Understanding the Phishing Ecosystem

Sophie Le Page

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the M.Sc. degree in Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In "phishing attacks", phishing websites mimic trustworthy websites in order to steal sensitive information from end-users. Despite research by both academia and the industry focusing on development of anti-phishing detection techniques, phishing has increasingly become an online threat. Our inability to slow down phishing attacks shows that we need to go beyond detection and focus more on understanding the phishing ecosystem.

In this thesis, we contribute in three ways to understand the phishing ecosystem and to offer insight for future anti-phishing efforts. First, we provide a new and comparative study on the life cycle of phishing and malware attacks. Specifically, we use public click-through statistics of the Bitly URL shortening service to analyze the click-through rate and timespan of phishing and malware attacks before (and after) they were reported. We find that the efforts against phishing attacks are stronger than those against malware attacks. We also find phishing activity indicating that mitigation strategies are not taking down phishing websites fast enough.

Second, we develop a method that finds similarities between the DOMs of phishing attacks, since it is known that phishing attacks are variations of previous attacks. We find that existing methods do not capture the structure of the DOM, and question whether they are failing to catch some of the similar attacks. We accordingly evaluate the feasibility of applying Pawlik and Augsten's recent implementation of Tree Edit Distance (AP-TED) calculations as a way to compare DOMs and identify similar phishing attack instances. Our method agrees with existing ones that 94% of our phishing database are replicas. It also better discriminates the similarities, but at a higher computational cost. The high agreement between methods strengthens the understanding that most phishing attacks are variations, which affects future anti-phishing strategies.

Third, we develop a domain classifier exploiting the history and internet presence of a domain with machine learning techniques. It uses only publicly available information to determine whether a known phishing website is hosted on a legitimate but compromised domain, in which case the domain owner is also a victim, or whether the domain itself is maliciously registered. This is especially relevant due to the recent adoption of the General Data Protection Regulation (GDPR), which prevents certain registration information to be made publicly available. Our classifier achieves 94% accuracy on future malicious domains, while maintaining 88% and 92% accuracy on malicious and compromised datasets respectively from two other sources. Accurate domain classification offers insight with regard to different take-down strategies, and with regard to registrars' prevention of fraudulent registrations.

Acknowledgements

Thank you to my supervisor Guy-Vincent Jourdan for giving me guidance and continuous feedback, for setting high standards and challenging me, and for providing the ultimate Masters experience with travel. Thanks to Gregor v. Bochmann for his insights and intellectual honesty, and for encouraging me when giving presentations. Thanks also to Vio Onut for constant reminders that results should be summarized, and that motivations and contributions should be clearly understood.

I also thank my colleague Qian Cui for his open feedback and guidance, and for teaching me the importance of clearly explaining steps for reproducibility. Thanks also to my colleague Emad Badawi for helping with code reuse and for offering advice whenever asked.

Thanks to my partner Chris Briglio for supporting me the whole way through, for thinking that my work is cool, and for explaining to laypersons why it is cool. Last but certainly not least, thank you to my mother and grandparents for being my own personal cheer squad.

It truly is a team effort.

Dedication

This is dedicated to the end users of the Internet.

Table of Contents

List of Tables

List of Figures

(in alphabetical order)

- **AP-TED**: Augsten and Pawlik Tree Edit Distance.
- **API**: Application Program Interface.
- **APWG**: The Anti-Phishing Work Group.
- **ASN**: Autonomous System Number.
- **BSD**: Branded Subdomain.
- **DOM**: Document Object Model.
- **DNS**: Domain Name Server.
- **GDPR**: General Data Protection Regulation.
- **HTML**: Hypertext Markup Language.
- **HTTP**: Hypertext Transfer Protocol.
- **IOC**: Input/Output Controller.
- **PCA**: Principle Component Analysis.
- **PTED**: Proportional Tree Edit Distance.
- **QC**: Quality of Clustering.
- **SVM**: Support Vector Machines.
- **TED**: Tree Edit Distance.
- **TLD**: Top Level Domain.
- **TNR**: True Negative Rate.
- **TPR**: True Positive Rate.
- **URL**: Uniform Resource Locator.

