positive rate for a low false positive rate.

(2) The classifier should be resilient to evasion techniques taken by miscreants. This requires the feature set to possess a substantial portion of features which are challenging enough to be forged.

(3) The feature collection system should be flexible regarding changes in the feature set and data collection process. Considering the fast changing nature of phishing websites, the feature set, which plays the most crucial role in classification, might need adjustments from time to time. Hence our system needs to be amenable to changes in an effortless style.

Based on the above objectives, the overall structure and components of the data collection and classifier training system are expressed as Figure 4.1. The detail of each component is discussed sequentially.

4.1.1 Classifier Training Workflow

Our data collection and classifier training system is constituted of the following parts: a set of disjoint feature collection and extraction pipelines, the training/testing data file generator, and the machine learning classifier.

4.1.2 Feature Crawling and Raw Data Processing Pipelines

The very first step of our workflow, given we already have the list of labeled websites, is to collect different sets of raw data corresponding to each group of features. As shown in Figure 4.1, a feature crawling and raw data processing pipeline comprises a Crawler, a Raw Data Database, a Raw Data Processor, and a Feature Database, yielding the set of feature values corresponding to the current group of features for later usage. Here each pipeline could correspond to one or more feature groups, such as those described in

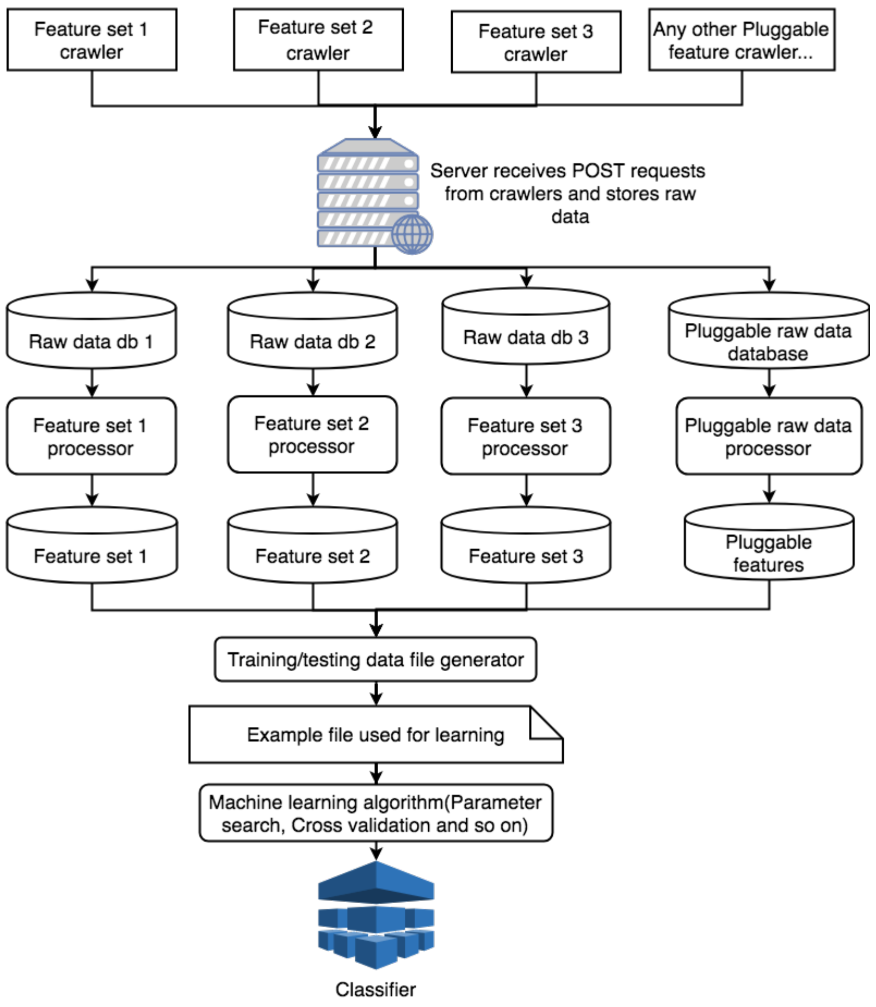

Feature set 1 crawler
Feature set 2 crawler
Feature set 3 crawler
Any other Pluggable feature crawler...
Server receives POST requests from crawlers and stores raw data
Raw data db 1
Raw data db 2
Raw data db 3
Pluggable raw data database
Feature set 1 processor
Feature set 2 processor
Feature set 3 processor
Pluggable raw data processor
Feature set 1
Feature set 2
Feature set 3
Pluggable features
Training/testing data file generator
Example file used for learning
Machine learning algorithm(Parameter search, Cross validation and so on)
Classifier

Figure 4.1. Classifier Training System with Decoupled Components

Chapter 3, to achieve convenient and mutually-independent collecting and parsing processes. The main advantage of this design is low coupling between different pipelines, that is to say, low entanglement exists between the processes for collecting and parsing different groups of features. This design, therefore, enjoys the benefits of a plugin-and-play pattern for any new feature groups to be added in the future. Similarly, removal of old feature groups is as simple as unplugging the relevant pipeline from the system. In our case, we divide the features based on the collection position (i.e., distributed collection or not) and the raw data contents (i.e., certificates, network trace, and others) for ease of deployment, resulting in four sets: certificate-distributed, others-distributed, network-local and others-local.

Given a pipeline for a feature group, it starts with obtaining raw data used for extracting the features. For example, the pipeline of certificate features downloads the X.509 certificates for each website in our labeled website list, and the pipeline of server status features performs server scanning using tools like *nmap*[41] and *traceroute*. The downloaded raw data for each group is stored in separate databases for subsequent parsing.

As the next step, the corresponding processor of each group retrieves records from the database, transforming raw data into the set of feature values that belong to that feature group based on the meaning of features discussed in Chapter 3. Extracted features are persisted in the corresponding database of each group of features.

4.1.3 Training/Testing Data File Generator

The training/testing data file generator assembles features from each of the separate feature database and produces the *csv* file used as machine learning input in which each of the examples is represented as a feature vector.

4.1.4 Machine Learning Classifier

After the feature vectors of URLs/websites are stored in a *csv* file, this file is fed to the machine learning algorithms to learn the classifier used for predicting the label of new URLs/websites, i.e., labeling them as positive or negative. Random Forests and Decision Tree algorithms are applied to the feature vectors in this case. Decision Tree is selected because of its understandability which conduces to the comprehension of the classification process and phishing characteristics, while Random Forests[55] is selected because:

(1) Random Forests algorithm is robust against noise and outliers in the training data.

(2) Random Forests algorithm can be easily parallelized.

(3) Random Forests algorithm gives a useful internal estimate of error and feature importance, which can be used for analysis later on.

Their performances will be discussed and analyzed in detail in Chapter 5.

In this thesis, we used Decision Tree and Random Forests algorithms implemented by scikit-learn[56].

4.1.5 Practical Usage

Our proposed system is naturally suitable for being applied by large-scale service providers such as Google or Bing. Considering they already have well-established distributed infrastructures for Internet-wide data crawling, collecting features should be a relatively easy integration.

Once the classifier is trained using the above-described system, users can query the system for the classification result of a URL to determine its legitimacy.

The query, however, can result in two scenarios. One is that the system has already performed a classification on this queried URL before and the results are preserved in the constructed blacklist. In this case, the result could be returned to the user directly. Another is that the queried URL has never been seen by the system previously. In this case, the system needs to start a new distributed feature collection process for the given URL, producing its feature vector. This feature vector is fed into the machine learning classifier to get a prediction result, which is conserved in the blacklist database for future reference.

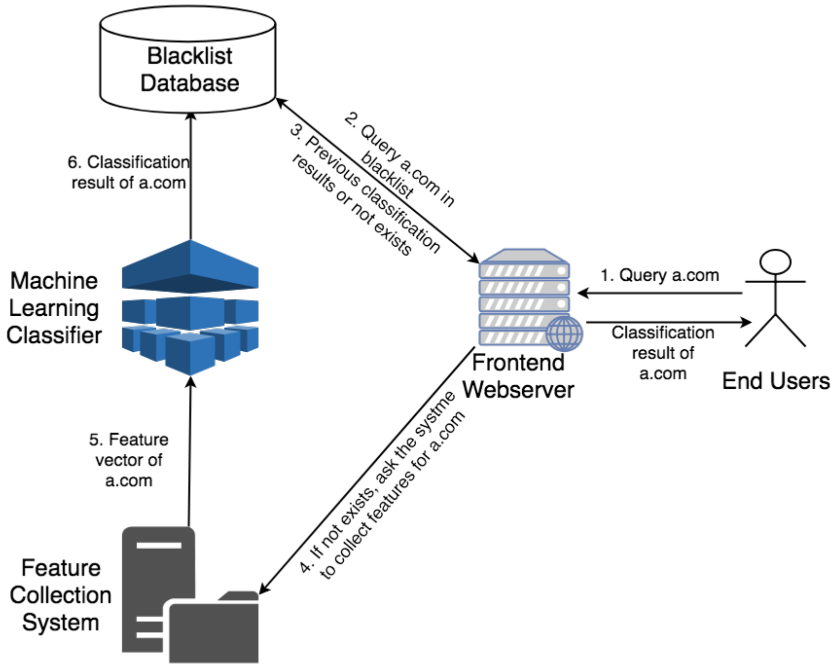

Blacklist Database
2. Query a.com in blacklist
3. Previous classification results or not exists
6. Classification result of a.com
Machine Learning Classifier
1. Query a.com
Classification result of a.com
Frontend Webserver
End Users
5. Feature vector of a.com
4. If not exists, ask the systme to collect features for a.com
Feature Collection System

Figure 4.2. User Query Handling

In the second case, the first user who submitted the initial query for the URL might not be able to get the desired classification result for the URL. However, any user querying for this URL subsequently will benefit from the query of the first user. Thus we argue that this system, once integrated with existing service providers, is advantageous for a majority of the users, protecting them from phishing attacks.

