THESIS

# UNBIASED PHISHING DETECTION USING DOMAIN NAME BASED FEATURES

Submitted by

Hossein Shirazi

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2018

## ABSTRACT

## UNBIASED PHISHING DETECTION USING DOMAIN NAME BASED FEATURES

Internet users are coming under a barrage of phishing attacks of increasing frequency and sophistication. While these attacks have been remarkably resilient against the vast range of defenses proposed by academia, industry, and research organizations, machine learning approaches appear to be a promising one in distinguishing between phishing and legitimate websites. There are three main concerns with existing machine learning approaches for phishing detection. The first concern is there is neither a framework, preferably open-source, for extracting feature and keeping the dataset updated nor an updated dataset of phishing and legitimate website. The second concern is the large number of features used and the lack of validating arguments for the choice of the features selected to train the machine learning classifier. The last concern relates to the type of datasets used in the literature that seems to be inadvertently biased with respect to the features based on URL or content.

In this thesis, we describe the implementation of our open-source and extensible framework to extract features and create up-to-date phishing dataset. With having this framework, named Fresh-Phish, we implemented 29 different features that we used to detect whether a given website is legitimate or phishing. We used 26 features that were reported in related work and added 3 new features and created a dataset of 6,000 websites with these features of which 3,000 were malicious and 3,000 were genuine and tested our approach. Using 6 different classifiers we achieved the accuracy of 93% which is a reasonable high in this field.

To address the second and third concerns, we put forward the intuition that the domain name of phishing websites is the tell-tale sign of phishing and holds the key to successful phishing detection. We focus on this aspect of phishing websites and design features that explore the relationship of the domain name to the key elements of the website. Our work differs from existing state-of-the-art as our feature set ensures that there is minimal or no bias with respect to a dataset. Our learning model trains with only seven features and achieves a true positive rate of $98\%$ and a classification accuracy of 97%, on sample dataset. Compared to the state-of-the-art work, our per data instance processing and classification is 4 times faster for legitimate websites and 10 times faster for phishing websites. Importantly, we demonstrate the shortcomings of using features based on URLs as they are likely to be biased towards dataset collection and usage. We show the robustness of our learning algorithm by testing our classifiers on unknown live phishing URLs and achieve a higher detection accuracy of $99.7\%$ compared to the earlier known best result of $95\%$ detection rate.

ACKNOWLEDGEMENTS

Immeasurable appreciation and deepest gratitude to my advisor Prof. Indrakshi Ray, for her help and support and words of encouragement.

My sincere thanks must also go to all of dbsec group members specially Prof. Indrajit Ray.

Special thank to Dr. Bruhadeshwar Bezawada for his comments, effort in checking and editing this thesis.

I would specially like to thank my wife, Hajar Homayouni who has been extremely supportive of me throughout this entire process as well as she was a graduate student. She made countless sacrifices to help me.

TABLE OF CONTENTS

LIST OF TABLES

# LIST OF FIGURES

# Chapter 1

# Introduction

Phishing, defined as, *the attempt to obtain sensitive information such as user-names, passwords, and credit card details, often for malicious reasons, by masquerading as a trustworthy entity in an electronic communication* [3], is a problem that is as old as the Internet itself. Trying to get unsuspecting users to give up their money, credentials or privacy is a particularly insidious form of social engineering that can have disastrous affects on people's lives. Often this type of attack arrives in the form of an email containing the first part of what Chaudhry *et al.* [4] describe as the *lure*, the *hook* and the *catch*.

The *lure* is what entices the user to click on a link. It can be advertising a way to get easy money, obtain an illicit product, or a warning that a user's account has been compromised or blocked in some fashion. The *hook* is often a website that is designed to mimic a legitimate website of a reputable organization such as a bank or other financial institution. The *hook* is used to trick the user into entering and submitting their credentials such as user-name, password, credit card number, etc. The *catch* is when the user has submitted private information and the malicious owner of the website collects and uses this information to exploit the user and his account.

Phishing attacks continue to be of persistent and critical concern to users, online businesses, and financial institutions. A phishing website lures users into divulging their sensitive information such as passwords, pin numbers, personal information, and credit card numbers, and uses such information for financial gains. According to current industry estimates, the annual financial losses due to phishing attacks across different economies surpasses \$3 billion. These losses are cumulatively borne by both the users and the online businesses, which are targeted by the phishing attacks. Especially, for online users, a phishing attack can mean a lot more than just financial losses as the loss of sensitive personal information has long-term future ramifications as well.

The major problem in detecting phishing attacks is the adaptive nature of strategies used by the phishers. Generating a phishing website has not only become trivial, but also the attackers are able

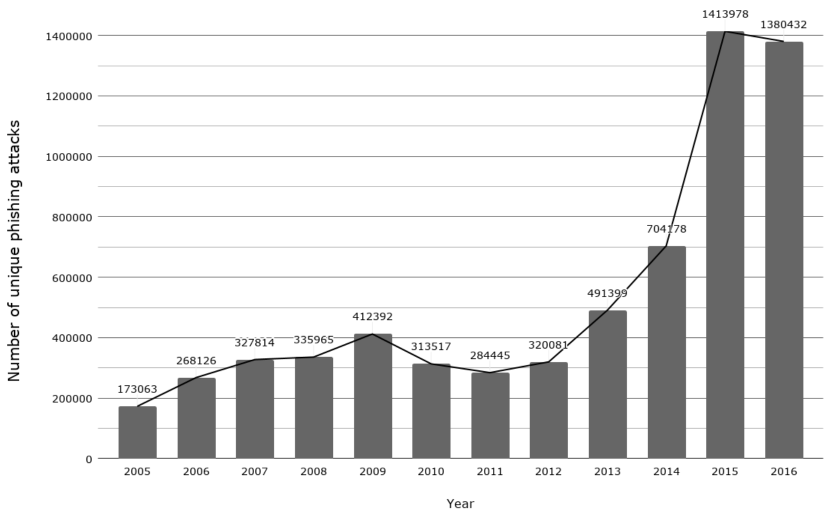


**Figure 1.1:** Number of unique attacks reported by APWG in year from 2005 to 2016

to bypass most defense strategies with relative ease. For instance, the evolution of *extreme phishing*, a complex form of phishing that targets the identity of users shows the severity and intensity of phishing attacks. Phishers are constantly improving phishing toolkits to generate websites that can evade nearly all forms of defenses available. Therefore, there is a need for developing phishing detection approaches that demonstrate robustness and resiliency against the adaptive strategies being used by the phishers.

## 1.1 The Number of Attacks Over Time

Figure 1.1 shows the number of unique attacks reported by Anti-Phishing Working Group (APWG) in each year. It shows that the number of attacks has steadily increased but it is obvious that in 2015, the number of these attacks has doubled with more than 1.4 millions ones.

Figure 1.2 is obtained from the report of APWG for the year 2016 [2], shows that there exist 195,475 unique phishing domain names used to attack targets during 2016, the most that have been recorded in any year since APWG started to work. Also, APWG found that from 195,475 used domains, 95,424 domain names have been registered maliciously by phishers. This is an all-time high, and almost three times as many as the number that APWG found in 2015.

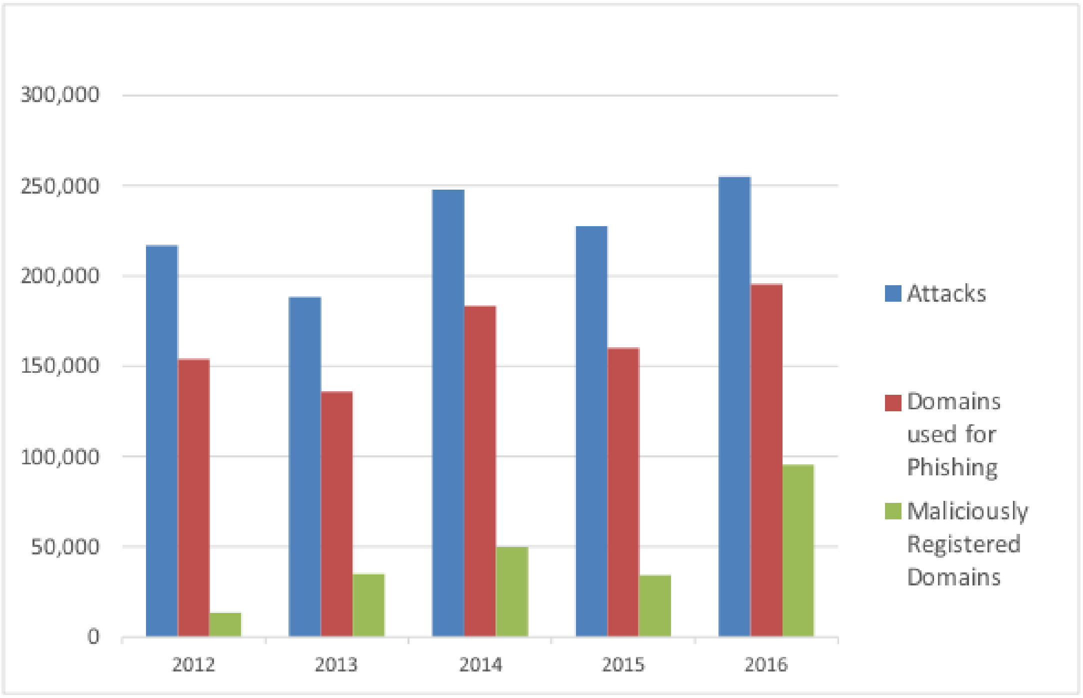


**Figure 1.2:** Phishing attacks and domains used from 2012 to 2016 [2]

## 1.2 Adversaries Motivation

In the last decade, phishing techniques have advanced. Phishers now have new motivation for conducting attacks. Table 1.1 shows the evolution of phishing attacks during the last decade and the important phishing related milestone at each year. While the term phishing was first used in 1996, in the next two decades, this kind of attacks targeted individuals, financial institutes, and military organizations.

These days, attackers are less curious about the security of systems but they try to have some financial benefits. Figure 1.3 shows that 30% of all reported attack was targeted at e-Commerce and Software/SaaS and 25% at banks and financial institutes. Among 1.3 million complaints received by the Federal Trade Commission in 2009, identity theft ranked first and accounted for 21% of the complaints costing consumers over 1.7 billion US dollars [5].

**Table 1.1:** Evolution of phishing during 1996 - 2017 [1]

| Year | Unique Attacks | Important Events |
|---|---|---|
| 1996 | | Term "phishing" was first used |
| 1997 | | Media declared the evolution of a new attack called "phishing" |
| 1998 | | Attackers started using message and newsgroups |
| 1999 | | Use of mass mailing to escalate the phishing attacks |
| 2000 | | First use of key loggers and phishers used it for getting login credentials |
| 2001 | | The first known direct attempt against a payment system affected E-gold, which was followed up by a "post-9/11 id check" shortly after the September 11 attacks |
| 2002 | | Use of screen loggers |
| 2003 | | Use of IM and IRC and the first known phishing attack against a retail bank was reported |
| 2004 | | Phishing is recognized as a fully organized part of the black market |
| 2005 | 173,063 | Term "spear phishing" was first used |
| 2006 | 268,126 | First phishing over VoIP |
| 2007 | 327,814 | More than 3 billion dollar lost to phishing scams |
| 2008 | 335,965 | Cryptocurrencies such as Bitcoin, introduced in late 2008, facilitate the sale of malicious software, making transactions secure and anonymous. |
| 2009 | 412,392 | Symantec Hosted Services blocked phishing attacks impersonating 1079 different organizations |
| 2010 | 313,517 | Facebook attracted more phishing attacks than Google and IRS |
| 2011 | 284,445 | 110 million customer and credit card records were stolen from Target customers, through a phished subcontractor |
| 2012 | 320,081 | 6 million unique malware samples were identified |
| 2013 | 491,399 | Red October operation attacked more than 69 countries |
| 2014 | 704,178 | 750,000 malicious emails were sent using IoT devices reached its peak |
| 2015 | 1,413,978 | Fancy Bear used a zero-day exploit of Java and launching attacks on the White House and NATO |
| 2016 | 1,380,432 | Fancy Bear carried out spear phishing attacks on email addresses associated with the Democratic National Committee (DNC). |
| 2017 | N/A | 76% of organizations experienced phishing attacks. Nearly half of information security professionals surveyed said that the rate of attacks increased from 2016. |

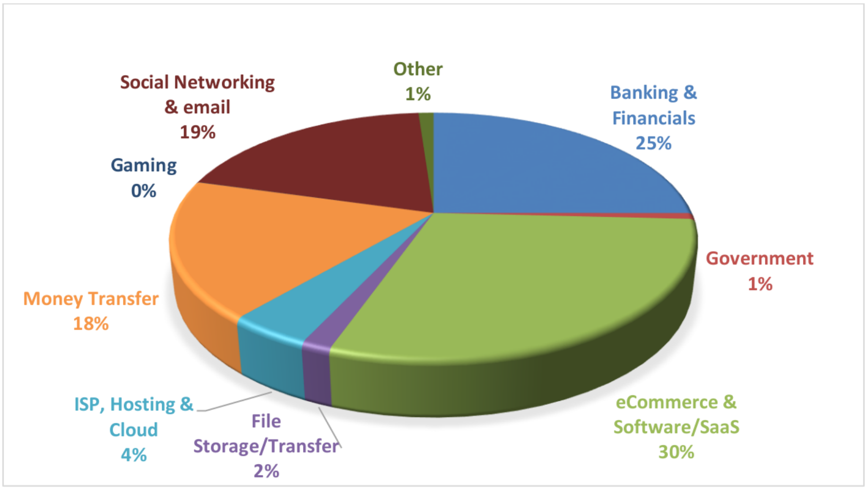


**Figure 1.3:** Phishing attacks by industry in 2016 [2]

## 1.3 Problem Statement

We focus on the problem of determining if a target website is a phishing one or not, based on the information provided on the website. We consider the standard definitions of a phishing website from literature [6, 7]. Typically, the content of a phishing website is textually and visually similar to some legitimate website. We focus on characterizing the nature of such websites using only the information from the website and training a machine learning classifier to distinguish between phishing and legitimate websites.