Chapter 1

Introduction

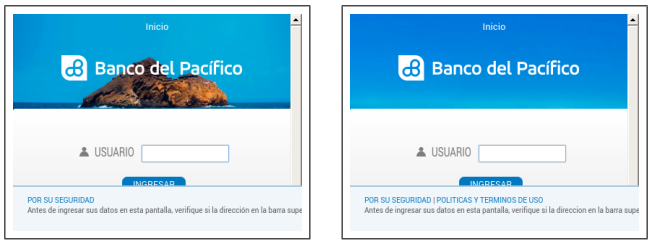
The Internet (short for interconnected network) provides people and companies with a new way of connecting. This has caused a shift in people's lifestyles, where for example online shopping and online education have begun to replace stores and schools. This has also caused a shift in companies who now focus on creating new Internet products and services in order to dominate the market. However, this shift gives criminals the opportunity to launch new types of crime using computers and networks, known as cybercrimes.

A phishing attack is a cybercrime that steals people's sensitive information by mimicking a legitimate website. Despite the tremendous efforts of academia and industry in recent years, anti-phishing research still faces many challenges. In this thesis, we look at the problem from a new perspective. Instead of focusing on detection techniques, we focus more on understanding the phishing ecosystem as a way to offer insight for future anti-phishing efforts.

In this chapter, we first define and discuss phishing attacks in Section 1.1. We then discuss the types of anti-phishing development in Section 1.2. In Section 1.3 we discuss the motivation of this research, followed by research challenges in Section 1.4. We summarize our contributions and organization of the thesis in Section 1.5 and Section 1.6 respectively.

1.1 Phishing Attacks

In "phishing attacks", phishing websites impersonate trustworthy websites in order to steal sensitive information from end-users, such as passwords and credit card details. Phishing attacks are typically carried out through URLs sent through emails, instant messaging, or social media platforms. These URLs typically redirect users to fake websites where sensitive information is requested. The process by which attackers deceive users in entering



(a) Legitimate site (b) Phishing site

Figure 1.1: Banking login page.

sensitive information is known as social engineering. For example, Figure 1.1 shows a phishing attack which mimics a bank login page. The attacker imitates every detail of the legitimate website, such as the logo, the favicon and even the external links, making it hard to visually distinguish it from the legitimate website.

Phishing attacks can typically be divided into two categories: generalized phishing (known as clone phishing) and targeted phishing (known as spear phishing). As the categories suggest, spear phishing attacks target a specific person or organization. To increase the probability of success and credibility, the attackers usually gather and use personal information of a person or members of the organization to lend legitimacy to their attack. For instance, an attacker may pretend to be an administrator and send phishing emails to members of the organization. Conversely, generalized phishing, as the main form of phishing attacks, does not have a specific attack target. To increase their probability of success and reach, the attackers use the same phishing attacks on large numbers of victims. Whereas spear phishing focuses on credibility to increase their probability of success, clone phishing focuses on increasing their reach. In order to make the phishing attacks more effective, attackers continuously improve the obfuscation technology to deceive victims and to evade anti-phishing efforts.

In recent years, although many anti-phishing products and solutions have been released, the threats of phishing attacks have been relentless. According to PhishMe's 2017 Enterprise Phishing Resiliency and Defense Report[1], a successful phishing attack on average costs a mid-sized company $1.6 million a year. Moreover, Wombat Security reports that in 2017, 76% of their surveyed companies have experienced spear phishing attacks[2]. In short, phishing attacks are a constant threat to individuals and the entire community.

[1] https://bit.ly/2XVZxSH
[2] https://bit.ly/2NmAbap

1.2 Anti-phishing development

We refer to phishing development as the efforts made by attackers to improve phishing attacks, while anti-phishing development are the efforts made by academia and industry to prevent phishing attacks. We group anti-phishing development efforts into three areas: detection techniques, understanding the phishing ecosystem, and education. Attempts to deal with phishing attacks through education include legislation, user training, and public awareness. While education is important, in this thesis we focus on comparing the efforts of detection techniques *vs* understanding the phishing ecosystem, making a point that more efforts should be put in understanding the phishing ecosystem. We give details of both these areas below.