The whole process is presented in Figure 4.2.

4.2 Data Collection

4.2.1 Data Sets

Our phishing list is obtained from PhishTank[13], an open source blacklist which contains phishing URLs that have been manually verified by community members. Our script downloaded the blacklist hourly from April 17th, 2018 until August 15th, 2018.

The benign list comes from a snapshot of the Majestic top one million list[15] on July 16th as this could be a good representative of legitimate websites, considering their relatively higher popularity among all websites. Note that using top ranked websites as benign examples is a common practice in relevant studies[11,18,47,51,57].

For all the URLs, we follow their redirection chain to get the landing URL as the actual target for crawling.

In summary, a total of 1,000,000 legitimate URLs and 100,930 phishing URLs are scanned by our crawler. Out of all the legitimate domains, the crawlers (except for certificate crawler) each successfully downloads data for around 82% to 84% of the legitimate domains, while the intersection of domains existing in each feature database covers 74% of the total legitimate domains crawled. Additionally, the certificate crawler is able to download certificates for 71.3% of legitimate domains, indicating a 71.3% HTTPS coverage among the Majestic top one million domains minimally. On the other hand, out of all the phishing URLs, the crawlers (except for certificate crawler) are able to obtain data for 72% to 78% of the phishing URLs, within which the intersection of all crawled data covering 69% of URLs, while the certificate crawler downloads certificates for 67% of phishing URLs, implying that the potential number of phishing attacks armed with HTTPS certificates have increased drastically since the fourth quarter of 2017, as pointed out by APWG report[7] that around 31% of phishing attacks were hosted on websites that have HTTPS certificates at fourth quarter, 2017. The low coverage of data collected for both classes of URLs could be due to the following reasons:

(1) Data loss happens on the server side. We only have one server dedicated to data storage, receiving all post requests from vantage points around the world and storing the raw data into corresponding databases. Hence the server could

be congested due to too many concurrent requests, leading to random request drops and data loss.

(2) Data loss happens in the vantage points. We utilize EC2 t2.micro[34] instances as distributed vantage points, with one virtual CPU and one GB memory. In order to complete the data collection within a reasonable amount of time, a large number of threads are used concurrently for the crawler, which might occasionally exhaust the system resource. Crawling threads might randomly fail in such cases, leading to the emergence of data loss for this URL.

(3) Data loss because of websites being offline. Since phishing websites are known to be short-living, as described by previous studies[49,58], our crawler is unable to gather data for websites that are offline when crawled.

(4) Data loss because of the limitation of the crawler. The python WHOIS library[59] we used raised an unexpected socket exception which is not caught properly by our code. The uncaught exception will result in the crash of a crawling thread, leading to data loss for a website.

4.2.2 Ground Truth Checking

SafeBrowsing[9] and Virustotal[60] are employed to check the Majestic million lists, further purifying our data sets.

A URL is removed from the legitimate data set if the URL appears in both Majestic top one million list and PhishTank blacklist. In total, 9 URLs are removed from the legitimate set.

Among all legitimate URLs, those that are marked as unsafe by SafeBrowsing are also removed. SafeBrowsing checking against the whole top one million list generates 1,563 URLs that are marked as unsafe even though they do not exist in the PhishTank blacklist.

Due to a relatively low query limitation of Virustotal's academic API (20,000 query per day), we only checked the four legitimate samples used in our experiments out of the whole Majestic million list. The four lists are 100,000 URLs sampled from the whole list, URLs ranked as top 100,000 in the list, URLs ranked as 450,000 to 550,000 in the list, and URLs ranked as bottom 100,000 in the list, which is described in detail in Chapter 5.

4.2.3 Crawler Setting

The data collection process utilizes eight AWS EC2 t2.micro[34] instances located in Ohio, North California, Mumbai, Tokyo, Sydney, London, Paris, and São Paulo. In other words, we have two vantage points in North America, two vantage points in Asia, one vantage point in Australia, two vantage points in Europe and one vantage point in South America. A server on campus is used to collect some of the features that do not require distributed crawling or are too sensitive to collect on commercial servers. All of our servers run Ubuntu 16.04.4 LTS with python 2.7.

The feature groups or features that are collected distributedly on AWS instances include certificate features, geographical locations, average distance measurement, and DNS features. Feature groups or features that are collected from the server on campus include server port status, server OS, HTTP header features, network features, URL lexical features, WHOIS features, and redirection features.

4.3 Machine Learning Algorithms

In this paper, we utilize Decision Tree and Random Forests for classification.

4.3.1 Decision Tree

Scikit-learn implements an optimized version[61] of the CART decision tree algorithm[62]. CART algorithm is similar to the well-known C4.5[63] algorithm. Based on CART decision tree algorithm, each node is constructed by selecting the feature or threshold that generates the most substantial improvement using "Gini measure of impurity"(see Equation 4.1 below). By default, it constructs binary trees only, that is, every parent node is split into two child nodes. The authors of CART argue that binary splits are preferable compared with multiway splits for the following reasons:

(1) The data is fragmented slowly.

(2) Repeated splits on the same attribute are possible. In this case, any attribute can be partitioned for as many times as possible.

The trees generated by CART are grown to a maximal size without using any stopping rules until no further splits are possible. Afterward, the trees are pruned such that the splits contributing the least to the overall performance are removed.

The "Gini measure of impurity" used by CART, given a tree node t, can be calculated using equation 4.1, of which j is the different class labels in the problem and $p(j)$ is the probability of the class j.

$$G(t) = 1 - \sum_{j} p(j)^2 \tag{4.1}$$

Once a parent node P is split into two child node L and R, the improvement, or gain, generated by the split is calculated using equation 4.2.

$$I(P) = G(P) - q \cdot G(L) - (1-q) \cdot G(R) \tag{4.2}$$

Here, q is the portion of instances that goes to the left subtree.

The reason why Gini is favored over entropy, by the authors of CART, is that:

(1) Gini can be computed more quickly.

(2) Using Gini can make the tree to generate "pure nodes", i.e., nodes with only one class label that do not require further split, more likely.

4.3.2 Random Forests

Different from Decision Tree, Random Forests[55] grows an ensemble of asymmetric trees with the purpose of adding randomness to the constructed Decision Trees. This approach improves classification performance considerably in most cases.

A random group of input variables, or features, and training data are selected to grow a series of decision trees. As a result, the best split among the subset of features randomly chosen from all features , rather than the best split among all features, is selected for splitting the node in each tree. The predicted class of a given input is the most popular vote among all trees in the forest. In contrast to the original publication, Random Forests implemented by scikit-learn combines individual classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class[64].

4.4 Evaluation Metrics

Stratified 5-fold cross-validation is applied in order to obtain the average performance of the classification process. Specifically, the labelled data set is randomly split into five subsets such that each subset has the same proportion of each class label as the overall proportion. Then one subset is chosen as the test set while the other four subsets are used for training the classifier. This process repeats for five times (using a different subset as the test set) to evaluate the average performance.

Firstly, the performance of our system is measured using four widely used metrics: precision, recall, accuracy and F-1 score.

Precision is the number of true positives (the examples predicted to be positive by the classifier that are in fact positive) divided by the total number of examples predicted as positive (that is, the sum of true positives and false positives) by the classifier. In other words, it is the percentage of correctly classified positive examples out of the total number of positives predicted by the classifier. Thus, a low precision indicates that the system is generating a large number of false alarms relative to correctly detected actual problematic cases. Precision can be calculated using equation 4.3.

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Recall, or true positive rate (TPR), on the other hands, refers to the percentage of correct positive predictions made by the classifier out of the total number of positive cases in the dataset. It can also be described as the number of true positives divided by the sum of true positives and false negatives. A low recall indicates that a lot of actual problematic cases are erroneously treated as benign by the classifier. Recall can be calculated using equation 4.4.