## 1.4 Limitations of Past Work

The state-of-the-art machine learning approaches for phishing detection can be broadly classified as email, content, and URL based.

The email-based approaches [8–10] focus on analyzing emails based on various features. However, there has been the considerable evolution of phishing emails against such approaches, which makes them inadequate for current day context. This is shown by the relatively high rate of success of *spear phishing* emails attacks [6] compared to other phishing methods.

The content-based approaches [11–20] perform in-depth analysis of content and build classifiers to detect phishing websites. These works rely on features extracted from the page content as well as from third-party services like search engines, and DNS servers. However, these approaches are not efficient due to a large number of training features and the dependence on third-party servers. Using third-party servers violates user privacy by revealing the user's browsing history. More importantly, several features used in these approaches, like URL related features, do not accurately model the phishing phenomenon. Furthermore, in most of these approaches, except [18], there is a critical issue of using biased datasets (see Section 1.5 for a detailed discussion) and the design of features that seem to work well for such datasets. Approaches like [17] examine DOM content of the pages looking for the similarity of attacks. But, with the advent of newer attacks like [21, 22] that closely mimic legitimate websites, such approaches will be ineffective.

The URL-based approaches [23–26] analyze various features based on the target URL such as length of the URL, page rank of the URL, number of dots in the URL, presence of special characters, hostname features like IP address, domain age, DNS properties, and geographic properties, among other features. While the intuition in these approaches is sound, *i.e.*, the URL is a good indicator of phishing attacks, the structural changes of modern-day URLs negates several lexical features identified by these approaches. For instance, these days, the URLs generated by websites like Google and Amazon, are long and contain many non-alphabetic characters, which dilute the lexical similarity of legitimate URLs. For this reason, the URL based approaches inadvertently tend to be biased towards the datasets being used and are likely to be ineffective in the future. A few hybrid detection mechanisms [27, 28] combine content and URL features, but suffer from the same problems as described. The work in [27] also discusses features based on the Fully Qualified Domain Name (FQDN) of the phishing website. However, their approach depends on the results of search engines and incurs a significant delay.

## 1.5 Bias in Datasets

There are two reasons for bias in datasets: *dataset usage* and *URL based features*. First, to create a labeled dataset, many researchers [11, 13, 14, 19, 20, 25] used `Alexa.com` website to create the list of legitimate websites. But, for the phishing dataset, they used anti-phishing sites like

PhishTank.com or Openphish.com. The reasoning is that Alexa.com ranks *domain name* of websites in the Internet and publishes the list of such highly ranked domains and a researcher will be able to generate the dataset based *only* on the first pages of these ranked websites. Whereas the anti-phishing sites list the entire URLs of the phishing web pages. For instance, many phishers use 000webhost.com, a free hosting service to host their phishing sites whereas this domain name itself is ranked highly in Alexa.com. For legitimate websites, the URL is just URL of the first page of the websites while for phishing websites, it is the URL of a specific web-page. As a concrete illustration, for the feature defined as *number of sub domains* in the URL, most of the legitimate website instances obtained from Alexa.com will not have any sub-domains while many of phishing web pages will have sub-domains.

The second reason is that URL based detection [23–26] does not guarantee good distinguishers between legitimate and phishing URLs. This is because adversaries have complete control over the URL, excepting domain name, to obfuscate against any number of measures. For instance, features based on URLs, like the length of the URL, number of subdomains in the URL, number of dots ("."), the presence of special characters, number of suspicious words and so on, are not necessarily unique to phishing URLs. This also explains the reason for the high True Negative Rate (TNR) in existing works.

Except the work by Marchal *et al.* [18] where they used unbiased datasets made available by *Intel Security*, no other work in literature has specifically addressed this concern. However, their approach focused on eliminating bias in datasets and did not explore intuitive feature design. Our work achieves similar classification accuracy with only seven features compared to the 200+ features used in [18]. If the same approaches were to be applied in real-world, these schemes will have lower detection rate, *e.g.* [25].

## 1.6 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents a comprehensive survey of existing detection approaches applied to the problem. In chapter 3 we introduce our initial attempt to solve this problem including discussion about pros and cons of this approach followed by experiments and results in chapter 4. In chapter 5 we discuss our proposed approach and the intuitive behind it. Also, we explain how this approach can be completely useful in fighting against phishing websites. In chapter 6 we discuss our machine learning model that we used in our experiments. We evaluate our proposed approach to conducting different experiments and practical measurement in chapter 7. Finally, chapter 8 concludes the thesis and discusses directions for future work.

# Chapter 2

# Literature Survey

Phishing attacks classify as social engineering attack. In this kind of attack, the adversary does not necessarily look for a vulnerability in the system but also, looks for unaware users to lure them. For example, an attacker creates a webpage similar to a login page of a well-known email provider and sends the link to the users and asks them to log in. In this example, there is not any security concern relates to the Email provider. If the end-user does not aware of the potential threats, they may be fooled by the attacker. During last decade, different researchers tried to come up with different approaches. From a higher perspective, we categorize all of these efforts in two major categories. In the first category, we discuss the approaches that try to address the problem in a human-based manner. The approaches in this category increase the knowledge of end-users and help them to make a good decision when they face a suspicious websites. In the second category, we study the software-based approaches. In this approach, different techniques adapt to distinguish between legitimate websites and phishing ones and takes them down without considering end-users. The result of this category may also be fed to the first category to help end-users. Figure 2.1 shows a classification framework for different existing approaches.

## 2.1 Human Related Approaches

The strategy of phishing attackers is based on taking advantage of unaware or inexperienced users. The users who do not know about these attacks are in more danger.

Knowledge management helps to increase user's information about the attacks and educate them when faced with it.But, the List-based approach shows a warning to prevent the user being fooled by the phishing website.

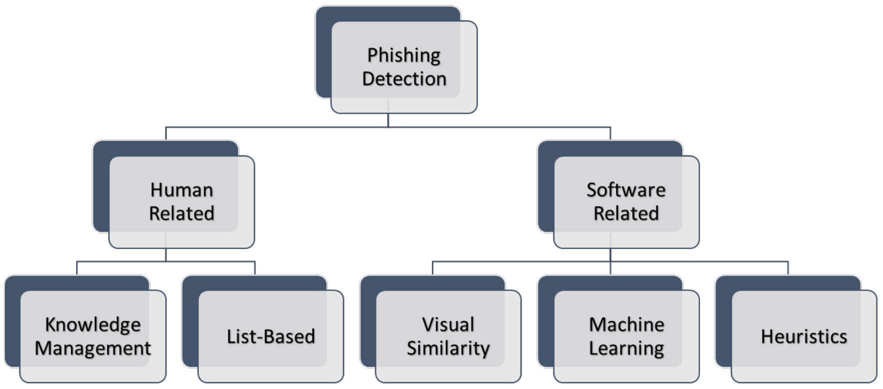


**Figure 2.1:** Classification framework for different phishing detection approaches

### 2.1.1 Knowledge Management and User Educating

The users are the ones who are under risk so it is beneficial to educate them and increase their ability to protect themselves against these attacks.

Matthew L. Jensen *et al.* [29] explored how an organization can utilize its employees to combat phishing attacks collectively through coordinating their activities to create a human firewall. They utilize knowledge management research on knowledge sharing to guide the design of an experiment that explores a central reporting and dissemination platform for phishing attacks. Results demonstrate that knowledge management techniques are transferable to organizational security and that can benefit from insights gained from combating phishing. Specifically, they highlight the need to both publicly acknowledge the contribution to a knowledge management system and provide validation of the contribution by the security team. They reported that doing only one or the other does not improve outcomes for correct phishing reports.

Steve Sheng *et al.* [30] design an online game that teaches users good habits to help them avoid phishing attacks and used learning science principles to design and iteratively refine the game. The participants were tested on their ability to detect phishing website from the legitimate one before and after playing the designed game including playing the game itself and reading an article about phishing. The results show that playing the game can increase the ability to find the phishing website of the participant. Nalin Asanka *et al.* [31] create a mobile version of a game aimed to enhance avoidance behavior through motivation of home computer users to protect against phishing threats [32].

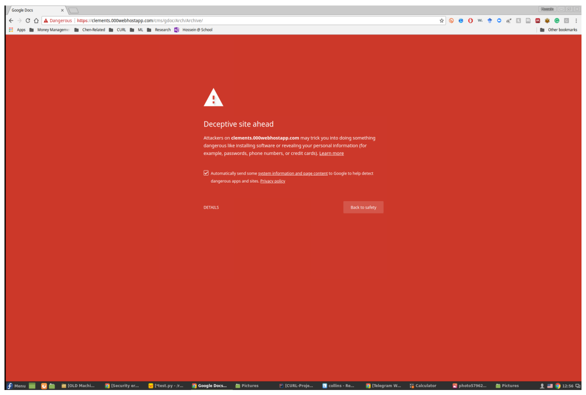
To explore the effectiveness of embedded training, researchers conducted a large-scale experiment that tracked workers' reactions to a series of carefully crafted spear phishing emails and a variety of immediate training and awareness activities [8]. Based on behavioral science findings, the experiment included four different training conditions, each of which used a different type of message framing. The results from three trials showed that framing had no significant effect on the likelihood that a participant would click a subsequent spear phishing email and that many participants either clicked all links or none regardless of whether they received training. The study was unable to determine whether the embedded training materials created framing changes in susceptibility to spear phishing attacks because employees failed to read the training materials

### 2.1.2 List-Based

List-based solutions have the fast access time but they suffer from the low detection rate especially for the zero-day attacks.

Afroz *et al.* [33] build profiles of trusted websites based on fuzzy hashing techniques. This approach combines white-listing with black-listing and heuristic approach to warn users of attacks. Jain *et al.* [34] used an auto-updated white-list of legitimate sites accessed by the individual user. When users try to open a website, which is not available in the white-list, the browser warns users not to disclose their sensitive information. However, all list-based approaches suffer from the problem of dynamic updates and scalability, which makes these approaches impractical for client-side detection.

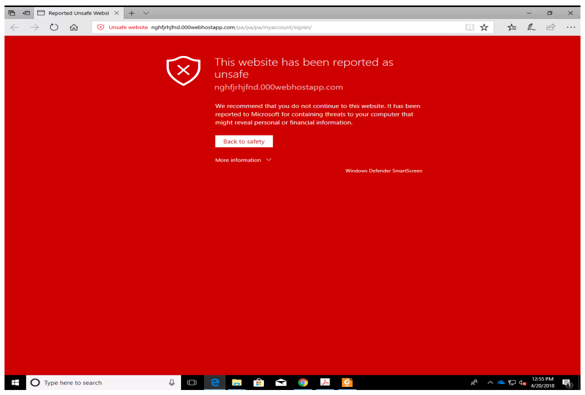
The modern browsers use list-Based approach in an embedded manner and update the list regularly. The browser checks every single website that users want to visit against that list and if the webpage is listed there, give a warning to the user. Figure 2.2 shows an example of that warning shown to the users by Google Chrome and Microsoft Edge.



Deceptive site ahead

Attackers on **clements.000webhostapp.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



This website has been reported as unsafe

nghfjrhjfnd.000webhostapp.com

We recommend that you do not continue to this website. It has been reported to Microsoft for containing threats to your computer that might reveal personal or financial information.

**Figure 2.2:** Warning of phishing attacks in two different browsers; Left: Google Chrome - Right: Microsoft Edge

Firefox checks each website that user visits against reported phishing, unwanted software, and malware lists. These lists are automatically downloaded and updated every 30 minutes by default when the "Phishing and Malware Protection" feature is enabled [35].

Microsoft SmartScreen, used in Windows 10 and both Internet Explorer 11 and Microsoft Edge, helps to defend against phishing by performing reputation checks on visited sites and blocking any sites that are thought to be phishing sites. SmartScreen also helps to defend people against being tricked into installing malicious against phishing attacks. Google's Safe Browsing infrastructure displays warning messages across Chrome, Android, and Gmail if the user tries to access a potentially malicious site or download malware and viruses [36].

## 2.2 Software-Related Approaches

Relying solely on the end-user in the fight against phishing attacks does not lead to beat the adversarial. The end-users are prone to fault in facing with phishing attacks and make an incorrect decision, albeit with vast education and awareness. Addressing this problem needs the help of Software-Related techniques to prevent, detect and take down phishing attacks and campaign. In this section, we will discuss different software-related techniques to fight against them namely Visual and Textual Similarity, Machine Learning, and Heuristics.

### 2.2.1 Machine Learning

Machine learning algorithms have been proven to have the ability to discover complex correlation among different data items of similar nature. Many algorithms consist of two steps: *learning* and *testing*. In the *learning* step, the algorithms try to learn from supporting examples and in the *testing* phase, the researchers evaluate the accuracy of the algorithms.

Attackers often use email to send out phishing URLs to the victim. Consequently, detecting potentially dangerous emails helps lead to prevent users to be caught in the phishing website. There is a wide literature on automating detection for phishing email by looking at the context of the email. For example, Basnet *et al.* [12] used 16 features to detect phishing email. While they use email messages as a source to extract the features, we only focus on the website itself rather than how the attacker tries to tempt the users.

Ma *et al.* [24] described an approach based on URL classification using statistical methods to discover the lexical and host-based properties of malicious web site URLs. They use lexical properties of URLs and registration, hosting, and geographical information of the corresponding hosts to classify malicious web pages at a larger scale. These methods are able to learn highly predictive models by extracting and automatically analyzing tens of thousands of features potentially indicative of suspicious URLs. The resulting classifiers obtain 95-99% accuracy, detecting large numbers of malicious websites by just using their URLs. However, their approach requires a large feature set and extracts host information with the help of third-party servers. In section 1.4 and 1.5 we discussed why using URL-based features and third-party services lead to a biased dataset.