1.2.1 Phishing Detection Techniques

Most of the literature on anti-phishing development focuses on automatic detection techniques, mainly in three directions. The first approach is to find similarities between a phishing site and the legitimate site that it mimics, such as performing visual comparisons between the phishing site and its target [28, 49]. The second approach is to try to find intrinsic characteristics of phishing sites, such as using machine learning to train a detection model to discriminate phishing sites and legitimate sites [38, 95], or by using machine learning to train a detection model to evaluate the reputation of the domains hosting the attacks [45]. The third approach compares the similarity with known attacks [29]: If a site is similar to a known attack, it is more likely to be a malicious site.

1.2.2 Understanding the Phishing Ecosystem

The literature on understanding the phishing ecosystem is much broader, and there are several different research directions which need continued efforts. These directions include case studies of phishing attacks within organizations and universities. They also include investigations into technical security measures since phishing attacks also exploit weakness in current web security. Lastly they include data analysis and empirical studies of the success, effectiveness and impact of phishing.

1.3 Motivations

A large focus of the efforts by both academia and the industry is on developing anti-phishing detection techniques. For example, browsers such as Google Chrome, FireFox,

Opera and Safari all use *Google Safe Browsing*[3] to provide their users with some level of built-in protection against phishing attacks. Microsoft Internet Explorer and Edge browsers also include a similar built-in defense mechanism, called *SmartScreen*[4]. In recent years, although many of these anti-phishing techniques have been released, the threats of phishing attacks have been relentless. The Anti-Phishing Work Group (APWG) reports over 280,000 unique attacks in the first quarter of 2016 [3], 144,000 in 2017 [5] and 260,000 in 2018 [9]. This indicates that we need to go beyond detection techniques and focus more on understanding the phishing ecosystem as a way to offer insight for future anti-phishing efforts. In this thesis, we provide suggestions and solutions for the following problems.

- **Lack of understanding cybercrime lifecycles.** Understanding phishing website removal times and number of visitors gives an idea of how well phishing detection techniques are working. Research has looked at phishing website removal times and at the number of visitors that the website attracts. However little research has looked at answering the same questions while comparing phishing to different cybercrimes, such as malware attacks.

  In this thesis we perform a comparative analysis of phishing and malware attack timespan and click-through traffic compared to its reported date. To do this we use the public analytics from shortened URL services. Our results show that anti-phishing efforts are stronger than anti-malware efforts, but that anti-phishing efforts are still too slow since the highest click-through occurs hours before the reported date.

- **Lack of understanding similarity between phishing attacks.** It has been found that phishers do not make brand new attacks from scratch. Instead, they create phishing variants based on existing attacks. Quickly identifying variants of known phishing attacks helps save time by concentrating efforts on identifying new phishing attacks. However no research has examined alternative methods to detecting variations in phishing attacks, and comparing it to the existing method.

  In this thesis we propose a new method which detects more phishing variants more precisely, and discuss the strengths and weaknesses of both methods. Our results show that no detection technique detects all variations, and confirms that the large majority of variations detected are agreed upon by both methods.

- **Lack of understanding how attackers commit their crimes.** Attackers hosting phishing websites either do so on a hacked (compromised) or owned (malicious)

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

[4] https://support.microsoft.com/en-us/help/17443/windows-internet-explorer-smartscreen-filter-faq

server. Determining whether a phishing attack is hosted on either server offers insight into how attackers commit their crimes and presents different remediation options. None of the current strategies offers an automatic method that weights criteria of both hacked and owned servers, while using public data that is widely available.

In this thesis we propose a domain classifier to classify known phishing attacks as being hosted on either hacked or owned servers. Our results show that domain presence and history are novel criteria that differentiate well compromised cases. Our results also show that the majority of phish feeds consist of compromised cases, where some feeds consisting of equally compromised and malicious cases.

1.4 Challenges

The current approaches in understanding the phishing ecosystem face a number of challenges in gaining insight into phishing attacks.

- **Hard to come by analytics.** The data needed to perform an analysis in understanding the phishing ecosystem has become increasingly hard to come by. For example, research that looks at phishing website removal times and number of visitors mostly makes use of resources such as log files, third party resources, and honeypots to gather data. However the last two resources are not always easy to get a hold of or to set up. As for the first resource option, log files are becoming scarce since people are instead using tools such as Google analytics to track their website click traffic. For academia and industry without enough resources, more efforts will need to be made to find alternative sources of analytics.