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

Accuracy is a very basic measure of the performance of a classifier, counting the total number of correctly classified cases out of all classes, which can be calculated using equation 4.5.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

F-1 score seeks to combine Precision and Recall, indicating the ability of the classifier to predict both positive and negative classes. Thus F-1 score is used as the criterion for parameter grid search described in Section 5.1. F-1 score is calculated as shown in equation 4.6

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.6}$$

Secondly, the Receiver Operating Characteristic (ROC) curve regarding “phishing” label is examined. The ROC curve illustrates how the classification ability of the classifier varies while the prediction threshold changes. ROC is generated by plotting the true positive rate (TPR) against false positive rate (FPR) using a series of classification thresholds reflecting various levels of classification sensitivities.

The ROC curved is plotted using TPR as the y-axis and FPR as the x-axis. A perfect classifier can achieve a TPR of 1 with an FPR of 0. While this is normally impossible in a realistic case, good classifiers should obtain a reasonably high TPR with a low FPR, indicating that most of the actual positive cases are correctly predicted with a small number of false alarms raised. As for the plot, this ideal case is implied by having a ROC curve whose Area Under Curve (AUC), that is, the area above the x-axis and under the ROC curve, is close to 1. And a perfect classifier would have the AUC equal to 1.

Hence, the ROC curve could serve as a natural tool for sensitivity/benefit analysis for the classifier. A system administrator, therefore, can use the ROC curve to select an appropriate threshold for the classifier based on the preference for a higher detection of problematic cases or a lower number of false alarms. True positive rate can be calculated using equation 4.4 and false positive rate is calculated using equation 4.7.

$$FPR = \frac{FP}{FP + TN} \tag{4.7}$$

5 Evaluation

In this chapter, we evaluate the performance of the system regarding phishing detection as well as provide insights into understanding the classification process. Stratified 5-fold cross validation is utilized to gather the classification performances. Specifically, the following aspects of the system are examined:

(1) Is our feature set effective for phishing detection and what are the outstanding features for classification?

(2) Can we achieve a good accuracy with low false positive rate?

(3) How does the system perform when classifying websites of different popularity?

(4) Does the proposed methodology works better than an existing study?

5.1 Experiment Setup

To obtain the average performance metrics for our system, stratified 5-fold cross validation is used to train and test our classifier. All legitimate examples in the following experiments are checked against SafeBrowsing[9] and Virustotal[60], removing URLs marked as unsafe by either of them, to further clean up our data set and strengthen our ground truth. Missing values are handled using a naïve strategy, relpacing them as special values such as -1 (for numerical features) or “missing” (for categorical features).

Our experiments are ran under *Ubuntu 16.04* with *python 2.7* and *scikit-learn 0.20*. The default implementation of Decision Tree and Random Forests from scikit-learn is used for the classification.

5.2 Hyperparameter Tuning

The Decision Tree classifier is trained using default setting of scikit-learn, without any hyperparameter tuning process. The hyperparameters of the Random Forests, i.e., *number of trees* and *maximum depth of trees*, are tuned with nested stratified 5-fold cross-validation grid search. The tuning process is performed with using 1 to 100 as the searching range for *maximum depth of trees* and 10 to 400 as the searching range for *number of trees*. A standalone example set containing 2,000 legitimate examples and 2,000 phishing examples that are randomly sampled from the full URL lists are used for the hyperparameter tuning. Those examples are then removed from the whole data set. As a result of the tuning process, for Random Forests classifier, the *number of trees* is set as 271 and the *maximum depth of trees* is set as 37. The hyperparameter tuning process is presented as Algorithm 3. The *GridSearch* function here uses the default implementation from scikit-learn[56].

5.3 Correlation of Features

First of all, the correlation between each pair of features proposed in Chapter 3 is calculated.

Here the widely used Pearson Correlation Coefficient is utilized. The formula used to calculate the Pearson Correlation Coefficient $\rho_{X,Y}$ between two random variables X and Y is presented in Equation 5.1. In the formulation, $\bar{X}$ and $\bar{Y}$ refers to the means of the two random variables.

$$\rho_{X,Y} = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}} \tag{5.1}$$

Note that Pearson Correlation Coefficient is mainly used for revealing linear correlations between random variables. Thus, using Pearson Correlation Coefficient might not be preferable for binary and categorical features (all categorical features are represented as distinct integers before calculating correlation and training/testing machine learning classifiers) when consider interdependency between them and other numerical features. For example, the operating system of the server, which is a categorical

Algorithm 3 Random Forests Hyperparameter Tuning Algorithm

Input: V is the input feature vectors
Output: n is the number of trees, d is the maximum depth of trees

```
1:  procedure NESTED CROSS VALIDATION GRID SEARCH($V$)
2:      $n\_range \leftarrow [1, 400]$
3:      $d\_range \leftarrow [1, 100]$
4:      $folds \leftarrow 5FoldStratifiedCrossValidaion(V)$
5:      $o\_results \leftarrow []$
6:      for each fold in folds do
7:          $i\_folds \leftarrow 5FoldStratifiedCrossValidaion(fold)$
8:          $classifier \leftarrow DefaultRandomForests()$
9:          $i\_results \leftarrow GridSearch(i\_folds, n\_range, d\_range, classifier)$
10:         $o\_results.add(i\_results)$
11:     end for
12:     $o\_results \leftarrow sort(o\_results)$      ▷ Sort the parameters based on performance
13:     $n \leftarrow o\_results[0].n$                ▷ Get the best number of trees
14:     $d \leftarrow o\_results[0].d$                ▷ Get the best depth of trees
15: end procedure
```

feature, might be highly related to TCP TTL value, which is a numerical value, since different operating systems have different initial TTLs. However, the integer-represented value of the operating system is less likely to have a linear correlation with the TCP TTL value, making their correlation coefficient value low. We may explore different correlation measurement techniques for different kind of features in the future work to reveal more potential interdependency between features.

Next, we enumerate some findings regarding the correlated feature pairs.

5.3.1 Case Study: Correlation of *port:443* and *has_certificate*

Not surprisingly, the open/close status of port 443 has a strong positive correlation with the existence of certificate for a domain/URL. The correlation between *port:443* and *has_certificate* is 0.8 based on our dataset. The correlation is less than 1, since some of the servers with an open port 443 do not have proper TLS support and thus fail to provide certificates to our crawler.

5.3.2 Case Study: Correlations of Average Distance Measurements

In this thesis, three novel distance features are proposed for the first time, based on our knowledge. These features reflect average distance to the target server from a set of widely distributed vantage points, measured as the RTT delay, router hop distance, and geographical distance. Thus, investigating the correlation between these features could strengthen the understanding of relationships between RTT, hop distance and geographical distance. The correlations between *average_rtt*, *average_hop_distance* and *average_geo_distance* are presented in Table 5.1.

Features	**average_rtt**	**average_hop_distance**	**average_geo_distnace**
average_rtt	**1**	**0.95**	**0.60**
average_hop_distance	**0.95**	**1**	**0.58**
average_geo_distance	**0.6**	**0.58**	**1**

Table 5.1. Correlationn between Distances

All of them have positive correlations with each other, which is in accordance with the common understanding of the network structure. However, the correlation between *average_rtt* and *average_geo_distance* and the correlation between *average_geo_distance* and *average_hop_distance* are significantly lower than the correlation between *average_rtt* and *average_hop_distance*. This directly indicates that more router hops within a connection means a farther network distance and thus a higher RTT.

5.3.3 Case Study: Correlations of Open Ports

Another set of novel features presented in this thesis are open ports of the server out of the 50 most frequently used ports, and the total number of ports that are open. While *open_ports_count* does not seem to be related to the open/close status of most of the individual ports, high positive correlations are discovered between *open_ports_count* and a set of email-related ports, which is presented in Table 5.2.

Ports	**correlation with open_ports_count**	**protocol**
port:25	**0.80**	**SMTP**[65]
port:110	**0.82**	**POP3**[66]
port:143	**0.82**	**IMAP**[67]
port:465	**0.77**	**SMTP-SSL**[68]
port:587	**0.77**	**SMTP**[69]
port:993	**0.81**	**IMAP-SSL**[68]
port:995	**0.85**	**POP3-SSL**[68]

Table 5.2. Correlationn between Email Ports and Number of Opened Ports

All of the ports that have high correlations with *open_ports_count* are related to email protocols. Meanwhile, those ports share high positive pairwise correlations as well, shown in Table 5.3.