Miyamoto *et al.* [37] provided an overview of nine different machine learning techniques, including Support Vector Machine, Random forests, Neural Networks, AdaBoost, Naive Bayes, and Bayesian Additive Regression Trees. They analyzed the accuracy of each classifier on the CANTINA dataset [11], a state of the art dataset, and achieved a maximum accuracy of 91.34%

using AdaBoost. They used a wide range of classifiers but based on adaptive nature of these attacks and not using an updating dataset cannot guarantee the resiliency of the solution.

Aburrous *et al.* [38] proposed association data mining algorithms to characterize and identify the rules to classify the phishing websites. They implemented six different classification algorithms and techniques to mine the phishing training datasets. They used a phishing case study that was applied to illustrate the website phishing process. The rules generated from their associative classification model showed the relationship between some important characteristics like URL and domain identity, security and encryption criteria, *etc.*. The experimental results demonstrated the feasibility of using Associative Classification techniques in real applications and its better performance as compared to other traditional classifications algorithms.

Xiang *et al.* [14] proposed a layered anti-phishing solution with a rich set of features. They proposed 15 features that exploit the Document Object Model (DOM) of webpages including using search engine capabilities, and third-party services, with machine learning techniques to detect phishing attacks. Also, they designed two filters to help reduce False Positive Rate (FPR) and achieve runtime speedup. The first is a near-duplicate phish detector that uses hashing to catch highly similar phish. The second is a login form filter, which directly classifies webpages with no identified login form as legitimate. The key shortcoming of this approach is that the experiments were conducted with biased datasets. The `Alexa.com` website, which provided most of the legitimate websites in this dataset, only gives the domain name of legitimate websites. While the phishing websites taken from `PhishTank.com` are mostly complete URLs of phishing web pages. So the types of data instances are different. Also, using third-party services to extract some features may endanger the privacy of users by revealing their browsing history. Moreover, this algorithm is not efficient as it uses a large number of features.

In 2015, Verma *et al.* [25] described an approach based on textual similarity and frequency distribution of text characters in URLs. For instance, they examined the character frequencies in phishing URLs and the presence of suspicious words as features. However, this approach is entirely based on URLs and is likely to be biased in the modern day context. Some of their features, like *presence of suspicious words*, will need to be updated frequently as newer phishing attack surfaces come in.

Jain *et al.* [19] described a machine learning based approach that extracts the features from client side only. Their approach examined the various attributes of the phishing and legitimate websites in depth and identified 19 features to distinguish phishing websites from legitimate ones. Their approach has a relatively high accuracy in detection of phishing websites as it achieved 99.39% true positive rate and 99.09% of overall detection accuracy. While their approach relied only on the client-side feature and did not use any third-party features, but there are some drawbacks in this approach. Fox example, their method of dataset creation is flawed. For phishing websites, they used `PhishTank.com`as a source of phishing websites. For legitimate websites, they used mostly `Alexa.com`, which ranks the most top ranked domain names in the world. While `PhishTank.com`generated the phishing pages, `Alexa.com`gives only domain names and not the internal pages of the domain. As a result, their features are biased with respect to the dataset. This factor was not considered in the feature extraction process. For example, one feature in their approach is the number of dots in the given URL. In the training phase, while all given legitimate instances consist of only domain names, the phishing instances consist of entire URLs. Another feature looks for suspicious words in the URL, but many legitimate websites also have these words.

Al-Janabi *et al.* [26] described a supervised machine learning classification model to detect the distribution of malicious content in online social networks (OSNs). Multisource features have been used to detect social network posts that contain malicious URLs. These URLs direct users to websites that contain malicious content, drive-by download attacks, phishing, spam, and scams. For the data collection stage, the Twitter streaming API was used. They just focused only on one OSN network (Twitter) and applied their approach. Their features can neither be extracted locally nor guarantee the security of users outside of the network during regular browsing.

Recently, Marchal *et al.* [16, 18] proposed a client-side detection approach and completed with a browser extension [39] using custom datasets from *Intel Security* and tried to eliminate bias in datasets. They developed a target identification component that can identify the target website that a phishing web page is attempting to mimic. However, their approach uses over $200+$ features for classification, which indicates a significant time for feature extraction and classification. Moreover, not much is known about the exact design of their features and the dataset used is not available to replicate their results.

Rao *et al.* [20] proposed a classification model based on an ensemble of features that are extracted from URL, source code, and third-party services. Their approach is inefficient and suffers from the same problems as other techniques using URL based features. Furthermore, this approach uses third-party servers and reveals a user's browsing history to untrusted servers.

### 2.2.2 Visual and Textual Similarity

Since over 90% of users rely on the website appearance to verify its authenticity [40], the adversaries try to create the visual appearance of phishing websites nearly identical to that of legitimate ones. Consequently, the researchers try to use the similarity between websites as a key feature to discriminate between legitimate and phishing websites. Some approaches use visual similarity between websites while others use textual similarity.

#### Visual Similarity

Chen *et al.* [41] proposed an approach for detecting visual similarity between two web pages. They tested their system using the most popular web pages to examine its real-world applicability. Accuracy in case of true positive and false positive rates reached 100 and 80 percent, respectively.

Fu *et al.* [42] used Earth Mover's Distance (EMD) to measure webpage visual similarity. They first converted the involved web pages into low-resolution images and then used color and coordinate features to represent the image signatures. Then they used EMD to calculate the signature distances of the images of the web pages. They employed an EMD threshold vector for classifying a web page as a phishing or a normal one. Also, they built up a real system which is already used online and it has caught many real phishing cases.

Routhu Srinivasa Rao *et al.* [43] proposed a combination of white-list and visual similarity based techniques. They used computer vision technique called SURF detector to extract discriminative key point features from both suspicious and targeted websites followed by computing similarity degree between the legitimate and suspicious pages.

All these approaches need a target website to compare the similarity between two webpages and detect one of them as phishing.

**Textual Similarity**

Zhang *et al.* [13] created a framework using a Bayesian approach for content-based phishing web page detection. The model takes into account textual and visual contents to measure the similarity between the protected web page and suspicious web pages. A text classifier, an image classifier, and an algorithm fusing the results from classifiers are described. But, this process is expensive and often results in false positives.

Recently, there has been a rise in *extreme* phishing attacks [21, 22], a form of fine-grained content mimicking phishing, on financial institutions where the phishing website mimics the legitimate website to an alarming degree. Typically, these websites are meant to defeat visual and textual similarity analysis. The high level of noise introduced in such websites is likely to defeat most content-based machine learning approaches in the past. Compared to past work, our approach relies on a nominal set of features for classification and does not examine the content of the websites in depth.

### 2.2.3 Heuristic Approaches

Neil *et al.* [44] implemented SpoofGuard, a plugin for Internet Explorer, that detects phishing attempts on the client-side. It weights different anomalies found in the HTML page of websites and assign a score. If the assigned score cross the certain threshold, it will label the website as a phishing and sends a warning to the user. This tool runs on the client-side and can detect phishing websites based on that anomalies in the page but as it is not using

Cui *et al.* [17] tried to find similarities between different attacks during a 10-month study by monitoring around 19000 websites. The study showed that 90% of phishing websites have similar DOM structure and over 90% of these attacks were actually replicas or variations of other attacks in the database.

# Chapter 3

# Initial Attempt: Fresh-Phish Framework

## 3.1 Creating a Phishing Dataset

The dataset based on features of websites on the internet quickly become out of date and stale. We built a framework that can address this problem. Using that, it is possible to add/remove a feature to/from the dataset. In addition, the user can redo the extracting step to get the updated values for currently defined features. In the initial attempt, we use features that Mohammad *et al.* [45] defined, but we implement them in the Python.

To create our dataset, we scanned the top 3000 sites in the `Alexa.com` database and 3000 online phishing websites obtained from `phishtank.com.` We made two assumptions here: first, all of the websites gotten from `Alexa.com` are legitimate websites. We believe this to be a valid assumption because of the ephemeral nature of phishing websites, they tend to pop in and out of existence (as is evidenced by the short domain registration times) to evade being blocked or tagged as phishing. The top sites ranked in `Alexa.com` must be popular and have been around for a longer period of time to attain this ranking. Second, we assumed that websites found on the `Phishtank.com` were phishing websites. `PhishTank.com` incorporates a community of registered users who report sites as phishing. Each member is ranked by the community and builds a good reputation by correctly reporting if a website is phishing or not. Since it is a very well-known repository for phishing websites, we can trust its decision for labeling a website as a phishing one.

## 3.2 Implemented Features

Mohammad *et al.* [45] used 29 different features to create their dataset and we used their definitions to create our own dataset. These features can be categorized into five categories: URL based, DNS based, External statistics, HTML based, and JavaScript based.

### 3.2.1 URL Based

URL based features are based on some aspect of the URL of the website. Attackers try to use the URL to deceive users by obfuscating it in some fashion. For example, URLs that have an IP address, an 'at' symbol (@), double slash, contain a prefix or suffix are all methods employed to disguise a URL. Other notable methods are the length of URL, whether the website has a sub-domain, uses a shortening service or uses a non-standard port.

1. **Having IP Address:** If an IP address is used as an alternative of the domain name in the URL, such as "http://125.98.3.123/fake.html", users can be sure that someone is trying to steal their personal information. In a Python script, we checked that if the website URL is in the form of an IP, we assume it as a phishing website otherwise it is legitimate.

2. **URL Length:** To ensure the accuracy of our study, we calculated the length of URLs in the data set and produced an average URL length. The results showed that if the length of the URL is greater than or equal 54 characters then the URL classified as phishing. By reviewing our dataset, we were able to find 1220 URLs whose lengths equal to 54 or more which constitute 48.8% of the total dataset size.

3. **Shortening Service:** URL shortening is a method on the web in which a URL may be made considerably smaller in length and still lead to the same webpage. This is accomplished by means of an "HTTP Redirect" on a domain name that is short, which links to the webpage that has a long URL. For example, TinyURL is a service that makes the URL shorter. The URL like "http://portal.hud.ac.uk/" can be shortened to "bit.ly/19DXSk4" using this service. If it used TinyURL, we will assume it as a phishing, otherwise, it is a legitimate website.

4. **Having At (@) Symbol:** A URL that contains a "@" symbol is not trusted as the browser generally ignores everything proceeding the "@". If the URL contains the "@" sign we marked it as phishing.

5. **Double Slash Redirecting:** URLs that contain "//" are marked as phishing as the double slash is used to redirect users to another site. Phishing URLs employ this method to hide

their real URL. An example is
`http://www.colostate.edu//http://www.phishing.com`.

6. **Prefix Suffix:** The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example, `http://www.Confirme-paypal.com/`. In our framework, we check whether that website uses a "-" in the name of URL or not. If it is used, we assume it as a phishing website.

7. **Having Subdomain:** Let us assume we have the following link: `http://www.hud.ac.uk/students/`. A domain name might include the country-code top-level domains (ccTLD), which in our example is "UK". The "ac" part is shorthand for "academic", the combined "ac.uk" is called a second-level domain (SLD) and "hud" is the actual name of the domain. To produce a rule for extracting this feature, we first have to omit the (www.) from the URL which is, in fact, a subdomain in itself. Then, we have to remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as "Suspicious" since it has one subdomain. However, if the dots are greater than two, it is classified as "Phishing" since it will have multiple subdomains. Otherwise, if the URL has no subdomains, we will assign "Legitimate" to the feature. We calculated the number of dots in a URL. If it is more than, we classify as phishing otherwise it is a legitimate website.

8. **Unusual Port:** Most legitimate websites use ports 80 for unencrypted traffic and port 443 for encrypted traffic. We mark the sites that use other ports as phishing.

### 3.2.2 DNS Based

DNS based features use information of the domain such as when the domain was first registered and how long the registration is valid.

1. **Domain Update Date:** This feature gets data of "Update Field" from WHOIS. This field demonstrates the latest time that domain owner updated the DNS record on the WHOIS database. The legitimate websites updated their information on the WHOIS database more often than the phishing website. If the updated date is less than half of a year, we mark this site as legitimate.

2. **HTTPS Token:** Phishing URLs will often try to make it look like the URL uses HTTPS. They will include HTTPS as part of the URL, for example, `http://https-colostate.edu.`We mark this URL as phishing.

3. **Age of Domain:** This feature can be extracted from WHOIS database. Most phishing websites live for a short period of time. By reviewing our dataset, we find that the minimum age of the legitimate domain is 6 months. Rule: If the age of domain is greater than 6 months, we will assume it as legitimate otherwise we will assume it as phishing.

4. **DNS Record:** This feature can be extracted from WHOIS database. For phishing sites, either the claimed identity is not recognized by the WHOIS database or the record of the host-name is not founded. If the DNS record is empty or not found then the website is classified as phishing, otherwise, it will classify as legitimate. We implement a Python script which gets DNS information from `www.WHOISXMLAPI.com`and check if the DNS record is empty or not.

### 3.2.3 External Statistics

External statistics based features use data gathered from places like Alexa's page rank and if the site is present in Google's search index.

1. **Page Rank:** This feature looks at if the site is ranked in the Alexa database. If the site is not ranked or has no traffic, then we mark the site as phishing.

2. **Google Index:** This feature examines whether a website is in Google's index or not. When a site is indexed by Google, it is displayed on search results. Usually, phishing web pages

are merely accessible for a short period and as a result, many phishing web pages may not be found on the Google index. To find Google index of each site, we send a request to Google website then search website inside the result. If a website is indexed by Google, we mark it as legitimate. Otherwise, we mark it as phishing.

### 3.2.4 HTML Based

The HTML served by a website contains many valuable features used to determine if the site is phishing or not. Examples of these features include whether the website has a favicon and if the images and JavaScript have the same source URL as the serving website. Other HTML based features are whether the site implements iFrames, how many links point outside the serving domain, etc.

1. **Favicon:** A favicon is a graphic image (icon) associated with a specific webpage. Many existing user agents such as graphical browsers and newsreaders show favicon as a visual reminder of the website identity in the address bar. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a phishing website. For this attribute, we checked the HTML code of each website and found where the Favicon is loading from. If it is loaded from a foreign domain, we assume that website is a phishing.