- **Hard to come by public information.** The recent adoption of General Data Protection Regulation (GDPR) prevents certain registration information to be made publicly available. Domain registration information has been used extensively in phishing detection techniques, and was a large topic of concern at the eCrime 2018 conference, the largest North American conference on phishing. Lack of insider knowledge of registration information will require more efforts by academia and industry to find alternative sources of information.

- **Hard to come by quality phishing datasets.** Good quality phishing attack datasets are important in any anti-phishing effort. However, public datasets such as PhishTank and OpenPhish often include several false positives. This requires a certain amount of manual checking to be performed. High quality phishing datasets

with low false positives and rich meta data are hard to come by unless working with industry. More efforts will need to be made in collaborating with industry to help richen an understanding of the phishing ecosystem.

1.5 Contributions

In this thesis, we contribute in three ways to understand the phishing ecosystem and to offer insight for future anti-phishing efforts.

- **An analysis of phishing attack click through and timespan.** In this thesis we present a new and comparative analysis of the life cycle of phishing and malware attacks using the public analytics of short URLs [58]. As far as we are aware, our work also constitutes the first analysis on the life cycle of malware attacks with respect to timespan and number of visitors, that is not limited to a single Enterprise or University. From our analysis, we find that phishing attacks are most active 4 hours before the reported date, while malware attacks are most active 4 days before the reported date. This concurs with other observations outside of shortening analytics which find that the timespan of phishing attacks has been reduced to hours. This shows that efforts against phishing attacks are stronger than efforts against malware, but that the majority of phishing victims are still reaching the attacks before the attacks are reported.

- **A comparison of methods to detect phishing attack variations.** In this thesis we evaluate the feasibility of applying Pawlik and Augsten's recent implementation of Tree Edit Distance (AP-TED [76]) calculations as a way to compare DOMs and identify similar phishing attack instances [55]. We also compare this tree method with an existing method that uses the distance between tag vectors to quantify similarity between phishing sites. Our results show that the tree method is more demanding for computing equipment, but it better discriminates the similarity with known attacks. Our results also show that there is still a high agreement between methods. This strengthens the understanding that most phishing attacks are variations of other attacks.

- **A domain classification of phishing attacks.** In this thesis we present a domain classifier to distinguish between phishing attacks hosted on hacked (compromised) or owned (malicious) servers [56]. Our solution uses criteria to detect both malicious and compromised cases from publicly available resources, making the solution widely

usable. We acquired high quality datasets in collaboration with industries such as the APWG and PhishLabs, and implemented a semi-supervised strategy to increase the size of our datasets by labeling APWG's unlabeled phish feed. Overall our classifier achieved above 90% accuracy on several of these datasets. Our results show that the majority of APWG's phish feeds consists of compromised cases, even though most efforts in the literature focus on detecting malicious domains.

1.5.1 Publications

We have published four papers out of this research:

- [58] Le Page, Sophie, Guy-Vincent Jourdan, Gregor V. Bochmann, Jason Flood, and Iosif-Viorel Onut. *Using URL shorteners to compare phishing and malware attacks*, in APWG Symposium on Electronic Crime Research (eCrime), 2018, pp. 1-13. IEEE, 2018.

- [55] Le Page, Sophie, Qian Cui, Guy-Vincent Jourdan, Gregor V. Bochmann, Jason Flood, and Iosif-Viorel Onut. *Using AP-TED to Detect Phishing Attack Variations*, in 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1-6. IEEE, 2018 (Short Paper).

- [59] Le Page, Sophie, Guy-Vincent Jourdan, Gregor V. Bochmann, Iosif-Viorel Onut, and Jason Flood. *Domain Classifier: Compromised Machines Versus Malicious Registrations*, in International Conference on Web Engineering, pp. 265-279. Springer, Cham, 2019.

- [57] Le Page, Sophie, Guy-Vincent Jourdan. *Victim or Attacker? Multi-dataset Domain Classification of Phishing Attacks*, in 2019 17th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2019.

An extended version of listed papers are included in the content of this thesis. For example, this thesis includes the entire content of [55] with additional detailed analysis of the phishing attack cases that were detected by one method and not the other, which was not included in the published paper due to page length restrictions.

The work done in the listed publications are the work, ideas and methods of the first author, while the other authors supervised the work. In the case of [55], the second author helped with the implementation of their method in order to compare it to the novel method proposed by the first author in the paper.