Ports	**port:25**	**port:110**	**port:143**	**port:465**	**port:587**	**port:993**	**port:995**
port:25	**1**	**0.85**	**0.85**	**0.83**	**0.87**	**0.84**	**0.86**
port:110	**0.85**	**1**	**0.98**	**0.87**	**0.86**	**0.96**	**0.95**
port:143	**0.85**	**0.98**	**1**	**0.87**	**0.86**	**0.97**	**0.94**
port:465	**0.83**	**0.87**	**0.87**	**1**	**0.86**	**0.89**	**0.87**
port:587	**0.87**	**0.86**	**0.86**	**0.86**	**1**	**0.87**	**0.85**
port:993	**0.84**	**0.96**	**0.97**	**0.89**	**0.87**	**1**	**0.97**
port:995	**0.86**	**0.95**	**0.94**	**0.87**	**0.85**	**0.97**	**1**

Table 5.3. Correlationn between Invididual Email Ports

The cumulative percentage of servers with a given number of opened email related ports are presented in Figure 5.1, in which the proportion of servers with at most *x* number of ports opened out of all ports showed in Table 5.2 is plotted.

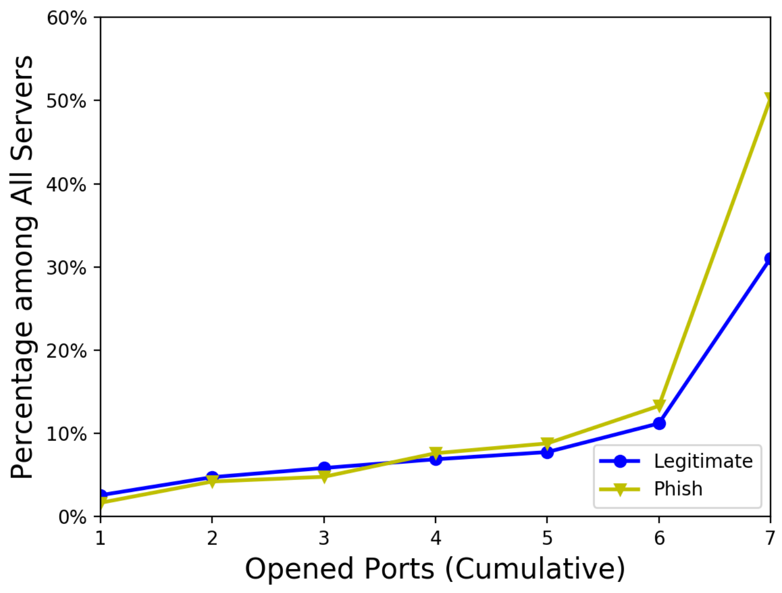


Figure 5.1. Cumulative Percentage of Servers with Number of Opened Ports

As the figure shows, the portion of servers with all email related ports opened out of all the legitimate servers scanned is 21%, while this portion is 37% for phishing servers. Furthermore, the percentage of servers with 6 or less of the above-listed ports opened within the legitimate set is almost the same with phishing set, with a value around 10%, indicating that both sets of servers tend to either open or not open all of them. However, phishing servers are more likely to open all the email related ports. This observation is consistent with our previous assumption that legitimate domain administrators tend to exploit servers for dedicated missions.

The high correlation between email-related ports and *open_ports_count* further implies that email servers tend to open more ports, possibly in order to account for different kinds of email protocols.

To better understand the relationship between each of the individual email port and the class label, the percentage of servers with each port opened grouped by class labels is exhibited in Figure 5.2. According to the figure, servers of phishing domains are around 15% to 20% more likely to have each of the email ports open compared to servers hosting legitimate domains. In addition, among all the ports, 110, 143, 993 and 995 are the four ports that have the largest percentage-wise differences between the two classes, indicated by the *Difference* bar in the chart. This observation further explains the reason why the Random Forests classifier assign port 110, 143, 993 and 995 with higher feature importance than any other ports within the top 50 commonly used ports we scanned.

5.3.4 Correlation Distribution

To provide a global overview of the pair-wise correlations between the features, the CDF of the correlation coefficient values among all feature pairs, along with a histogram showing the number of feature pairs in different correlation ranges, are presented in Figure 5.3a. Based on the figure, while almost all feature pairs express positive correlations, a significant portion of feature pairs has a correlation around 0. While lack of linear correlation does not entail feature independence, this distribution of correlation coefficients provides little indication of obvious feature redundancy. There exist, however, some features with correlation even higher than 0.9, which is worthwhile to scrutinize.

A heat map for the correlations of features is presented in Figure 5.3b, in which the features are numbered in the order they appear in Chapter 3.

Features with extremely high correlations chiefly locate at the upper left corner of the map, around the index 0 to 15, which represents correlations between certificate extensions. The reason why certificate extensions are highly correlated with each other could be concluded as:

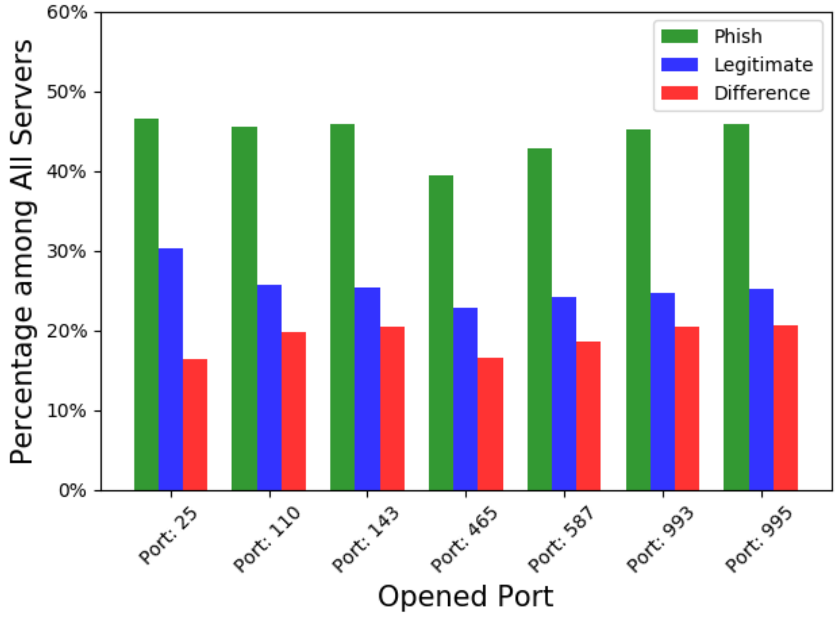
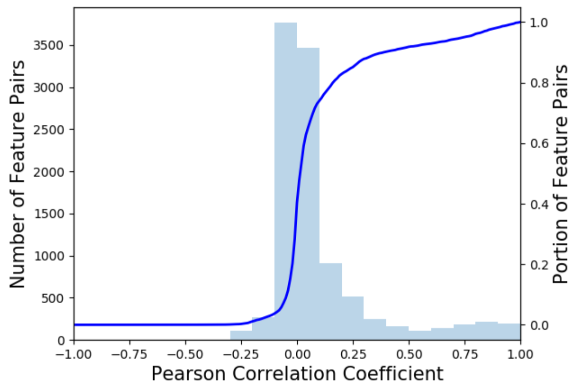


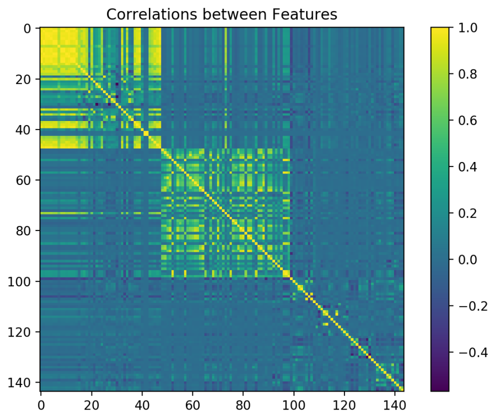
Figure 5.2. Per Port Percentage of Servers

(1) Certificate extensions cooperate with each other in practice to deliver relevant information. For example, *Key Usage* defines how the certificate might be used for, and *Certificate Policies* provides the legal rules associated with the certificate. Intuitively they could be interrelated, and a high correlation is factually found between the two extensions.

(2) Values within certificate extension features can only contain -1, 0 and 1 (in which -1 means certificate is not downloaded for this URL, 0 means this extension does not exist in the associated certificate of the URL, and 1 means this extension exists in the certificate). Some of the extensions that are not used frequently by domain administrators mainly appear as 0 (this extension does not exist in the certificate downloaded) in our feature vectors. Features with most of the values being 0 will therefore appear as having high correlation. This situation requires further attention and will be addressed in our future work on feature selection.



(a) CDF and Counts for Pair-wise Correlations



(b) Heat Map for Pair-wise Correlations

Figure 5.3. Correlation Distribution

Besides, relatively high correlations are detected between certificate extensions and features around 40, which includes matching ratios of *Issuer and Subject Information* and *Other Certificate Features*. This reveals there is some internal interdependency within certificate extensions, certificate issuer and certificate version.