2. **Request URL:** This feature examines whether the external objects contained within a webpage such as images, videos and sounds are loaded from another domain. In legitimate webpages, the webpage address and most of the objects embedded within the webpage are obtained the same domain. We implemented a Python script which looks at all of the addresses and marks them as domain-inside or domain-outside. If more than half of addresses are domain-outside, we will mark the site as phishing otherwise it is a legitimate one.

3. **URL of Anchor:** This feature looks at the links in the website. If the links in the website point to a domain different from the domain of the website more than 50% of the time, then the site is marked as phishing.

4. **Links in Tags:** This feature looks at the domain in the tags of the header such as <SCRIPT>, <META>, and <LINK> tags. If more than 50% of these tags point to a domain different from that of the site, the site is marked as phishing.

5. **Submit Form Handler:** This feature examines the action of the submit form on the page. If the action is, "None", "blank", or "about:blank", then we mark the site as phishing. Legitimate sites will point to a URL.

6. **Redirect Page:** If the site uses the HTML 301 redirect in the header, then we mark the site as phishing.

7. **Using iFrame:** HTML used the <IFRAME> tag to display another page inside of the current page. This feature looks at if there is an <IFRAME> tag in the page and its border is set to transparent. If these two things are present, then we mark the website as phishing.

8. **Links Pointing to Page:** This feature looks at how many links from other websites are pointing the target site. If there are no links to the target page, then it is marked as phishing. We did not implement this and assigned the score of this feature as neutral.

### 3.2.5 JavaScript Based

JavaScript-based features look for specific ways that JavaScript can be used to trick the end user. Some of these include using JavaScript to submit form data to email, mouse over techniques that hide URLs or prevent right clicks and pop-up windows.

1. **Submitting to Email:** This feature looks for a "mailto:" action in the submit form. If it exists, then we mark the site as phishing.

2. **On Mouse Over:** This method looks for the on mouse over re-writing of links in the status bar. This type of ruse has become less effective as browsers usually ignore this. We used the Python library Dryscrape to run a headless instance of web-kit. This allows us to run and evaluate JavaScript linked or embedded in the page. If the window.status JavaScript call exists in conjunction with onMouseOver then this site is marked as phishing.

3. **Right Click:** This feature looks for JavaScript code that disables the right-click action on a web page. This is meant to deter users from looking at the HTML source code for the site. It looks specifically for "event.button==2" in the JavaScript. If that presents, we mark the site as phishing.

4. **Pop-up Window:** This method uses Dryscrape which implements web-kit and can scrape a web page for JavaScript has well as HTML. JavaScript has alert, confirm, prompt, window.open methods if any of these are found then the site is marked as phishing.

## 3.3 Modifying Features Definition

Mohammad *et al.* [45] defined all features of dataset with binary values: $-1$ as legitimate, and $1$ as legitimate. Since many of the features like the length of URL or age of the domain cannot be defined using binary values, they used a threshold to convert non-binary values to binary ones. This approach has several problems. First, defining a threshold and converting continuous values to binary ones, a lot of useful information that can help classifiers to make a better decision is missed. Second, determining the accuracy and efficacy of the threshold and fine tuning it periodically is an important issue that must be addressed. The choice of the correct threshold becomes very critical as it is the threshold which decides whether a feature is phishy or not instead of the classifier so we decided to remove the threshold and use the actual values of the features that we have obtained from [45].

## 3.4 Added Feature: Most Important Words

In this section, we explain TF-IDF algorithm and how it works and how we used it to make a new effective feature.

### 3.4.1 TF-IDF Algorithm

TF-IDF, a well-known technique in information retrieval, is a numerical statistic intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, text mining, and user modeling [46].

In this technique, the algorithm tries to find out most important words in a text. The *term frequency* (TF) shows the number of appearances of a word in a text and the term *inverse document frequency* (IDF) says the number of appearances of the same word in the entire corpus. For example, if a word is repeated in a text 10 times more than its regular frequency in the corpus, it will get a higher weight in comparison with a word that has the same frequency in both the given text and corpus. The algorithm will give high weights to the words with high repetition in a document and low repetition in the entire corpus and give low weights to the words with low repetition in the document and high in the corpus. By sorting all the weights, it is possible to find out the most important words in the given text.

### 3.4.2 Used Corpus

These algorithms need a corpus with which to compare the given text. For this purpose, we used Manually Annotated Sub-Corpus $MASC$ from Open American National Corpus $OANC$ [47]. It includes approximately 500,000 words of contemporary American English written and spoken data. While this is an annotated dataset, we just used the plain data to have a normal distribution of English words in our corpus.

### 3.4.3 Adapting TF-IDF

For any given website in the dataset, we first prepared a plain-text version of it which the users see it in the browser since the phishers try to deceive users by the plain-text version of the website. In the plain-text version, there are no HTML tags or scripts. Then we gave this version to the TF-IDF algorithms and calculated the weight of each word in it and sorted all the words.

We select the 5 most important words on the website and we put them against the domain name of each website. We then do a search on these 5 words. If the website appears in the response to the search, we mark this feature as $-1$ or legitimate. otherwise, we mark it as $1$ or phishy. The phishers try to mimic a phishy website as a clean one and all of the items on the web pages refer to a targeted website. The only thing that phishers cannot change is the domain name so they try to hide it. The search using the word set will return the domain names of the clean websites, but not the phishy ones. We chose 5 most important words because our experiments and statistical calculations indicated that the optimum number of the most important words is 5. Note that, Yue *et al.* [11] also used 5 most important words in his experiments. We used Python package of *TfidfVectorizer* developed by Scikit-learn [48] to implement this algorithm.

## 3.5 Added Feature: Shared in GooglePlus

Today, the social networks play an important role in our lives and many users spend a lot of time on them. We believe that gathering statistics from those networks and adding them to the dataset will improve the accuracy of the classifier. We added the number of times that a webpage was shared in the social network Google Plus. We used a Python library named SeoLib [49] for measuring this feature.

## 3.6 Added Feature: Content Security Policy

The Content-Security-Policy (CSP) HTTP response header helps users to cut the risk of cross-site scripting (XSS). The website administrator can declare what dynamic resources are allowed to be loaded in this particular web pages [50]. It also needs to be supported by web browsers. By this declaration, if a resource is out of the domain, the browser will not load it. As this is a very new capability, few numbers of websites support this as now.

# Chapter 4

# Initial Attempt: Experiment and Evaluation

The implemented framework starts with loading the list of websites from a Commas Separated Values (CSV) file. The input file contains the URLs of websites whose we plan to extract feature values and add them to the final dataset. If we know the website is legitimate, we will label it as a $-1$. Otherwise, we will label it as $1$. The *DataLoader* class is responsible for loading all of URL and labels and sending them to the *Evaluator* class. The *Evaluator* class goes through all of the given URLs and measures features and puts it in a result vector. For example, if the feature is the "Length of URL", the *Evaluator* class will calculate the length of given URL and add it to the result vector. Some of the features, like "the Domain Age", needs external information so the framework will use external APIs to get the values. When all the features have been evaluated, the vector is completed and will be returned as a result of this step. The framework will convert this vector to a comma-separated string and append it at the end of the dataset file. When framework goes through all of the websites, this process will finish. The Algorithm 1 describes the formal algorithm of this process.

**Algorithm 1** Creating Dataset

```
1: procedure CREATE DATASET($SourceAddress$)
2:     $URLs, Labels \leftarrow DataLoader(SourceAddress)$
3:     while end of URL list do
4:         $FeatureVector \leftarrow Evaluator.MeasureFeature(URL)$
5:         $VectorInCSV \leftarrow CSV(FeatureVector)$
6:         $DatasetFile \leftarrow Append(VectorInCSV)$
7:     return $DatasetFile$
```

In the next step, we get information from 6,000 different websites including 3,000 legitimate ones that we obtained from `Alexa.com` and 3,000 phishing websites that we obtained from `Phishtank.com`. We measured the values for each feature and added them to the dataset.

**Table 4.1:** Performance evaluation for dataset

| **Classifier** | **Accuracy** | **AUC** |
|---|---|---|
| TensorFlow Adagrad | 89.3 | 90.1 |
| TensorFlow Adadelta | 91.7 | 91.3 |
| TensorFlow GradientDescent | 91.6 | 91.1 |
| TensorFlow Linear | 81.5 | 85.4 |
| SVM Linear | 80.1 | 81.4 |
| SVM Guassian | 93.7 | 96.6 |

To evaluated the efficacy of the dataset, we used Support Vector Machine (SVM) which is a well-known machine learning classifier that uses some kernel function for classification. We used two different kernel functions: Linear and Gaussian. We ran all of our trained classifiers against the data collected. Using TensorFlow and TFcontrib (Abadi et al., 2016) we built a Deep Neural Network (DNN) using the following built-in optimization: GradientDescent, Adagrad, Adadelta. We also used TensorFlow to implement a linear classifier. We used Stratified-K-Fold cross-validation with five-fold.

In Table. 4.1 we show the performance of the implemented approach. We get the best accuracy by using to SVM with a Gaussian kernel. The accuracy is 93.7% which is considered very good.

We developed a framework that can extract any defining feature and create the up-to-date dataset. We took 3000 clean websites and 3000 phishing websites and subsequently we trained our classifiers over this dataset. We also analyzed a TensorFlow-based neural network, a TensorFlow-based linear classifier, an SVM with a Gaussian and linear kernel. We achieved an accuracy of 93.7%.

Having this framework helped us to extract features and conduct the experiment more easily but we had some more concerns. Using third-party services can not preserve the privacy of the users. Also, it takes much longer time to decide whether a website is phishing or not. We need to have a set of features and approaches that can be implemented on the client-side completely and will not use any third-party services. In addition, TF-IDF feature is a data-agnostic feature that relies on the dataset. So we need to define a better solution to address all these problems. In the next chapters, we will describe our final solution that uses the domain name as the viable parameter in defining feature and how we did address all problems mentioned prevoisuly.

# Chapter 5

# Improved Approach: Domain Name Based Features

With the Fresh-Phish framework, we learned several lessons and were able to get a better understanding of the features that need to be used for phishing detection. For instance, one lesson we learned is that the problem of phishing cannot be viewed in a statistical way and needs to consider the intent of the attackers for fooling the users. Therefore, the features should be finally chosen for phishing detection, should reflect this intuition. Another important lesson we learned was there is a strong correlation between the domain name and the nature of the websites. Based on these observations, we now describe our solution. Our work is the first solution to be entirely focused on the domain name of the phishing website. In our work, the domain name is the string before the top-level domain identifier, *e.g.*, for the URL `google.co.uk`, the domain name is `google`. We only concern ourselves with examining the landing page of this website and with the information that can be extracted from this page without the help of third-party servers, search engines or DNS servers.

Our approach is based on the intuition that the domain name of the phishing websites is a key indicator of a phishing attack. We design several features that are based on the domain name and train a machine learning classifier based on sample data. The trained classifier is used to test a suspicious website against these features. In the following, we describe the key challenges in our proposed approach and our solutions to these challenges.

The primary challenge is to justify the design of domain name based features. Towards this end, first, we highlight the subtle distinctions between the impact of the domain name and the URL of a phishing website. A URL can be a complex combination of special characters, numerical values, and other obfuscating features. A phisher has much control over the formation and structure of the URL and therefore, can generate noisy URLs that can bypass most machine learning approaches. On the other hand, the phisher has limited control over the domain name, *i.e.*, the adversary can generate several types of URLs with the same domain, but the domain name remains fixed throughout. While the phisher can change domain names, it usually takes some time to register a domain name and use it for such attacks. Second, domain name based features are likely to be more independent of the content in the phishing pages. The structure of the page layout, the HTML tags, and the dynamic content will no longer be a major part of the detection algorithm. Third, a phishing domain name typically can contain additional characters or numbers to give the illusion of a legitimate website. Such peculiarities continue to be part of several malicious phishing domains, *e.g.*, `gooogle.com`. These variations are subtle and are likely to provide sufficient statistical distinctions between legitimate and phishing websites. Hence, based on these arguments, we claim that the domain name based features are likely to exhibit more regularity than URL based features.

The next challenge is that the designed features could be data driven, that is, they can be biased with respect to the training data. To address this, our features exploit the relationship between the domain name and the visible content of the web page. For instance, one feature calculates the rank of the domain against all visible text on the web page, which is exhibited by almost all phishing websites. But, designing such features is a major challenge and requires deep analysis of the phishing websites over a period of time. Therefore, to get higher detection accuracy, we combine these with other features that are based on observations of phishing domain names reported on `PhishTank.com` and from observations of other researchers in the community. Our feature set is a mix of existing features and some new features. However, since our features are based on the domain names, we redesign the existing features and derive new features that reflect the nature of the phishing website.

The penultimate challenge concerns the validity of the features. Among the several possible domain name based features, many features may not add value to the machine learning classifier. We perform a statistical validation against a small sample of the data to verify the distribution of the features across the phishing and legitimate websites. With this approach, we were able to eliminate several features and our final classifier consists of only seven features.

The final challenge is testing the resiliency of the domain based features to detect unknown or zero-day phishing attackers. To address this, we tested the classifier against a blacklist of URLs that are taken from the latest updates on `OpenPhish.com`. Our approach showed excellent resiliency and was able to detect 97-99.7% of the URLs, in varying learning conditions.

## 5.1 Key Contributions

(a) We describe a machine learning (ML) based approach for phishing detection that relies entirely on domain name based features. Our approach is the first approach that has the combination of several benefits such as not using third-party servers, search engines, suspicious words and URL specific features.

(b) Our approach achieves 97% accuracy on a set of 2000 URLs with a five-fold cross-validation.

(c) Our approach achieves 97-99.7% detection rate on live blacklist data from `OpenPhish.com`, validating our base hypothesis of bias in datasets and at the same time, demonstrating the remarkable robustness of our learning model against phisher induced noise.

(d) The run-time detection speed of our approach is 4 times faster for legitimate websites and 10 times faster than the state-of-the-art work [18] in this domain.

(e) We demonstrate the bias induced in the learning model by certain features, such as URL length, which raises the question of revisiting many of the existing works in literature.