1.6 Organization

The rest of this thesis is structured as follows:

- In Chapter 2, we present a comparison of the current anti-phishing strategies in detection techniques *vs* understanding the phishing ecosystem, arguing for a stronger focus in the latter. We then give an overview of the existing strategies in understanding the phishing ecosystem, while analyzing the limitations and problems of these strategies.

- In Chapter 3, we discuss our method to collect click analytics of phishing and malware attacks. We then give an in-depth analysis on several of the click metrics, followed by interesting comparisons between types of attacks for each metric.

- In Chapter 4, we first introduce the current vector method for detecting similarities in phishing attacks. We then introduce a tree method to detect phishing variations. At the end of this chapter we present a comparison of the performance and accuracy of these two methods.

- In Chapter 5, we describe our machine-learning model for domain classification. In order to test the accuracy of our model, we discuss prediction results on several datasets. At the end of this chapter we present an analysis of the classification of several unlabeled phishing attacks.

Chapter 2

Literature Review

2.1 Introduction

In this thesis we claim that more focus should be put on understanding the phishing ecosystem. To give substance to this claim, we compare the current literature on anti-phishing detection techniques *vs* the literature around understanding the phishing ecosystem. Specifically, we show in the table below a list of phishing research published from 2016 onwards, from the top security conferences. On one hand, we find that there is a significant body of academic work focusing on the specifics of automatic detection of phishing attacks, such as detecting phishing websites, phishing emails, malicious domains, and spear phishing. On the other hand, we find that there is a similar amount of literature focusing on the understanding of the phishing ecosystem, even though this area of research has more ground to cover. Understanding the phishing ecosystem is larger than sole detection, and includes research areas such as case studies of phishing attacks within organizations and universities, investigations into technical security measures, and data analysis and empirical studies of the success, effectiveness and impact of phishing. For this reason, we argue that more effort is needed in understanding the phishing ecosystem.

In the remainder of this chapter we will be giving an overview of phishing detection in general, and then looking at three areas of understanding the phishing ecosystem: click through and timespan analysis, similarity methods, and domain classification.

2.2 Phishing Detection

Most published literature on phishing detection falls into one of the following categories: blacklists, comparing websites with targets, and looking at intrinsic characteristics.

Anti-Phishing Development	Type	Work	Description
Detection Techniques	Detect phishing websites	Bahnsen et al. [15]	RNN approach without feature creation.
		Asudeh et al. [13]	Multi-phase framework for visual similarity detection.
		Sonowal et al. [83]	Screen reader phishing detection for visually impaired.
		Cui et al. [29]	Method to detect phishing variations.
		Ardi et al. [12]	Target site phishing detection.
		Marchal et al. [66]	Designing and evaluating detection techniques.
	Detect phishing emails	Bogawar et al. [20]	Detect phishing emails based on sentiment.
		Tayal et al. [85]	Data mining techniques for phishing email detection.
	Detect malicious phishing domains	Liu et al. [61]	Detect shadow domains.
		Zuo et al. [103]	Hidden server malicious URL detection.
		Hao et al. [45]	Recognition of malicious domains at registration.
		Tian et al. [89]	Squatting domain builder and classifier.
		Elsayed et al. [34]	Detect IDN domain name, unicode characters.
	Detect spear phishing	Ho et al. [46]	Detect spear phishing attacks in enterprise setting.
Understanding the Phishing Ecosystem	Case studies	Holub et al. [47]	Tracking Ukrainian Bitcoin.
		Bidgoli et al. [16]	Undergrad student vulnerability to cybercrime.
	Technology investigation	Li et al. [60]	Understanding WebView cross-app attack.
		Aonzo et al. [7]	Shows phishing attacks are more practical due to Mobile Androids.
	Phishing kit analysis	Oest et al. [75]	Understand anti-phishing ecosystem through phishing kit analysis.
		Han et al. [43]	Understand phishing lifecycle through phishing kits.
	Data Analysis	Le Page et al. [58]	Click analysis of phishing life cycle.
		Wullink et al. [93]	Improve security through big data traffic analysis.
		Vargas et al. [90]	Exposing phishing patterns through data analysis.
		Kintis et al. [51]	Empirical study of domain combosquatting.
		Thomas et al. [88]	Understanding the risk of stolen credentials.
		Tajalizadehkhoob et al. [84]	Empirical study of hosting provider influence in avoiding web compromise.
		Zhang-Kennedy et al. [102]	Information visualization about internet phishing trends.
		Hu et al. [48]	Empirical study of spear phishing.