Similarly, region around row 40, column 40 also exhibits intermittently located high temperatures, indicating potential patterns exist in Issuer sub-fields, serial number, signature algorithm and certificate issuer.

Also, high correlations emerge between feature with index 75, which is the open/close status of port 443, and almost all certificate features, reflected as the yellow line at row 75 and column 75 in the heat map.

Additionally, moderately high correlations indwell sparsely at features around 50 to 100. All of them are features for open/close status of ports. High correlation for ports reflects the fact that some ports are opened together to deliver specific services, as in the email ports case discussed in Section 5.3.3.

5.4 Full Feature Set Evaluation

In this section, we evaluate the performance of our system based on the the feature set presented in Chapter 3. Decision Tree and Random Forests algorithm described in Section 4.3 are leveraged for the classification purpose. Both classifiers are trained using all features.

100,000 legitimate domains sampled randomly from Majestic top one million and 97,430 phishing URLs obtained from PhishTank are used for generating the example set used for training and testing. All the sampled legitimate domains are checked against

SafeBrowsing and Virustotal, within which SafeBrowsing reports 158 of them as unsafe, while Virustotal finds 696 of them risky. 94 URLs are reported as malicious by both of them. All the domains that are marked as unsafe by either of them are removed. As a result, 73,654 legitimate examples and 64,657 phishing examples are generated, consonant with the percentage described in Section 4.2.1.

5.4.1 Classification Results

Decision Tree and Random Forests are then evaluated against this example set using stratified 5-fold cross validation. Note that the hyperparameters of the Random Forests, in this case, the number of trees and the depth of trees, are learned from a standalone example set containing 4,000 examples as described in Section 5.1.

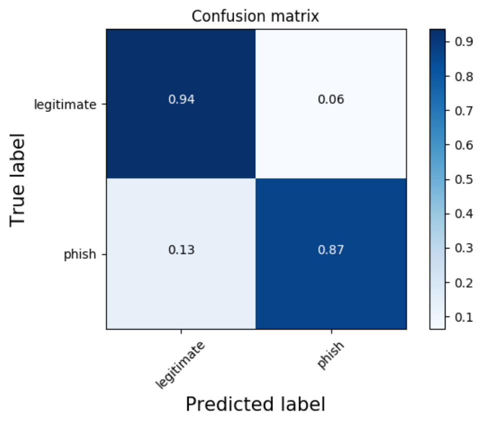
Commonly used performance metrics, including precision, recall, accuracy and F1-score, of both classifiers are presented in Table 5.4. According to the table, Random Forests classifier outperforms the Decision Tree classifier significantly. High F1-scores of Random Forests classifier imply that the detection power of it is balanced between the two class labels, showing that the feature set can be effectively used for phishing detection. Specifically, Random Forests have impressive precision and accuracy of 0.97 and 0.96, respectively. That means, out of all the URLs predicted as positive/phishing by the classifier, 97% of them are actually phishing URLs. And out of all the test cases we have, 96% of them are correctly labeled.

Meanwhile, to understand the classifiers in depth regarding their performance against each label, the confusion matrices of both classifiers are presented in Figure 5.4. Based on the Figure, Random Forests classifier enjoys the benefits of both high TP rate and TN rate with low FP rate and FN rate, which can be further tuned in accordance with the

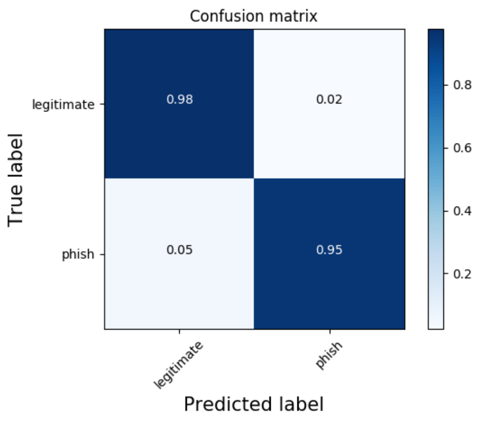
Metrics	**Decision Tree**	**Random Forests**
Precision	**0.92**	**0.97**
Recall	**0.87**	**0.95**
Accuracy	**0.90**	**0.96**
F1-score	**0.89**	**0.96**

Table 5.4. Classification Performance

utilization scenario. Note that even though the Random Forests classifier is not tuned specifically in accordance to reduce false positives, our classifier can still reach a FP rate as low as 0.02, which is a desirable behavior. Moreover, the performance of Random Forests is superior to the Decision Tree in every aspect of the confusion matrix, indicating its higher suitability in the current phishing detection problem, despite its low understandability.



(a) Confusion Matrix of Decision Tree



(b) Confusion Matrix of Random Forests

Figure 5.4. Confusion Matrix of Decision Tree and Random Forests

Another critical performance characteristic for machine learning classifiers is the ROC curve. And the area under the ROC curve, known as AUC, is a direct quality metric of this characteristic. ROC and AUC of both classifiers are presented Figure 5.5. According to the figure, the Random Forests classifier accomplishes an average AUC (across the five folds) of 0.99, which is close to the ideal case. Furthermore, the ROC curves of all folds gathered via stratified 5-fold cross validation are almost identical to each other, indicating that the classifier's performance is stable.

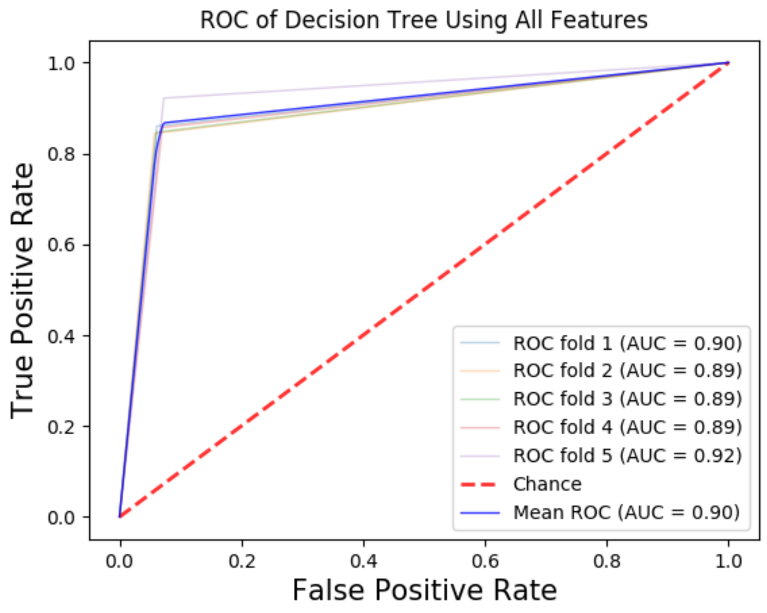
On the other hand, the Decision Tree classifier achieves inferior average AUC and per-fold AUCs, which is around 0.90. The variance of the per-fold ROC curve is also higher than the Random Forests classifier.

In conclusion, Random Forests classifier exhibits great performance against our data set, which strengthens the analysis in Chapter 3, proving the effectiveness of the methodology and feature set presented in this thesis.

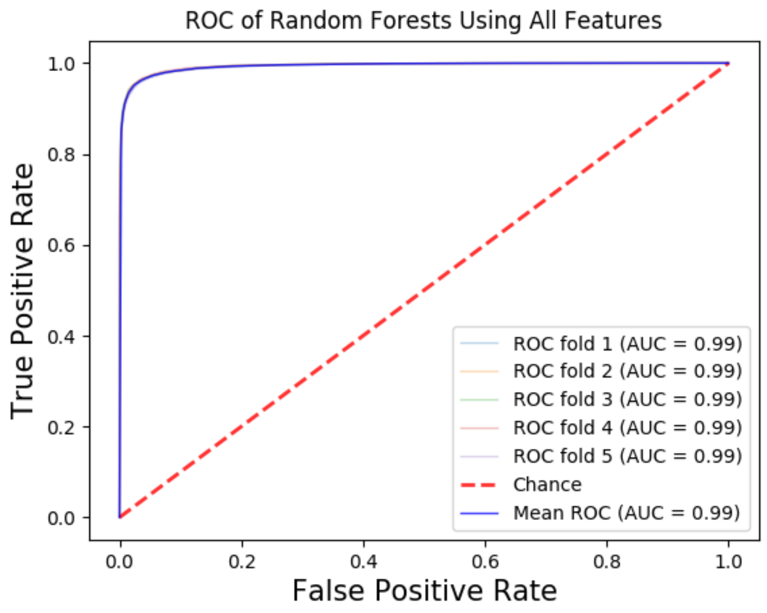
5.4.2 Feature Analysis

To understand the classification process, a Decision Tree is trained and printed using all examples in the data set used in this section. Here we enumerate the top few features selected by Decision Tree to inspect the underlying differences between phishing and legitimate URLs. The otherwise hard to understand decision boundaries behind the classification process can also be revealed through this analysis.