## 5.2 Correlation of Domain Name to Phishing Intent

In Figure 5.1, we show the differences between a legitimate website and phishing website. Figure 5.1 shows two webpages: one image belongs to legitimate `Facebook.com`which the domain name is Facebook. Another one belongs to a phishing website that mimics the `Facebook.com,` with a different domain name: Sanagustanturismo. The intuition behind this is adversaries try to hide their domain name in a phishing webpage while the legitimate websites try to emphasize the domain name in the webpage. Based on this intuition, we defined the features. For example, in the legitimate picture of Facebook.com, the page title contains the domain

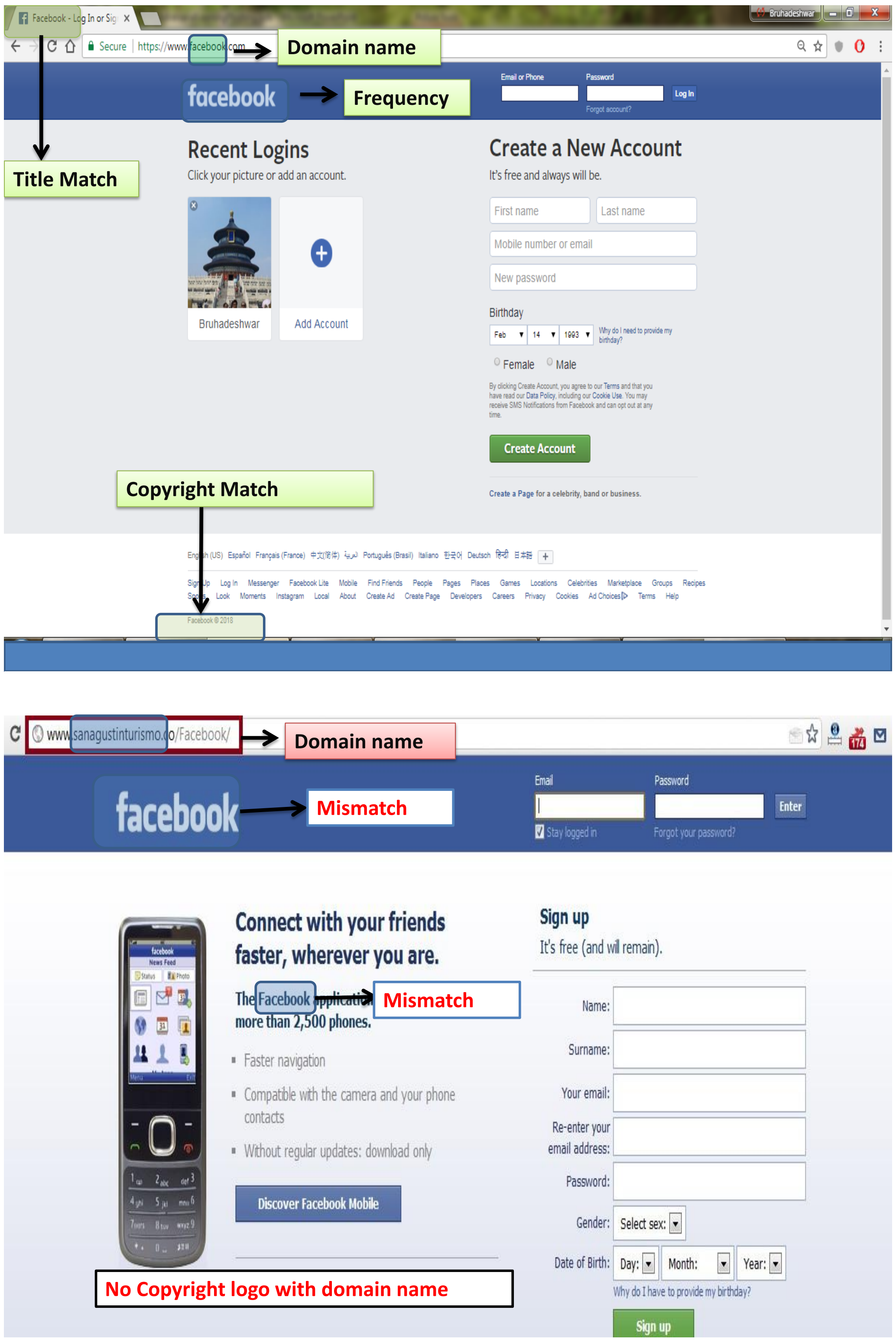


**Figure 5.1:** Domain name features for websites - Top: Legitimate website, Bottom: Phishing website

name. While in the phishing page, an adversary tries to hide the actual domain name, it does not include the domain name. Also, the legitimate Facebook.com includes the domain name near to the copyright logo while the phishing page does not have the copyright logo.

## 5.3 Feature Engineering and Validation

In this section, we characterize the phishing attacks using domain name based features and provide the necessary statistical validation for each of these features. As far as possible, our feature design attempts to be content-agnostic, *i.e.*, the feature design attempts to model the principles of phishing attacks and reduce the dependence of the features on specific data values. This is a challenging task and hence, we also consider additional features that are based on observations from real phishing attacks. Our feature sets consists of two types of features: binary, *i.e.*, the feature value is 0 or 1, and non-binary, *i.e.*, the feature is real-valued. In summary, the key point of our feature engineering is that all features depend on the domain name of the website and the relation of the domain with respect to the content of the website. These aspects ensure that our features are not affected by biased datasets and are robust to noise.

To validate the intuition behind each feature, we tested the empirical cumulative distribution function (ECDF) of the feature for 1000 known phishing websites against 1000 known legitimate websites. The ECDF of a real-valued random variable $X$, or just distribution function of $X$, evaluated at x, is the probability that $X$ will take a value less than or equal to x. For $x$-axis distribution, we used the feature values in the dataset, and for $y$-axis, we used the probability that features value will take values less than or equal to x. For the binary features, we counted the number of 1s for legitimate and phishing websites for each feature. For the non-binary features, we draw an ECDF plot for phishing and legitimate websites.

We also indicate if the features are "New", meaning designed by us, or "Existing", meaning that other researchers have designed it, in parentheses for each feature.

## 5.4 Domain-Base features

### 5.4.1 Feature 1 (Existing) : Domain Age

The domain age, given in number of years, is the count of time since the domain was registered and became active. Intuitively, the domain age of a phishing domain will be smaller than the domain age of a legitimate website. To evaluate this feature, we obtained the Whois information of registered domain from WhoisXMLAPI service and calculated the age of domain in years. Although this feature has been used by many other researchers in the past, we did not use it since we need to obtain it from a third party server. Further, in a sample experiment, which we do not report here, we discovered a surprising result that it does not have any positive impact on the classification accuracy.

### 5.4.2 Feature 2 (New) : Domain Length

The attackers who want to register a domain for phishing have to choose longer domain name in comparison with the legitimate website. The length of the domain name is the number of characters in the domain name string. As shown in Figure 5.2, the ECDF of this feature shows sufficient distinction between the legitimate websites and the phishing websites.

### 5.4.3 Feature 3 (Existing) : URL Length

The URL length is a popular feature among all known phishing detection approaches and is based on the intuition that phishing URLs are longer than legitimate URLs. We describe this feature here primarily to highlight the issue of dataset bias discussed in Section 1.5. In Figure 5.2, we show the ECDF of this feature. On the surface, it seems an excellent feature, however, it is completely data dependent and most existing works have generated results that are likely to be heavily influenced by the distribution of this feature in the phishing and legitimate datasets. We generated two sets of classification results: with and without the URL length, to demonstrate the impact of classification due to this feature. The average accuracy of classification increases by $2\%$ because of this feature and reaches $99\%$, which matches the state-of-the-art result when the only accuracy is considered. If the feature extraction time is also considered, we show that our results are better than the state-of-the-art works.

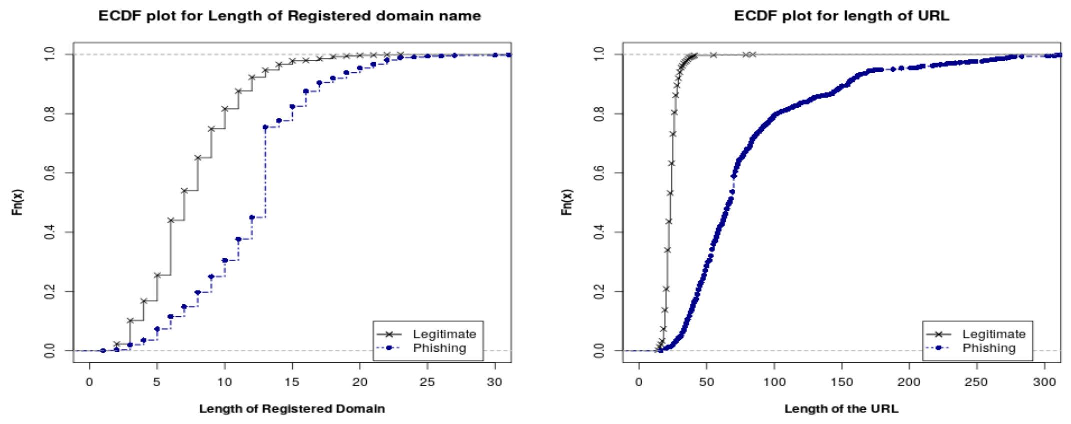


**Figure 5.2:** Domain length, Length of the URL

### 5.4.4 Feature 4 (Existing) : Link Ratio in BODY

This feature is defined as the ratio of the number of hyperlinks pointing to the same domain to the total number of hyperlinks on the web page. The intuition is that, in the process of making a phishing website similar to the legitimate website, the attackers refer the hyperlinks on the landing page to a legitimate domain name, which is different from the domain name displayed in the address bar of the browser. This feature is content-agnostic as the ratio can compute for any phishing website that exhibits this behavior. For example, the phishers create a phishing page to mimic a well-known payment service where all links on the page are to a legitimate website except the login-form in which the users need to enter their information. Accordingly, the ratio of the links referring to current domain compared to all links found in the website will be different when compared between a phishing website and a legitimate website. To evaluate this feature, we find all of the links in the page and the ratio of links referring to the current page over the number of all links found on the page. However, some legitimate websites also exhibited this behavior and therefore, we used a scaling process to derive the final value of the feature. For instance, if for a given website the ratio was in the range $[0.1, 0.2]$, we assigned the value $20$ to this feature. Figure

5.3, shows the ECDF of this feature (of the raw ratios) with sufficient separation between the two distributions.

### 5.4.5 Feature 5 (Existing) : Links in Meta-header (HEADER)

This is the defined as the ratio of the number of hyperlinks in the *META* section of the page that point to the same domain to the total number of links in the *META* section. The attackers use some of the scripts in *META* section of the HTML page from a legitimate website to refer to a website outside of the current domain. Therefore, if this ratio is small then it indicates a phishing website, as shown in Figure 5.3. However, this feature was not found to be that useful as shown in Figure 5.3 as many legitimate websites tended to use many outside links like `Google Ads` or `Google Analytics`, images from other websites and so on. Therefore, we did not use this feature in our final classification.

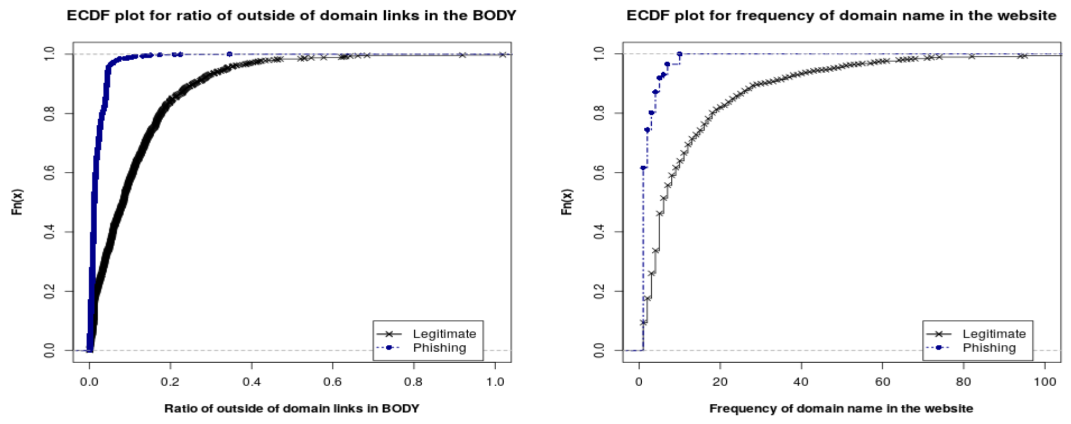


**Figure 5.3:** Link Ratio in BODY, Link Ratio and Domain Name Frequency

### 5.4.6 Feature 6 (New) : Frequency of Domain Name

This feature counts the number of times the domain name appears as a word in the visible text of the web page. The intuition is that many web pages repeat the domain name several times on their web page, as part of disclaimers, privacy terms and so on. Therefore, if the domain name does not appear at all on the web page, then there is something suspicious about such a web page.

**Table 5.1:** Binary feature distribution

| Feature | Legitimate | Phishing |
|---|---|---|
| HTTPS Present | 0.92 | 0.23 |
| Non-alphabetical Characters | 0.05 | 0.36 |
| Copyright Logo Match | 0.26 | 0.0 |
| Page Title Match | 0.87 | 0.03 |

This is yet another feature that captures the relationship of the domain name to the web page. The ECDF of this feature is shown in Figure 5.3.

The ECDF is shown in Figure 5.3. This feature was used in a different way by other researchers, notably [14], but with the help of search engines. Our definition captures a different essence compared to their approach and does not rely on third-party servers.

### 5.4.7 Feature 7 (Existing) : HTTPS Present

An SSL certificate is issued for a particular domain name. Most legitimate websites used SSL certificates and operated over HTTPS protocol. Therefore, if a website uses HTTPS, the feature value is 1 and if not, it is 0.