A blacklist is an access control mechanism which blocks the listed elements in order to warn users who are visiting a fraudulent website. To prevent phishing attacks, major browsers have integrated dynamic blacklists. For instance, Firefox, Safari, and Chrome use Google's blacklist, *Google Safe Browsing*, to protect users against phishing attacks.

In practice, blacklists do not prevent phishing attacks in real time, giving attackers a delay to spoof victims and collect stolen information. Sheng et al. [81] report that 47%-83% of phishing attacks were blacklisted after 12 hours. Furthermore, Han et al. [44] reported 62% of phishing attacks were blacklisted on average 20 days after installation. This delay may be caused by the compiling and publishing process.

Since the essence of phishing attacks is to mimic legitimate websites to trick the victims into entering their sensitive information, another phishing detection strategy is to compare the similarity between phishing sites and legitimate sites. These studies each explore different techniques and metrics to determine similarity, such as layout similarity, visual similarity, keywords similarity, and embedded links similarity.

For page layout similarity, Rosiello et al. [79] present a browser extension using the similarity of a DOM tree to detect phishing attacks. If the legitimate site is similar to a new site, the new site is reported as phishing. Zhang et al [100] suggest using the spatial layout features to compare the page layout similarity, to help catch pages with partially similar components. However both works suffer from the issue that it is hard to detect phishing websites that use background images to represent most of the components of the target site.

For page visual similarity, Dhamija et al. [31] propose a scheme to prevent attackers to mimic the site by embedding a visual identity in the page. However this method has limitations that make it difficult to apply in practice, such as requiring additional server-side modifications to support the verification, and requiring human intervention to complete the verification. Fu et al. [35] suggest a method and visual signature of the low resolution page screenshot to evaluate the visual similarity. Chen et al. [24] present a method to fetch the visual features of the page screenshot, applying a common compressor such as gzip or bzip2 to compress the screenshot, and use the Normalized Compression Distance (NCD) to measure similarity between compressed images. Other studies report good results by using only partial visual information as the detection identifier such as the favicon [37].

For page keyword similarity, Zhang et al. [101] propose a text-based method "Cantina", and extract keywords using the *Term Frequency and Inverse Document Frequency* (TF-IDF), which are submitted to the search engine to query pages containing similar keywords. Xiang et al. [96] improve "Cantina" by combining natural language processing techniques, and suggest extracting keywords from the page title and copyright text that are more representative of the target brand. Chang et al. [23] present a solution using the image segmentation technique to extract the site logo, and use Google Image Search to retrieve keywords related to the site logo.

For embedded link similarity, Liu et al. [62] suggest a graph-based method using embedded links to identify the phishing target. Given a suspicious page, they first construct a set of associated pages called "parasitic community", which consists of all pages reached by the embedded links (direct links) and the pages with the similar keywords obtained by querying the search engine (indirect links). A crawler is then sent to build connections for the parasitic community. The page with the strongest connection to the suspicious page is identified as the phishing target. Ramesh et al. [78] propose instead of retrieving the site with strongest connection to the suspicious page, to look at the intersection between direct and indirect links. Then, the IP addresses of the common links are compared with the suspicious page. If the suspicious page does not match any IP, it is flagged as phishing.

With the widespread use of machine learning techniques, some studies attempt to use machine learning to discover the intrinsic characteristics of phishing attacks. The goal of these approaches is to train a binary classifier by learning the relations between data features and the ground truth (phishing or legitimate). The prediction performance depends on two factors: feature selection and model selection. Overall works [**?**] and [95] have achieved the best two results (highest accuracy and lowest false postive rate when classifying phishing or legitimate sites) by applying SVM and Bayesian network. However, it does not mean that these two algorithms are better than others since the authors use different feature sets and datasets. Due to the lack of common datasets and feature sets,

it is hard to conclude which machine learning algorithm is most suitable for phishing page detection. Therefore, algorithms should be compared in the model selection process, but most studies lack this necessary step. Feature engineering also plays a crucial role in machine learning. Many studies attempt various features to improve the performance of phishing detection. These include URL, HTML, web, and security based features.