The top three levels of the tree constructed by the Decision Tree algorithm are presented in Figure 5.6. Inside each node, the first line indicates the split rule of the node, the second line shows the Gini impurity (Gini impurity of a node is the probability of an inorrect classification of an example if the example was randomly assigned a class label with probability equal to the fraction of the examples with this label, out of all the examples under the given node) of the node, the third line displays the total number of examples in the node, the fourth line means the respective number of examples for each label in the node, in which the left-hand side number represents the number of legitimate examples, and the fifth line indicates the majority class of the node.



(a) ROC curve of Decision Tree



(b) ROC curve of Random Forests

Figure 5.5. ROC curve of Decision Tree and Random Forests

According to the figure, the very first feature selected by the Decision Tree classifier is *http_header_count*, which is a novel feature proposed in this thesis. The classifier splits a large portion of the phishing URLs to the left-hand side branch based on *http_header_count<=6.5*. That is to say, a big fraction of phishing websites, in fact, has a smaller number of HTTP headers than legitimate ones. To further understand this issue, the cumulative distribution of *http_header_count* is presented in Figure 5.7. According to the figure, almost half of the phishing websites does not have any HTTP header associated. This phenomenon could be due to that those phishing websites do not support HTTP *HEAD* method, which is used by our crawler to download HTTP headers.

In the second level of the decision tree, *match_website_altname* and *num_of_dots* are selected for the partition.

Firstly, *match_website_altname<=0.714* is used to split a majority of phishing examples to the left branch. Note that this is one of the features newly proposed in the thesis as well. As described in Section 3.1, *match_website_altname* is a direct measure of credibility of the certificate linked with current website.

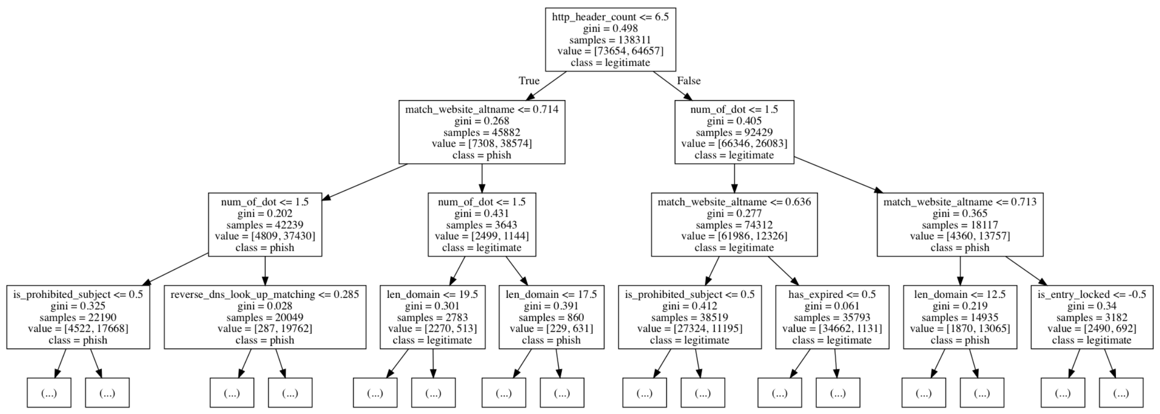


Figure 5.6. Decision Tree Structure (Top 3 Levels)

Consequently, to discover the statistical heterogeneity of *match_website_altname* between phishing websites and legitimate websites, the CDF of this feature against both classes is presented in Figure 5.8.

Based on the figure, around 90% of the phishing examples whose certificate is successfully collected have a *match_website_altname* less than 0.714. while only around 30% of legitimate examples have associated certificates with the matching ratio less than 0.714. The distribution difference expounds the reason behind the split. Furthermore, this confirms our intuition to include *match_website_altname* as a feature. The matching ratio here is calculated using the default string sequence matching function provided by *difflib* of python 2.7[40]. Potentially, better performance could be achieved if a more complicated matching algorithm is used.

The other split rule at this level is *num_of_dot<=1.5*, with a majority of legitimate examples being allocated to the left branch (the number of dots is less than or equal to

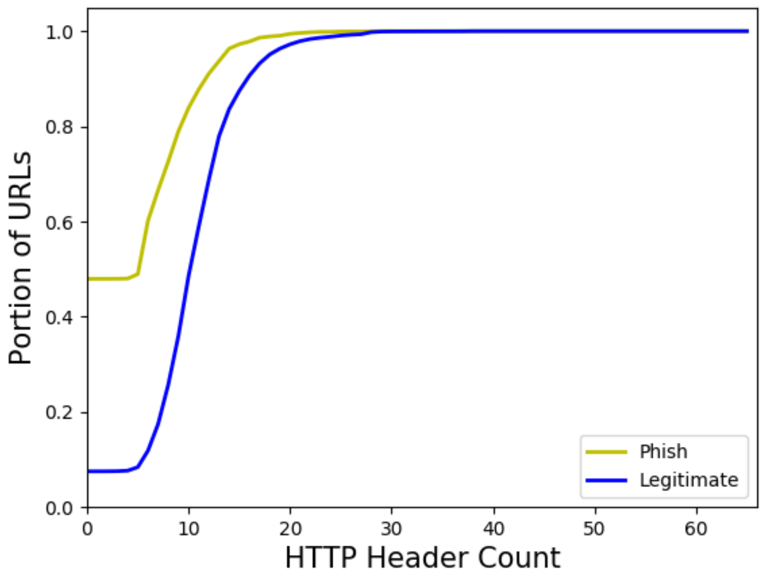


Figure 5.7. CDF of HTTP header count

1.5). This is confirmed by the overall CDF of the number of dots within the domain, as presented in Figure 5.9. Based on the figure, the number of dots in legitimate domains are in general less than the number of dots in phishing domains. The number of dots in the domain is a feature leveraged by several previous studies[19,48,70].

In conclusion, two out of three features selected at the top two level of the decision tree are new features proposed in this thesis, indicating the importance of the novel features proposed in this thesis.

5.5 Evaluation against Websites with Different Ranks

To evaluate the performance of the classifiers against legitimate websites with different ranks, we used the websites ranked top 100,000, 450,001 to 550,000 and 900,001 to

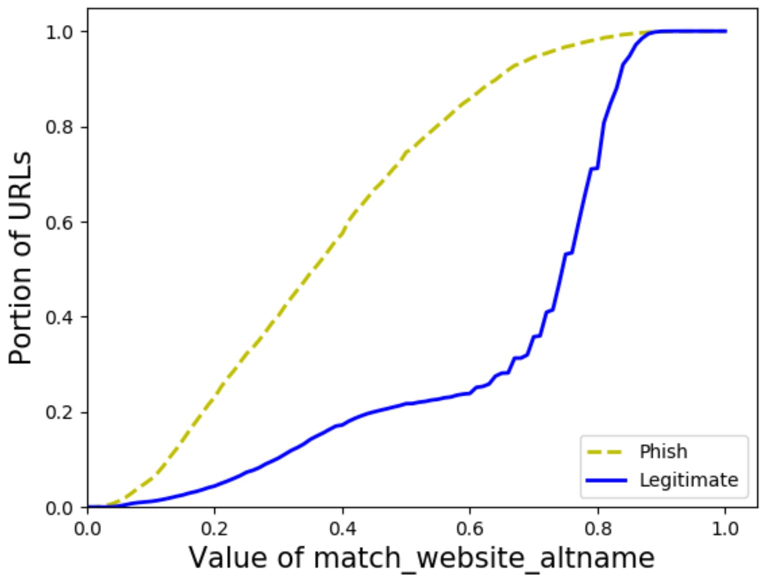


Figure 5.8. CDF of Matching Ratio between Website and SubjectAltnativeName

1,000,000 from the Majestic million list as test sets. Within all three data sets, those domains that have appeared in the data set used in the previous section, as well as those used for tuning hyperparameters, are removed. Intuitively, discrimination between legitimate and phishing websites should be easier for more popular websites. A good classifier should be able to not only detect phishing websites but also avoid falsely identifying regular websites as phishing ones.

The three sampled sets from Majestic million are all checked against Virustotal and SafeBrowsing. Out of the top 100,000 websites, Virustotal recognizes 560 as malicious, while SafeBrowsing reports 128 as malicious. Out of the websites ranked 450,000 to 550,000, Virustotal reports 537 as malicious, and SafeBrowsing states 159 as malicious. Out of the bottom 100,000 websites, Virustotal treats 346 as malicious. Meanwhile, SafeBrowsing treats 159 as malicious. Any URL that is reported as malicious by either of these two sources is removed from our legitimate sets. All the results show low overlappings between the SafeBrowsing check and Virustotal Check. This indicates two conclusions:

(1) SafeBrowsing utilizes a different classification strategy from Virustotal.