### 5.4.8 Feature 8 (New) : Non-alphabetical Characters in Domain Name

The attackers use non-alphabetical characters, like numbers or hyphen, to generate new unregistered phishing domain names, which are very similar to legitimate domain names. If domain name includes any non-alphabetic character, this feature will be evaluated to 1, otherwise, it will be set to 0. Many past works [19, 25] have considered a variant of this feature, *i.e.*, they examined the number of special characters in the entire URL. However, as we stated earlier, generating custom based noisy URLs is a relatively easy task for the attackers. Table 5.1 shows the percentage distribution of the binary features.

### 5.4.9 Feature 9 (New) : Domain Name with Copyright Logo

Many legitimate websites use the copyright logo to indicate the trademark ownership of their organization name. Usually, the domain name is placed before or after the copyright logo for such websites. To generate this feature, we considered up to 50 characters before and after the copyright logo, removed the white spaces, and checked for the presence of the domain name in the resulting string. Surprisingly, we found that none of the phishing websites placed their actual domain names along with the copyright logo. To do so, would have aroused the suspicion of any web user and therefore, we found this feature to be an excellent distinguisher.

### 5.4.10 Feature 10 (New) : Page Title and Domain Name Match

Many legitimate websites repeat the domain name in the title of the web page. We found that many phishing websites used this feature to deceive users into believing that they were visiting legitimate websites. But, clearly, a phishing website would not use the phishing domain name on the title page as it would be clearly visible to the user. Our intuition proved right and we found that less than 3% of the phishing websites were using this feature, but over 87% of legitimate websites had this feature. Table 5.1, shows the distribution of these features on the sample dataset.

**A Comparison with [18].** In [18], although the authors have alluded to the use of the domain name as *one* of the factors and described several features, they did not base their approach entirely on this aspect as we have done in our work. Some of the features common with our work are Feature 6, the frequency of occurrence of the domain name, and Feature 10, the match of the domain name with the title along with some more domain name based features. Furthermore, the approach in [18] uses many other features, over 200, to perform the final classification and even ignored some domain name based features. For instance, they ignored the Feature 9, domain name match with copyright logo, which we found very useful in detecting phishing websites.

# Chapter 6

# Machine Learning Models

We briefly describe the classifiers and the configuration settings used in our experiments. We used five standard classifiers: Support-Vector Machine (SVM) with two different kernels, *linear* and *Gaussian* , Decision tree, Gaussian Naive Bayes, $k$-nearest-neighbors ($k$NN) and Gradient Boosting. We also considered Majority voter, which uses a major voting approach to classify an instance. In our approach, we assume that a particular classifier is attempting to classify a given website into one of two classes: legitimate or phishing.

## 6.1 Support Vector Machine (SVM)

SVMs [51, 52] are supervised learning models that represent data as points in space and construct a hyperplane in high-dimensional space to separate the data points into different categories.

We used two different kernel functions: linear and Gaussian, to classify a set of $n$ points $\{\vec{x_1}, y_1\}, \cdots, \{\vec{x_n}, y_n\}$ where each $\vec{x}$ is a $d$-dimensional vector and $d$ corresponds to the number of features characterizing $\vec{x}$ and $y_i$ corresponds to one of the classification labels $\{-1, 1\}$ of $\vec{x_i}$. Broadly speaking, the linear kernel uses a linear mathematical function such as, $\vec{w}.\vec{x} - b = 0$ where $b$ is a system parameter, to define a best possible hyper-plane that divides the group of points $\vec{x_p}$ where $y = 1$ from the points where $y = -1$.

Similarly, a Gaussian kernel uses a radial-basis function (RBF) defined as:

$$k(\vec{x_i}, \vec{x_j}) = exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$$

It achieves a similar result based on $\gamma$, a system defined parameter. For Gaussian kernel, we need to specify two important parameters: *cost* and $\gamma$. The *cost* parameter determines the margin for examples to be ignored because they are classified incorrectly [48]. The $\gamma$ parameter adjusts how similar two points are in order to be considered "similar" and labels them identically. One well-known approach for assigning *cost* and $\gamma$ parameters are by using a grid search, where one axis of the grid represents *cost* and the other denotes $gamma$. In our experiments, we used a grid search to find the best available value for *cost* and $\gamma$.

## 6.2 Decision Tree Classifier

Decision tree classifier [53] is a decision tree based classifier, which consists of tree nodes with values "learned" from the training instances and branches leading towards the best possible decision about the input instance. A decision tree's interior node contains an attribute test, such as a range, on one more variable. Each internal node can be seen as a "splitter" that partitions the instance space into two or more smaller sub-spaces based on the attribute test. The leaf nodes of a decision tree corresponding to the decisions or classes of a particular input instance that satisfies the conditions along the path from the root node to the leaf node. The general problem of learning an optimal decision tree is proven to be NP-complete. For large attribute spaces, decision trees can create complex trees, a problem known as *overfitting*, which indicates the poor quality of data space partitioning. Decision trees perform quite well on small attributes spaces. Our approach uses only seven attributes, which makes decision trees quite effective. We used the default values set in the Scikit-learn Python package [48].

## 6.3 Gradient Boosting

Gradient boosting [54, 55] is a gradient descent based learning approach that produces a prediction model as an ensemble of weak prediction models. The learning starts with a "weak" model, typically Gradient Boost Regression Tree (GBRT) that tries to learn the data space and is iteratively improved by the next model that reduces the error of the previous model. The goal of gradient boosting is to combine weak learning models into a single strong model as shown: $F(x) = \sum_{m=1}^{M} \gamma_m h_m(x)$. Typically, $h_m$ is GBRT of fixed depth, which is iteratively improved over $M$ trials and $\gamma_m$ is the regression parameter for that particular iteration. At each step, the model is improved as follows:

$$F_{m+1}(x) = F_m(x) + \gamma_{m+1} h_{m+1}(x)$$

The $h_{m+1}$ is chosen to minimize the loss function $L$ in the current model's fitting of a data point $x_i$: $F_m(x_i)$ as shown:

$$F_{m+1}(x) = F_m(x) + \underset{h}{\mathsf{argmin}} \sum_{i=1}^{n} L(y_i, F_m(x_i) + h(x))$$

For implementation we used Scikit-learn library [48] and we set the tool-kit specific parameters as follows: $n\_estimators = 100$, which denotes the number of weak learners, and the maximum depth of each tree is controlled by $max\_depth$ parameter. We set the $learning\_rate = 1.0$ and $max\_depth = 1$.

## 6.4 Gaussian Naive Bayes Classifier

Gaussian Naive Bayes classifier [56] is a set of supervised learning algorithms based on applying Bayes theorem with the assumption of independence between every pair of features. Briefly, given a data vector: $\vec{x} = \{x_1,\ x_2, \cdots,\ x_n\}$ with $n$ distinct features, the Bayes conditional probability model assigns the following conditional probability: $P(C_k|x_1,\ x_2, \cdots,\ x_n)$, for each possible type of class $C_k$. Since each of the features is independent, this equation can be computed in a straight-forward manner using Bayes theorem and the chain rule of conditional probability. Using this result, the Bayes classifier is constructed by calculating the *maximum a posteriori* probability and assigning the corresponding class label to the data instance in question. The main advantage of naive Bayes is that it only requires a small amount of training data to estimate the parameters necessary for classification.

## 6.5 k-nearest-neighbors ($k$NN)

$k$NN [57, 58] is an effective supervised learning method for many problems including security techniques [59]. The $k$NN algorithm is based on the clustering of the elements that have the same characteristics; it decides the class category of a test example based on its $k$-neighbors close to it. The value of $k$ in the kNN depends on the size of the dataset and the type of the classification problem [60].

## 6.6 Majority Voter

This is a classifier that we defined to combine the results from other classifiers as a voter to decide whether a given instance is legitimate or not. The classifier is based on the majority-vote rule, *i.e.*, the common decision of the majority of classifiers is the decision of this classifier. To make this classifier, we employed all another classifier that we discussed and trained them in this classifier. For testing purpose, we test given instance with all other classifiers and get the predicted value for that using the majority vote.

# Chapter 7

# Improved Approach: Performance and Evaluation

## 7.1 Experimental Methodology

We conducted two sets of experiments to assess the performance of our model trained with various machine learning classifiers. The first set of experiments were conducted on a prepared dataset and the second set of experiments were conducted on live unknown phishing dataset from `OpenPhish.com`. Only one past work [25] demonstrated a similar result on unknown datasets with a detection rate of $95\%$. In contrast, our approach achieves much higher detection accuracy, close to $99.7\%$.

## 7.2 Feature Set

For our experiments we did not include the domain age (*Feature 1*) and link ratio in HEADER (*Feature 5*) The domain age feature requires a third-party service, which violates user privacy, and the link ratio in HEADER feature did not really exhibit significant variation across the legitimate and phishing datasets. For the other experiments, we performed the experiments in two variations: without including the URL length feature (*Feature 3*) and by including this feature. The URL length feature is one such feature that will exhibit significantly different distribution for phishing and legitimate URLs, as phishing URLs are typically longer in the datasets available in public domain.

We report the various classification metrics including true positive rate and accuracy. We show the time taken to extract the feature values for each website, the training time for each classifier, and the time is taken by the classifier to predict whether a website is phishing or not. We implemented our approach using the Sci-kit [48] library in Python 2.7 on a desktop running Fedora 24 OS with Intel Core$^{®}$ 2 Duo CPU E8300$^{©}$ processor having 6 GB RAM and clocking at 2.4 GHz.

**Table 7.1:** Number and source of instances used in creating dataset

| # | Source | # instances | Category | Usage |
|---|---|---|---|---|
| 1 | `Alexa.com` | 1000 | Legitimate | Train & Test |
| 2 | `PhishTank.com` | 1000 | Phishing | Train & Test |
| 3 | `Openphish.com` | 2013 | Phishing | Test |

## 7.3 Datasets

For creating a dataset and extracting the features, we used three different sources. For the list of legitimate websites, we obtained the 1000 top ranked websites from the `Alexa.com` and assumed them as legitimate. For the phishing websites, we got 1000 phishing websites from `PhishTank.com` and another 2013 phishing websites from `OpenPhish.com`. We describe the use of these datasets in the following discussion. Note that, all our features, except URL length, only examine the domain name and its relation to the web page, they do not behave differently for URLs of different lengths.

### 7.3.1 Dataset 1: DS-1

This set includes 1000 legitimate websites from `Alexa.com` and 1000 phishing websites from `PhishTank.com`. In the experiments, we trained and tested on this dataset with 80% data for training and 20% data for testing using five-fold cross-validation.

### 7.3.2 Dataset 2: DS-2

This dataset includes 1000 legitimate websites from `Alexa.com` and 3013 phishing websites from both `PhishTank.com` and `OpenPhish.com`. For this dataset, we considered 1000 legitimate and 1000 phishing websites for training without cross-validation. The remaining 2013 websites were used for testing. Table 7.1 shows the testing and training configurations of these datasets.

## 7.4 Performance Metrics

For evaluating our approach, we used standard metrics used to judge any machine learning based anti-phishing detection approach. We calculated True Positive Tate (TPR), True Negative Rate (TNR), Positive Predictive Value (PPV), Accuracy (ACC) and Area Under Curve (AUC) to evaluate the performance of proposed approach. Table 7.2 shows the metrics used for classification of phishing and legitimate websites.

- $N_L$ denotes the total number of legitimate websites in the dataset.
- $N_P$ denotes the total number of phishing websites in the dataset.
- $N_L \rightarrow L$ denotes number of legitimate websites classified as legitimate.
- $N_L \rightarrow P$ denotes number of legitimate websites classified as phishing.
- $N_P \rightarrow P$ denotes number of phishing websites classified as phishing.
- $N_P \rightarrow L$ denotes number of phishing websites classified as legitimate.

**Table 7.2:** Performance metrics used in the study

| Measure | Formula | Description |
|---|---|---|
| TPR | $\frac{N_P \rightarrow P}{N_P} * 100$ | Rate of *correctly* classified phishing |
| TNR | $\frac{N_L \rightarrow L}{N_L} * 100$ | Rate of *correctly* classified legitimate |
| PPV | $\frac{N_P \rightarrow P}{N_P \rightarrow P + N_L \rightarrow P} * 100$ | Rate of *correctly* predicted phishing over *total* predicted phishing |
| ACC | $\frac{N_L \rightarrow L + N_P \rightarrow P}{N_L + N_P} * 100$ | Rate of classified *correctly* in the dataset height |

## 7.5 Experiment1: Performance on DS-l

We designed two different experiments to evaluate the accuracy of classifiers on DS-1.

In the first experiment, we used all the features described in Section 5.2 except domain age, URL length and Links in Meta-header features. The age of the domain is a feature that requires the help of a third-party server.

In the second experiment, we included URL length and demonstrated the increase in classification accuracy. However, this increase is only due to the bias of the dataset with respect to this feature, *i.e.*, for phishing websites we have full URLs and for legitimate websites we have only the index pages of the domains.

### 7.5.1 Results without URL Length Feature

*Our domain name based approach achieves 97% accuracy and validates our basic hypothesis.* We ran our experiments with a five-fold cross-validation and show the results in Figure 7.1 and Figure 7.2. For each of the parameters, we show the maximum value achieved and the average value across all the validations. Of all the classifiers, Gradient Boosting performed the best with a maximum accuracy of $99.55\%$ percentage and an average accuracy of $97.74\%$. For Gradient Boosting and Majority Voting, the TPR is very high, $98.12\%$ and $97.46\%$, respectively, showing the high phishing detection capability of the classifiers. There is a similar trend in TNR for these two classifiers, *i.e.*, it is very high, showing the ability of the learning models to identify legitimate websites. We note that our average accuracy of $97.74\%$ is very high when compared several existing works that used a rather large and diverse set of features.