2.3 Understanding the Phishing Ecosystem

2.3.1 Click through and Timespan Analyses

In this section we look at research which tries to understand the impact of phishing and malware attacks through analytics such as the timespan of attacks, and the number of visitors an attack attracts. We then look at the research around short URLs. URL shortening services provide users with a smaller equivalent of any provided long URL. Unfortunately attackers abuse short URLs to mask the final destination where the victim will land after clicking on the malicious link. Initially, the research around short URLs analyzed the acceptance and use of short URLs over long URLs. This research then quickly evolved into analyzing the spam usage for short URLs. In turn this later evolved into the detection of and countermeasures for malicious short URLs.

2.3.2 Phishing and Malware Click through and Timespan Analysis

For phishing, in 2007 Ludl *et al.* [63] found that for phishing websites not yet blacklisted, the shortest addition to a blacklist was 9 minutes while the longest was 11 days. This was at a time where some blacklists could be bypassed if you simply added or removed a '/' to the end of a malicious URL, as shown in their results. In 2007 Moore *et al.* [69] found the mean lifetime of phishing attacks to be 58 hours. They found surprisingly, that the user responses continued at a fairly high level after the reported date, until the site is removed. They could not tell whether this was caused by ongoing spamming activity, or by users catching up with email backlogs in their inboxes. In 2008 McGrath *et al.* [67] showed that some phishing domains last for at least 3 days without being discovered by anti-phishing tools. In 2009 Sheng *et al.* [82] did an empirical analysis of blacklists and found that 63% of the phishing campaigns lasted less than two hours. They also found that blacklists were ineffective for protecting users initially, as most of them caught less than 20% of phishing attacks within the first hour. In 2014 Gupta *et al.* [42] analyzed anti-phishing landing

pages and found a forty six percent decrease in click through from trained users. In 2016 Han *et al.* [43] estimated the success rate of phishing kits by monitoring the activity of real visitors to infected honeypots, of which 9% submitted some data to the phishing page. In 2017 Cui *et al.* [29] analyzed replicas of phishing attacks as clusters, and found that around 80% of the clusters were active for less than one month. The long-lasting attacks within a cluster stayed active anywhere between 2 months to the entire 10 months of their observations.

For malware, in 2007 Provos *et al.* [77] presented the state of malware on the web and emphasized the importance of this rising threat of injecting malicious content on popular web sites. In part of their analysis they analyzed only the URLs with presence on the Internet that lasted longer than one week to determine how often the binary of the malwares changed. From their graph, some of these malware lasted over 20 months. However they did not comment on the portion of their data that is short-lived, and did not give a distribution of the timespan of the malware URLs. In 2014 Yen *et al.* [97] looked at malware in enterprises. Within that web-based malware, they looked at encounter rate based on job type and type of website. They found that websites deemed appropriate for business accounted for 31% of malware encounters.

2.3.3 Short URL Analysis

In 2008 McGrath *et al.* [67] found evidence that phishers were exploiting URL shortening services as far back as their dataset from 2006. Though they found the numbers not to be large, they warned of phishing continuing to abuse URL shortening services.

In 2010 Kandylas *et al.* [50] performed a comparative study of long URLs and short Bitly URLs on Twitter and found that Bitly short URLs received more clicks than an equal random set of long URLs. To further comprehend short URL distinctive characteristics, in 2011 Antoniades *et al.* [6] studied the lifetime of short URLs which revealed that the timespan of 50% short URLs exceeded 100 days. They also found that short URLs were mostly used on social networks and propagated through word of mouth, often pointing to news and informative content.

In 2010 Grier *et al.* [40] found that 8% of 25 million URLs posted to Twitter pointed to phishing, malware, and scams listed on popular blacklists. They found that Twitter is a highly successful platform for coercing users to visit spam pages, with a click-through rate of 0.13%. They found that blacklists are too slow at identifying new threats, allowing more than 90% of visitors to view a page before it became blacklisted. Around this time other studies started looking at malicious short URLs in emails and highlighted their privacy and

security implications [73, 74]. In 2011 Chhabra *et al.* [26] grabbed phishing attacks from PhishTank, and did a target and referrer analysis, focusing on referrer ties to Twitter. In 2012 Klien *et al.* [52] did a geographical analysis and presented the global usage pattern of short URLs by studying the usage logs of their own URL shortening service, and found 80% of short URL content to be spam related. This is possibly because their shortening service did not have spam countermeasures. Conversely in 2013 Maggi *et al.* [64] performed a large scale study on 25 million short URLs over a 2 year period, and found very few short URLs used for spam. This is possibly because they looked at it from a user perspective *vs* service perspective. Their results also highlight that the countermeasures adopted by these services to detect spam are not very effective and can be easily by-passed. Experimental results from their data shows that Bitly allows users to shorten malicious links.