(2) Virustotal's checking is more sensitive than SafeBrowsing.

Those three sets are then classified through the Random Forests classifier trained using the whole data set used in the previous section to evaluate the prediction performance against websites with different ranks. The classification results are presented in Table 5.5. Since all of the examples used for testing in this experiments are negative examples (legitimate examples), only TN rate (which is also Accuracy in this case) and FP rate are presented in the table. According to the table, legitimate websites with three different rank range are all classified with high TN rate, which demonstrates the ability of the classifier that is trained on a relatively small sample to treat legitimate websites with different popularities justly, further indicating that the proposed methodology has a high discrimination power.

Ranks	**TN rate/Accuracy**	**FP rate**
1 to 100,000	**0.98**	**0.02**
450,001 to 550,000	**0.98**	**0.02**
900,001 to 1,000,000	**0.98**	**0.02**

Table 5.5. Classification Results for Domains with Different Ranks

5.6 Comparison with an Existing Study

Since there are several existing studies[18–21] related to phishing detection with machine learning that claims to have good performance, we are interested in comparing our methodology with previous studies. A previous study published by Dong et al.[18] that is repeatable based on our current data set is selected. We compare their classification results with ours. The evaluation is conducted based on the same data sets used in Section 5.4.

5.6.1 Full Data Set Evaluation

Dong et al. proposed a phishing classification system that solely relies on certificate features. We evaluate their methodology as follows.

First of all, all URLs within the example set that have no certificate associated with them are removed. The resulting examples are then used to train a Random Forests classifier with their feature set. Stratified 5-fold cross validation is used for generating average metrics for the classifier.

Since the hyperparameters used by their algorithm are not given out in the paper, the hyperparameters used by the Random Forests are tuned by performing a grid search on the standalone example set mentioned in Section 5.2, within which URLs without certificates are removed.

The classification results are presented in Table 5.6. According to the table, Random Forests classifier trained using feature set from Dong et al. cannot achieve the performance claimed in their paper, which is 0.955 for Accuracy and 0.937 for Recall. Meanwhile, our classifier is around 4% to 6% better on all quality metrics.

Metrics	**Random Forests**
Precision	**0.91**
Recall	**0.91**
Accuracy	**0.92**
F1-score	**0.91**

Table 5.6. Classification Results using Dong et al.'s Features

The ROC curve and the confusion matrix of the Random Forests trained using their feature set are presented in Figure 5.10a and Figure 5.10b, respectively. It is evident that their ROC curve is worse than ours, with a smaller AUC. Meanwhile, the classifier trained using features presented in our paper has higher TP and TN rate, and lower FP rate and FN rate.

5.6.2 Evaluation against Websites with Different Ranks

Similar to Section 5.5, we are also interested in the performance of their methodology against the legitimate website with different popularity ranks.

As their classification only uses certificate features, all the URLs without certificates are removed from the example set, analogous to the previous subsection. The performance, including TN rate and FP rate, is presented in Table 5.7. According to the table, the Random Forests classifier trained using their proposed feature set has a 0.94 TN rate on the top 100,000 websites, while it is worse by 0.02 on the bottom 100,000 websites. Our classifier, on the other hand, shows almost identical classification performance against the top 100,000 websites and the bottom 100,000 websites, indicating that our classifier can be used to classify websites with different popularity ranks with confidently. The significantly higher variance of the performance against legitimate websites of different ranks when compared with ours indicates their methodology does not generalize as well to websites with different popularity ranks.

In conclusion, their proposed method cannot reach the alleged performance. The performance decline of their methodology could be due to the following reasons.

- They only use legitimate websites ranked top 100,000 as their non-phishing examples, which are conceivably easier to classify than lower ranked ones.
- The ecosystem of phishing URLs has changed since their experiment. Specifically, APWG phishing activity trends report[7] points out that the percentage of phishing sites which provide valid HTTPS certificates has increased from less than 5% in 2016 to around 31% in 4th quarter of 2017, and our data collection even suggests that certificates could be downloaded for more than 60% of the phishing URLs, making the boundary between legitimate and phishing websites blurry when it comes to the certificate.

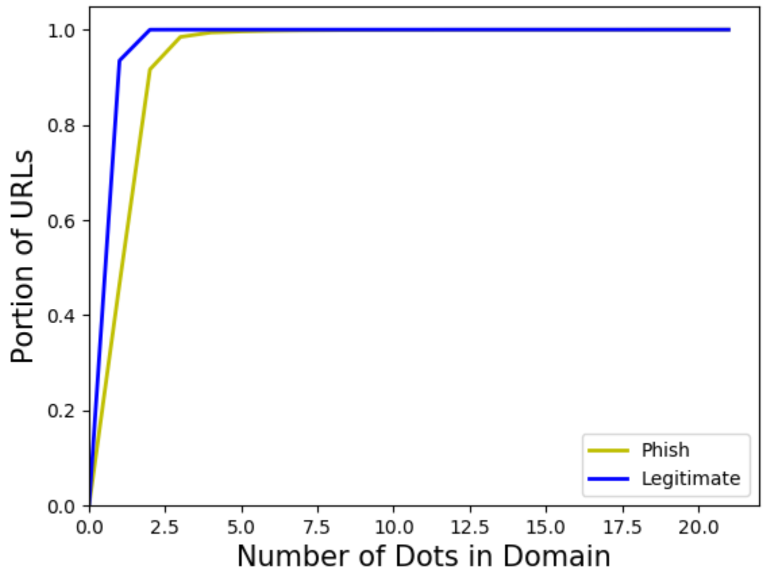
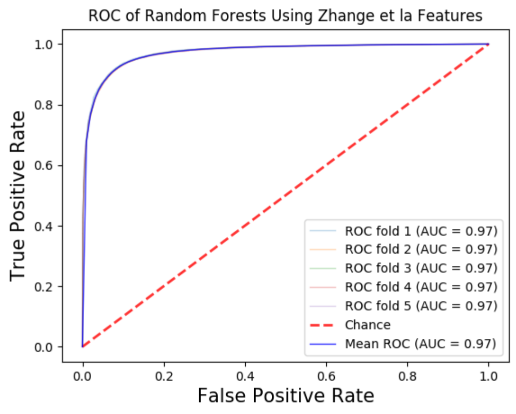
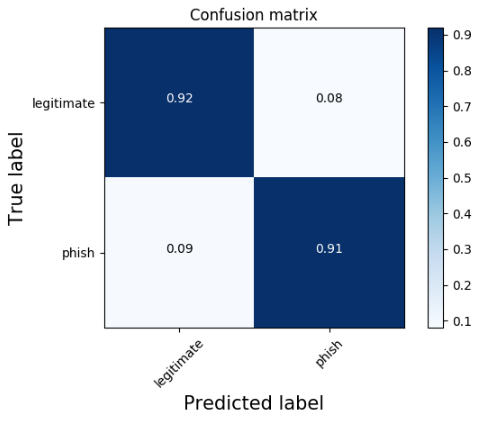


Figure 5.9. CDF of the Number of Dots in the Domain



(a) ROC Curve



(b) Confusion Matrix

Figure 5.10. Random Forests Trained using Features from Dong et al.

Ranks	**TN rate/Accuracy**	**FP rate**
1 to 100,000	**0.94**	**0.06**
450,001 to 550,000	**0.93**	**0.07**
900,001 to 1,000,000	**0.92**	**0.08**

Table 5.7. Classification Results for Domains with Different Ranks Using Dong et al.'s Features

6 Discussions and Conclusions

In this thesis, an automatic phishing detection system is presented. The system utilizes a rich set of features, including hard-to-forge features, captured from many different aspects of the corresponding URLs, as well as from distributed vantage points. With using the Majestic top one million list as legitimate URLs and blacklists downloaded hourly from PhishTank as phishing URLs, data collection and machine learning experiments are conducted to evaluate the proposed methodology. In addition, a previous study is analyzed based on our contemporary data set.

Experiment results show that our approach achieves good accuracy for phishing detection, indicating the effectiveness of the proposed mechanism. The previous study that is analyzed, on the other hand, exposes a performance decline due to the evolution of the phishing ecosystem, while our proposed methodology and feature set demonstrates significant superiority. Meanwhile, differences between legitimate and phishing websites are revealed based on the case study of some features.

Nevertheless, there are some limitations in our study as well. First of all, the legitimate and phishing examples used as ground truth in the experiments are downloaded from single sources, i.e., Majestic top one million list and phishtank.com. Further evaluation of the mechanism against different sources of URLs is needed. Besides, as is described in Section 4.2.1, our data collection process fails to gather information for some of the URLs we have, due to the reasons mentioned. Based on our best knowledge, no bias is found within the dataset because of data loss. However, further evaluation of the mechanism with more complete data set is preferable.