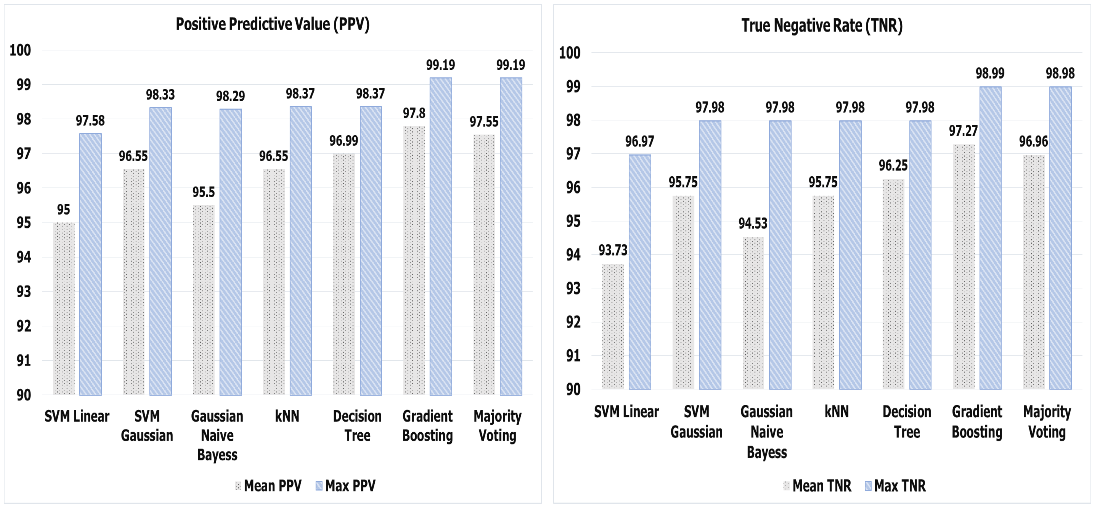


**Figure 7.1:** PPV and TNR on DS-1 without URL length feature

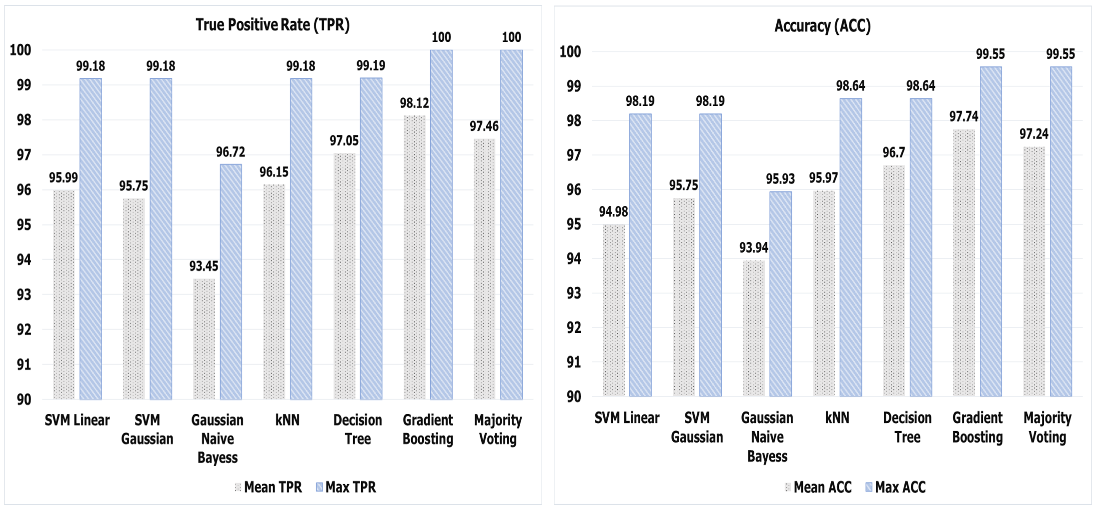


**Figure 7.2:** TPR and ACC on DS-1 without URL length Feature

### 7.5.2 Results with URL Length Feature

*The URL length feature results in higher accuracy and clearly demonstrate the bias induced in the dataset.*

We show the results of these experiments in Figure 7.3 and Figure 7.4. There is an increasing trend across all the classifiers for all the parameters considered. The main benefit is seen in the TNR whose average value increased from $95\%$, in the previous experiment, to $98\%$ across SVM linear, $k$NN, Decision tree and Majority voting classifiers. Excepting Gaussian Naive Bayes, all other classifiers recorded a TPR of $98\%$ and above, on an average, with the maximum hitting $100\%$ for three classifiers. The accuracy also showed an increasing trend with the average accuracy increasing to $98.8\%$ for Gradient Boosting, and the maximum accuracy reaching $99.55\%$ for several other classifiers. This experiment clearly shows that features like URL length that depend on the URL tend to impact classification accuracy depending on the dataset.

In Table 7.3 and Table 7.4, we show the corresponding table of performance metrics of our classifiers. For clarity, we show both the mean and maximum achieved values for each metric, except for the AUC.

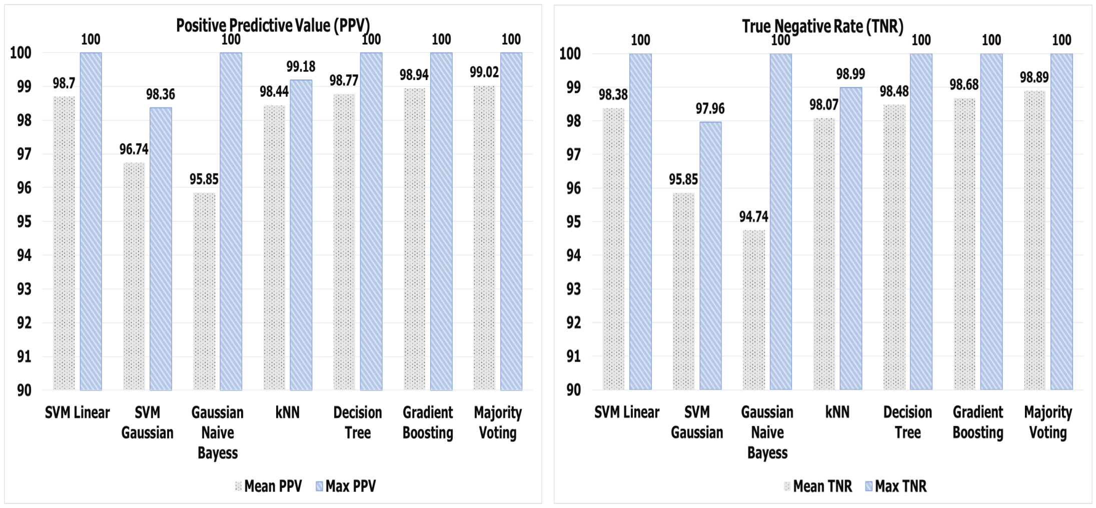


**Figure 7.3:** PPV and TNR on DS-1 with URL length feature

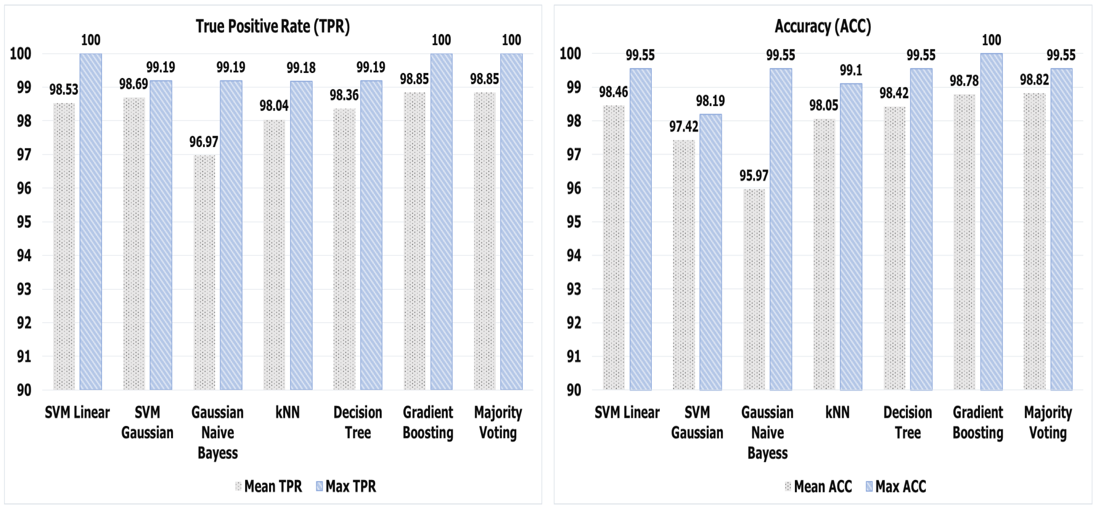


**Figure 7.4:** TPR and ACC on DS-1 with URL length feature

**Table 7.3:** Performance metrics for DS-1 without URL length

| Classifier | TPR | | TNR | | PPV | | ACC | | AUC |
|---|---|---|---|---|---|---|---|---|---|
| | Mean | Max. | Mean | Max. | Mean | Max. | Mean | Max. | Mean |
| **SVM Linear** | 95.99 | 99.18 | 93.73 | 96.97 | 95 | 97.58 | 94.98 | 98.19 | 0.9879 |
| **SVM Gaussian** | 95.75 | 99.18 | 95.75 | 97.98 | 96.55 | 98.33 | 95.75 | 98.19 | 0.9872 |
| **Gaussian NB** | 93.45 | 96.72 | 94.53 | 97.98 | 95.5 | 98.29 | 93.94 | 95.93 | 0.9816 |
| **kNN** | 96.15 | 99.18 | 95.75 | 97.98 | 96.55 | 98.37 | 95.97 | 98.64 | 0.9814 |
| **Decision tree** | 97.05 | 99.19 | 96.25 | 97.98 | 96.99 | 98.37 | 96.7 | 98.64 | 0.9872 |
| **Gradient Boosting** | 98.12 | 100 | 97.27 | 98.99 | 97.8 | 99.19 | **97.74** | 99.55 | 0.9926 |
| **Majority Voting** | 97.46 | 100 | 96.96 | 98.99 | 97.55 | 99.19 | **97.24** | 99.55 | 0.9914 |

**Table 7.4:** Performance metrics for DS-1 with URL length

| Classifier | TPR | | TNR | | PPV | | ACC | | AUC |
|---|---|---|---|---|---|---|---|---|---|
| | Mean | Max. | Mean | Max. | Mean | Max. | Mean | Max. | Mean |
| **SVM Linear** | 98.53 | 100 | 98.38 | 100 | 98.7 | 100 | **98.46** | 99.55 | 0.9942 |
| **SVM Gaussian** | 98.69 | 99.19 | 95.85 | 97.96 | 96.74 | 98.36 | 97.42 | 98.19 | 0.9919 |
| **Gaussian NB** | 96.97 | 99.19 | 94.74 | 100 | 95.85 | 100 | 95.97 | 99.55 | 0.9867 |
| **kNN** | 98.04 | 99.18 | 98.07 | 98.99 | 98.44 | 99.18 | **98.05** | 99.1 | 0.9903 |
| **Decision tree** | 98.36 | 99.19 | 98.48 | 100 | 98.77 | 100 | **98.42** | 99.55 | 0.9814 |
| **Gradient Boosting** | 98.85 | 100 | 98.68 | 100 | 98.94 | 100 | **98.78** | 100 | 0.9941 |
| **Majority Voting** | 98.85 | 100 | 98.79 | 100 | 99.02 | 100 | **98.82** | 99.55 | 0.9943 |

### 7.5.3 Timing Analysis for DS-1

Although the timing analysis can vary from machine to machine, we attempt to show that our approach is quite efficient even when it is run from a relatively low-end desktop configuration such as ours.

#### Feature extraction timings

*Our feature extraction time is very low, of the order of few milliseconds, and demonstrates the efficiency of our feature set.*

Table 7.6 shows the results of our feature extraction. The total time for extracting features of a legitimate website is about $0.117$ seconds and for a phishing website is $0.02$ seconds, which indicates the real-time nature of our approach. This is extremely low compared to the state-of-the-art approach in [18] where the extraction time was in the order of a few seconds. Furthermore, we would like to emphasize that the average loading time of a web page like `msn.com`, is around 1 second and our feature extraction adds only a few milliseconds overhead to this process.

**Table 7.5:** Training and classification timings

| **Classifier** | **Training Time** (in ms) | **Testing** (in $\mu$s) |
|---|---|---|
| SVM Linear | 1339.85 | 6.74 |
| SVM Gaussian | 703.62 | 38.32 |
| Gaussian Naive Bayes | 2.28 | 1.47 |
| kNN | 7.36 | 14.85 |
| Decision tree | 2.49 | 0.80 |
| Gradient Boosting | 2737.56 | 450.48 |
| Majority Voting | 177.73 | 3.25 |

**Table 7.6:** Feature extraction timings

| **Feature** | **Legitimate** ($\mu$s) | **Phishing** ($\mu$s) |
|---|---|---|
| HTTPS Present | 4.12 | 3.87 |
| Domain Length | 63.45 | 66.45 |
| Page Title Match | 26.9 | 32.3 |
| Frequency Domain Name | 333.8 | 33.09 |
| Non-alphabetic Characters | 32.64 | 13.68 |
| Copyright Logo Match | 2737.56 | 450.48 |
| Link Ratio in Body | 114482.87 | 19445.67 |
| URL Length | 0.3576 | 0.5066 |
| Total Time (in seconds) | 0.117 | 0.02 |

**Training and Classification Timings**

*Our classifier training and classification times are very low, of the order of few microseconds, and again demonstrates the efficiency of our approach.*

The testing times reported are the average across the five-fold cross-validation and do not include the feature extraction time. The training can be done off-line and the testing takes a few microseconds to perform, after the feature extraction. Given that cumulative time for feature extraction and testing is less than 2 milliseconds, we claim that our approach can be deployed in practice as a client-side browser plug-in.

**Table 7.7:** Performance metrics for DS-2 without and with URL length

| **Classifier** | **TPR without URL Length** | **TPR with URL Length** |
|---|---|---|
| SVM Linear | 94.09 | 94.24 |
| SVM Gaussian | 92.75 | 90.81 |
| Gaussian Naive Bayes | 91.06 | 92.75 |
| kNN | 93.74 | **99.7** |
| Decision tree | 97.91 | 97.27 |
| Gradient Boosting | **98.21** | **99.75** |
| Majority Voting | 95.33 | 97.67 |

## 7.6 Experiment 2: Performance on DS-2

In this experiment, we examine the robustness of our learning approach on unknown and unseen data. We obtained a list of 2013 live phishing websites from `OpenPhish.com`. Although a higher number of sites were listed, many sites were unavailable and few were blocked by the corresponding ISPs. We trained the classifier in two modes: without including the URL length feature and with the URL length feature included. Finally, we tested the resulting classifier on the 2013 data instances and show the results in Table 7.7. These results show the remarkable performance of our approach. Unlike the previous approach [25], which attempted a similar experiment, for many of our classifiers, the TPR largely remains *unchanged* across both the experiments and even shows a slight increase for Decision tree and Gradient Boosting classifiers. Furthermore, when including URL length, the TPR even reaches $99.7\%(!)$ for $k$NN and Gradient Boosting. This result also confirms our hypothesis that the domain name based features can accurately capture the nature of a phishing website.

## 7.7 Comparison with Previous Work

We compare our results empirically with existing state-of-the-art solutions in Table 7.8. Our basis for comparison is the number of features, the accuracy, whether client-side features only are used or third-party features are included and average accuracy. We did not include the run-times of the approaches as that is a system specific metric. However, we note that our scheme reports micro-second level feature extraction and classification time, even when running on a relatively low-performance laptop with Core 2 Duo processor.

**Table 7.8:** Comparison with state-of-the-art approaches

| **Approach** | **♯ Leg** | **♯ Phish** | **♯ Features** | **ACC** | **Client Side** |
|---|---|---|---|---|---|
| Cantina [11] | 2100 | 19 | 7 | 96.97 | No |
| Cantina+ [14] | 1868 | 940 | 15 | 97 | No |
| Verma *et al.* [25] | 13274 | 11271 | 35 | 99.3 | Partial |
| Off-the-Hook [18] | 20000 | 2000 | 210 | 99.9 | Yes |
| Our App. Without URL Length | 1000 | 3013 | 8 | 97.7 | Yes |
| Our App. Wth URL Length | 1000 | 3013 | 9 | 98.8 | Yes |

# Chapter 8

# Conclusions and Future Work

## 8.1 Conclusion

In this work, we considered the problem of phishing detection using machine learning approaches. In our first attempt, called Fresh-Phish Framework, we identified several interesting features by looking at the problem from the statistical point of view. From the lesson of Fresh-Phish, we found that viewing problem purely from statistical point of view is insufficient to solve it effectively. The intent of the phishing attacker also needs to be considered for an effective solution.

We described the first approach towards the design of *only* domain name based features for detection of phishing websites using machine learning. Our feature design emphasized on the elimination of the possible bias in classification due to differently chosen datasets of phishing and legitimate pages. Our approach differs from all previous works in this space as it explores the relationship of the domain name to its intent for phishing. With only seven features we are able to achieve a classification rate of $97\%$ with cross-validated data. Furthermore, we were able to show a detection rate of 97-$99.7\%$ for live black-listed URLs from `OpenPhish.com`. This shows that our approach is able to adapt to the complex strategies used by phishers to evade such detection mechanisms. As our features explore the content found in the visible space of the web page, an attacker will need to put huge effort to bypass our classification. In trying to bypass our approach, an adversary may end up designing a page that will make any user suspicious of its intent. Furthermore, we demonstrated the shortcoming of using URL features such as URL lengths, that seem to give higher accuracy but may not do so in the near future. Our feature extraction and classification times are very low and show that our approach is suitable for real-time deployment. Our approach is likely to be very effective in modern day phishing strategies like extreme phishing that are designed to deceive even experienced users.

## 8.2 Future Work

In future, we wish to explore the robustness of machine learning algorithms for phishing detection in the presence of newer phishing attacks. We are also developing a real-time browser add-on that will provide warnings when visiting suspicious sites.

# Bibliography

[1] B. B. Gupta, Aakanksha Tewari, Ankit Kumar Jain, and Dharma P. Agrawal. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications*, pages 1–26, March 2016.

[2] APWG. Global phishing survey: Trends and domain name use in 2016, 2017. [Online; accessed 21-October-2016].

[3] Wikipedia. Phishing — Wikipedia, the free encyclopedia, 2016. [Online; accessed 21-October-2016].

[4] Junaid Ahsenali Chaudhry, Shafique Ahmad Chaudhry, and Robert G Rittenhouse. Phishing attacks and defenses. *International Journal of Security and Its Applications*, 10(1):247–256, 2016.

[5] Rui Zhao, Samantha John, Stacy Karas, Cara Bussell, Jennifer Roberts, Daniel Six, Brandon Gavett, and Chuan Yue. Design and evaluation of the highly insidious extreme phishing attacks. *Computers & Security*, 70(Supplement C):634–647, September 2017.

[6] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani. Systematization of knowledge (sok): A systematic review of software-based web phishing detection. *IEEE Communications Surveys Tutorials*, 19(4):2797–2819, 2017.

[7] Neda Abdelhamid, Fadi A. Thabtah, and Hussein Abdel-jaber. Phishing detection: A recent intelligent machine learning comparison based on models content and features. In *IEEE Int. Conf. on Intelligence and Security Informatics (ISI)*, pages 72–77, 2017.

[8] André Bergholz, Jan De B., Sebastian Glahn, Marie-Francine Moens, Paaß Gerhard, and Siehyun Strobel. New filtering approaches for phishing email. *Journal of Computer Security*, 18(1):7–35, 2010.

[9] John Yearwood, Musa Mammadov, and Arunava Banerjee. Profiling phishing emails based on hyperlink information. In *Proc. of IEEE/ACM Advances in Social Network Analysis and Mining (ASONAM)*, pages 120–127, 2010.

[10] Jingguo Wang, Tejaswini Herath, Rui Chen, Arun Vishwanath, and H Raghav Rao. Research article: Phishing susceptibility: An investigation into the processing of a targeted spear phishing email. *IEEE Trans. on Professional Communication*, 55(4):345–362, 2012.

[11] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM, 2007.

[12] Ram B. Basnet, Srinivas Mukkamala, and Andrew H. Sung. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry. Studies in Fuzziness and Soft Computing*, volume 226, pages 373–383. Springer, 2008.

[13] Haijun Zhang, Gang Liu, Tommy W. S. Chow, and Wenyin Liu. Textual and visual content-based anti-phishing: A bayesian approach. *IEEE Trans. on Neural Networks*, 22(10):1532–1546, 2011.

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

[15] R. Gowtham and Ilango Krishnamurthi. A comprehensive and efficacious architecture for detecting phishing webpages. *Computers and Security*, 40:23–37, 2014.

[16] Samuel Marchal, Kalle Saari, Nidhi Singh, and N Asokan. Know your phish: Novel techniques for detecting phishing sites and their targets. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 323–333. IEEE, 2016.

[17] Qian Cui, Guy-Vincent Jourdan, Gregor V Bochmann, Russell Couturier, and Iosif-Viorel Onut. Tracking phishing attacks over time. In *Proc. of the 26th Int. World Wide Web (WWW) Conf.*, pages 667–676, 2017.

[18] Samuel Marchal, Giovanni Armano, Tommi Grondahl, Kalle Saari, Nidhi Singh, and N. Asokan. Off-the-hook: An efficient and usable client-side phishing prevention application. *IEEE Trans. on Computers*, 66(10):1717–1733, 2017.

[19] Ankit Kumar Jain and B. B. Gupta. Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*, Dec 2017.

[20] Routhu Srinivasa Rao and Alwyn Roshan Pais. Detection of phishing websites using an efficient feature-based machine learning framework. *Neural Computing and Applications*, January 2018.

[21] Rui Zhao, Samantha John, Stacy Karas, Cara Bussell, Jennifer Roberts, Daniel Six, Brandon Gavett, and Chuan Yue. The highly insidious extreme phishing attacks. In *Proc. of the 25th Int. Conf. on Computer Communication and Networks (ICCCN)*, pages 1–10. IEEE, 2016.

[22] Rui Zhao, Samantha John, Stacy Karas, Cara Bussell, Jennifer Roberts, Daniel Six, Brandon Gavett, and Chuan Yue. Design and evaluation of the highly insidious extreme phishing attacks. *Computers & Security*, 70:634 – 647, 2017.

[23] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. A framework for detection and measurement of phishing attacks. In *Proc. of the 5th ACM Workshop on Recurring Malcode (WORM)*, pages 1–8. ACM, 2007.

[24] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious urls. In *Proc. of the 15th ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 1245–1254. ACM, 2009.

[25] Rakesh Verma and Keith Dyer. On the character of phishing urls: Accurate and robust statistical learning classifiers. In *Proc. of 5th ACM Conference on Data and Applications Security and Privacy (CODASPY)*, pages 111–122, 2015.

[26] Mohammed Al-Janabi, Ed de Quincey, and Peter Andras. Using supervised machine learning algorithms to detect suspicious urls in online social networks. In *IEEE/ACM Advances in Social Network Analysis and Mining (ASONAM)*, pages 1104–1111, 2017.

[27] Choon Lin Tan, Kang Leng Chiew, KokSheik Wong, and San Nah Sze. Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder. *Decision Support Systems*, 88(C):18–27, 2016.

[28] Wei Zhang, Qingshan Jiang, Lifei Chen, and Chengming Li. Two-stage elm for phishing web pages detection using hybrid features. *World Wide Web*, 20(4):797–813, 2017.

[29] Matthew Jensen, Alexandra Durcikova, and Ryan Wright. Combating Phishing Attacks: A Knowledge Management Approach. January 2017.

[30] Steve Sheng, Bryant Magnien, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Anti-Phishing Phil: The Design and Evaluation of a Game That Teaches People Not to Fall for Phish. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, pages 88–99, New York, NY, USA, 2007. ACM.

[31] Nalin Asanka Gamagedara Arachchilage, Steve Love, and Konstantin Beznosov. Phishing threat avoidance behaviour: An empirical investigation. *Computers in Human Behavior*, 60(Supplement C):185–197, July 2016.

[32] Deanna D Caputo, Shari Lawrence Pfleeger, Jesse D Freeman, and M Eric Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy*, 12(1):28–38, 2014.

[33] Sadia Afroz and Rachel Greenstadt. Phishzoo: Detecting phishing websites by looking at them. In *Proc. of 5th IEEE Int. Conf. on Semantic Computing (ICSC)*, pages 368–375. IEEE, 2011.

[34] Ankit Kumar Jain and BB Gupta. A novel approach to protect against phishing attacks at client-side using auto-updated white-list. *EURASIP Journal on Information Security*, 2016(1):9, 2016.

[35] Wikipedia. Mozilla. phishing protection, 2017. [Online; accessed 04-April-2017].

[36] Author: Lily Hay Newman Lily Hay Newman Security. Inside GoogleâĂŹs Global Campaign to Shut Down Phishing.

[37] Daisuke Miyamoto, Hiroaki Hazeyama, and Youki Kadobayashi. An evaluation of machine learning-based methods for detection of phishing sites. In *International Conference on Neural Information Processing*, pages 539–546. Springer, 2008.

[38] Maher Aburrous, M Alamgir Hossain, Keshav Dahal, and Fadi Thabtah. Predicting phishing websites using classification mining techniques with experimental case studies. In *Proc. of 7th Int. Conf. on Information Technology: New Generations (ITNG)*, pages 176–181. IEEE, 2010.

[39] Giovanni Armano, Samuel Marchal, and N Asokan. Real-time client-side phishing prevention add-on. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 777–778. IEEE, 2016.

[40] S. Afroz and R. Greenstadt. PhishZoo: Detecting Phishing Websites by Looking at Them. In *2011 IEEE Fifth International Conference on Semantic Computing*, pages 368–375, September 2011.

[41] Teh-Chung Chen, Scott Dick, and James Miller. Detecting Visually Similar Web Pages: Application to Phishing Detection. *ACM Trans. Internet Technol.*, 10(2):5:1–5:38, June 2010.

[42] A. Y. Fu, L. Wenyin, and X. Deng. Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD). *IEEE Transactions on Dependable and Secure Computing*, 3(4):301–311, October 2006.

[43] R. S. Rao and S. T. Ali. A Computer Vision Technique to Detect Phishing Attacks. In *2015 Fifth International Conference on Communication Systems and Network Technologies*, pages 596–601, April 2015.

[44] Neil Chou, Robert Ledesma, Yuka Teraguchi, John C Mitchell, et al. Client-side defense against web-based identity theft. In *NDSS*, 2004.

[45] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. An assessment of features related to phishing websites using an automated technique. In *Internet Technology And Secured Transactions, 2012 International Conference for*, pages 492–497. IEEE, 2012.

[46] Wikipedia. Tf-idf — Wikipedia, the free encyclopedia, 2016. [Online; accessed 20-July-2017].

[47] American National Corpus (ANC). The open american national corpus, 2017. [Online; accessed 20-July-2017].

[48] 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.

[49] Andrey Bernatsky. Python lib that provides you with some usefull and simple methods to obtain website metrics, 2017. [Online; accessed 20-July-2017].

[50] CSP. Content security policy (csp), 2017. [Online; accessed 20-July-2017].

[51] Mark A Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[52] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[53] Rokach Lior. *Data Mining with Decision Trees: Theory and Applications*, volume 81. World Scientific, 2014.

[54] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems (NIPS) 13*, pages 512–518, 2000.

[55] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[56] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proc. of the 11th Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[57] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Trans. on Systems, Man, and Cybernetics*, 6(4):325–327, 1976.

[58] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Trans. on Systems, Man, and Cybernetics*, 15(4):580–585, 1985.

[59] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6. IEEE, 2009.

[60] David Bremner, Erik Demaine, Jeff Erickson, John Iacono, Stefan Langerman, Pat Morin, and Godfried Toussaint. Output-sensitive algorithms for computing nearest-neighbour decision boundaries. *Discrete & Computational Geometry*, 33(4):593–604, 2005.