Besides, there are more than 140 features used by the machine learning process currently. Further data analysis focused on understanding the classification process and reducing the total number of features used would be preferable. For example, answering the question, what is the smallest possible set of features that could lead to the same level of performance, would be interesting. This could significantly reduce the effort needed for data collection. Nonetheless, decreasing the total number of features has the side effect of facilitating attackers by reducing the effort needed to deceive the system.

It is worthwhile to point out that the battle between attackers and guardians of the cyber-security is endless. Even though our proposed features present high classification power against state-of-the-art phishing practice, they could be stale and ineffective someday as a result of the evolution of phishing ecosystem. Hence constant analysis and update of the features is required to maintain the high detection power of the system.

Appendix A

Abbreviations

PKI: Public Key Infrastructure

CA: Certificate Authority

SSL: Secure Socket Layer

TLS: Transport Layer Security

URL: Uniform Resource Locator

HTML: Hypertext Markup Language

DOM: Document Object Model

HTTPS: Hyper Text Transfer Protocol Secure

OS: Operating System

RTT: Round Trip Time

TTL: Time to Live

TP: True Positive

FP: False Positive

TN: True Negative

FN: False Negative

ROC curve: Receiver Operating Characteristic curve

AUC: Area Under Curve

CDF: Cumulative Distribution Function

Complete References

[1] https://tools.ietf.org/html/rfc5280.

[2] https://tools.ietf.org/html/rfc5246.

[3] https://tools.ietf.org/html/rfc2818.

[4] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. Analysis of the https certificate ecosystem. In Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13, pages 291–304, New York, NY, USA, 2013. ACM.

[5] 2013 microsoft computing safety index (mcsi) worldwide results summary.

[6] https://apwg.org/.

[7] http://docs.apwg.org/reports/apwg_trends_report_q4_2017.pdf.

[8] A quarter of phishing attacks are now hosted on https domains: Why?

[9] https://safebrowsing.google.com/.

[10] https://en.wikipedia.org/wiki/microsoft_smartscreen.

[11] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean-Michel Picod, and Elie Bursztein. Cloak of visibility: detecting when machines browse a different web. In Security and Privacy (SP), 2016 IEEE Symposium on, pages 743–758. IEEE, 2016.

[12] Sidharth Chhabra, Anupama Aggarwal, Fabrício Benevenuto, and Ponnurangam Kumaraguru. Phi.sh/$ocial: the phishing landscape through short urls. In CEAS, 2011.

[13] https://www.phishtank.com/index.php.

[14] https://www.opendns.com/.

[15] https://majestic.com/reports/majestic-million.

[16] Yoshiro Fukushima, Yoshiaki Hori, and Kouichi Sakurai. Proactive blacklisting for malicious web sites by reputation evaluation based on domain and ip address registration. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on, pages 352–361. IEEE, 2011.

[17] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. LEET, 10:6–6, 2010.

[18] Zheng Dong, Apu Kapadia, Jim Blythe, and L Jean Camp. Beyond the lock icon: real-time detection of phishing websites using public key certificates. In Electronic Crime Research (eCrime), 2015 APWG Symposium on, pages 1–12. IEEE, 2015.

[19] Guang Xiang, Jason Hong, Carolyn P. Rose, and Lorrie Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. ACM Trans. Inf. Syst. Secur., 14(2):21:1–21:28, September 2011.

[20] Angelo PE Rosiello, Engin Kirda, Fabrizio Ferrandi, et al. A layout-similarity-based approach for detecting phishing pages. In Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on, pages 454–463. IEEE, 2007.

[21] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In KDD, 2009.

[22] Tu Ouyang, Soumya Ray, Mark Allman, and Michael Rabinovich. A large-scale empirical analysis of email spam detection through network characteristics in a standalone enterprise. Computer Networks, 59:101–121, 2014.

[23] Ram Basnet, Srinivas Mukkamala, and Andrew H Sung. Detection of phishing attacks: A machine learning approach. In Soft Computing Applications in Industry, pages 373–383. Springer, 2008.

[24] Alexander Moshchuk, Tanya Bragin, Steven D Gribble, and Henry M Levy. A crawler-based study of spyware in the web.

[25] Mahmoud T Qassrawi and Hongli Zhang. Detecting malicious web servers with honeyclients. Journal of Networks, 6(1):145, 2011.

[26] Andreas Dewald, Thorsten Holz, and Felix C Freiling. Adsandbox: Sandboxing javascript to fight malicious websites. In Proceedings of the 2010 ACM Symposium on Applied Computing, pages 1859–1864. ACM, 2010.

[27] Kevin Zhijie Chen, Guofei Gu, Jianwei Zhuge, Jose Nazario, and Xinhui Han. Webpatrol: Automated collection and replay of web-based malware scenarios. In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pages 186–195. ACM, 2011.

[28] Bassam Sayed, Issa Traoré, and Amany Abdelhalim. Detection and mitigation of malicious javascript using information flow control. In Privacy, Security and Trust

(PST), 2014 Twelfth Annual International Conference on, pages 264–273. IEEE, 2014.

[29] http://www.malwaredomainlist.com/mdl.php.

[30] https://aws.amazon.com/cn/alexa-top-sites/.

[31] Sean Ford, Marco Cova, Christopher Kruegel, and Giovanni Vigna. Analyzing and detecting malicious flash advertisements. In 2009 Annual Computer Security Applications Conference, pages 363–372. IEEE, 2009.

[32] David Y. Wang, Matthew Der, Mohammad Karami, Lawrence Saul, Damon McCoy, Stefan Savage, and Geoffrey M. Voelker. Search + seizure: The effectiveness of interventions on seo campaigns. In Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, pages 359–372, New York, NY, USA, 2014. ACM.

[33] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. In Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, pages 373–380, New York, NY, USA, 2014. ACM.

[34] https://aws.amazon.com/ec2/.

[35] https://www.mozilla.org/en-us/about/governance/policies/security-group/certs/.

[36] https://en.wikipedia.org/wiki/diginotar.

[37] https://en.wikipedia.org/wiki/comodo_group.

[38] https://blogs.technet.microsoft.com/pki/2014/02/21/a-novel-method-in-ie11-for-dealing-with-fraudulent-digital-certificates/.

[39] https://tools.ietf.org/html/rfc6125.

[40] https://docs.python.org/2/library/difflib.html#difflib.sequencematcher.

[41] https://nmap.org/.

[42] Tu Ouyang, Soumya Ray, Michael Rabinovich, and Mark Allman. Can network characteristics detect spam effectively in a stand-alone enterprise? In International Conference on Passive and Active Network Measurement, pages 92–101. Springer, 2011.

[43] https://dev.maxmind.com/geoip/geoip2/geolite2/.

[44] https://www.maxmind.com/en/geoip-demo.

[45] https://en.wikipedia.org/wiki/great-circle_distance.

[46] https://www.iana.org/assignments/message-headers/message-headers.xhtml.

[47] Joby James, L Sandhya, and Ciza Thomas. Detection of phishing urls using machine learning techniques. In Control Communication and Computing (ICCC), 2013 International Conference on, pages 304–309. IEEE, 2013.

[48] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In Proceedings of the 16th international conference on World Wide Web, pages 649–656. ACM, 2007.

[49] D Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. LEET, 8:4, 2008.

[50] https://whois.icann.org/en.

[51] Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In Proceedings of the 20th international conference on World wide web, pages 197–206. ACM, 2011.

[52] https://www.ietf.org/rfc/rfc2460.txt.

[53] https://www.chromium.org/home.

[54] https://www.seleniumhq.org/.

[55] Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[56] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. Journal of machine learning research, 12(Oct):2825–2830, 2011.

[57] Abubakr Sirageldin, Baharum B Baharudin, and Low Tang Jung. Malicious web page detection: A machine learning approach. In Advances in Computer Science and its Applications, pages 217–224. Springer, 2014.

[58] Qian Cui, Guy-Vincent Jourdan, Gregor V Bochmann, Russell Couturier, and Iosif-Viorel Onut. Tracking phishing attacks over time. In Proceedings of the 26th

International Conference on World Wide Web, pages 667–676. International World Wide Web Conferences Steering Committee, 2017.

[59] https://github.com/joepie91/python-whois.

[60] https://www.virustotal.com/.

[61] http://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart.

[62] Leo Breiman. Classification and regression trees. Routledge, 2017.

[63] J Ross Quinlan. C4. 5: programs for machine learning. Elsevier, 2014.

[64] http://scikit-learn.org/stable/modules/ensemble.html#random-forests.

[65] https://tools.ietf.org/html/rfc5321.

[66] https://tools.ietf.org/html/rfc1939.

[67] https://tools.ietf.org/html/rfc2060.

[68] https://tools.ietf.org/html/rfc8314.

[69] https://tools.ietf.org/html/rfc2476.

[70] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Learning to detect malicious urls. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3):30, 2011.