Several research papers exist on the detection of malicious short URLs [92, 41, 72]. In 2013 Wang *et al.* [92], reported results showing that the majority of the clicks were from direct sources and that the attackers utilized popular websites to attract more attention by cross-posting the links. In 2013 Yoon *et al.* [98] proposed an alternative to short URLs by using relative words of target URLs, thus hinting about the target URL, making it then comparatively safe from phishing attacks. In 2014 Gupta *et al.* [41] performed an exploratory study on Bitly's spam URL and account detection mechanism to expose the gaps in security mechanisms. Lastly, what first got us started on the topic of URL shortening was a conference workshop [11] given in 2017. The workshop proposed to use shortened URLs as a representative sample that can be extended to the overall phishing population to measure the impact of attacks.

Problems with Click through and Timespan Analysis

There has been little work comparing the analytics of different types of attacks. The only other work from the list above that has done this is Grier *et al.* in 2010 [40], who looked at the blacklist evasion of scam, phishing and malware on Twitter. However, their use of Google Safebrowsing to identify phishing and malware, and Twitter's use of Google Safebrowsing API to filter links, biases their analysis. In addition their referrers only include Twitter, whereas analytics from short URLs include several other referrers such as Facebook.

2.3.4 Similarity Methods

In this section we look at research proposing similarity-based measurements in order to determine how often phishing attacks are variations of other phishing attacks. We first

introduce the motivation behind similarity based methods for detection, followed by an overview of the existing similarity methods.

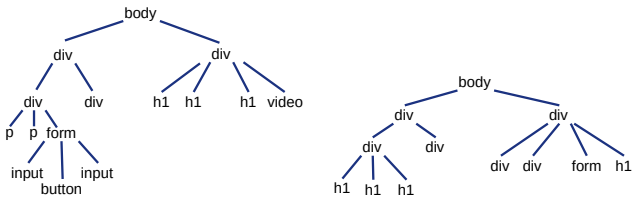
Similarity-Based Detection

It has been found that phishers reuse or modify previous phishing attacks [95], meaning that many phishing attacks are variations of other phishing attacks. Thus a form of phishing detection is to detect an attack that happens to be a replica or a variation of another already known attack. Such a strategy does not detect novel phishing sites, but rather new iterations of known ones. The collection of such phishing attack *instances* are referred to as a *phishing class*. Considering how an anti-phishing strategy works helps understanding the important role of detection of phishing classes in a phishing protection strategy. A typical anti-phishing strategy consists of a first step of fast filtering, flagging a number of sites as potential phishing sites. Then follows a final step of much slower analysis authoritatively discriminating whether previously flagged sites are indeed attacks or not. Phishing classes are known to be a trivial but abundant category among phishing sites. Following the first step, thus providing fast additional screening for phishing classes ahead of the final step. The modified strategy proposed here would quickly identify members of known phishing classes among the sites flagged by step 1. This intermediate step would provide fast positive identifications for upward of 90% of the attacks [29], thus freeing up resources for the last and slow step of in-depth analysis of sites ambiguously flagged at step 1.

From the authors work in [29] they further showed that phishing attacks tend to be relaunched many times, after sometimes small modifications. In [30], the authors further analyzed the details of the modifications and evolution over time. They show that phishing attacks tend to be derived from a small set of master versions. They also show that phishing attacks tend to go through only a couple of iterations over their lifespan and tend to evolve independently from one another, without much cross-coordination.

DOM Similarity

The Document Object Model (DOM) is a tree structure in which each node represents one HTML element of a web page. In previous research, a variety of techniques have been used to compare the similarity of DOMs [86]. New approaches to compare the similarity of DOMs have been introduced recently to compare phishing attacks to known ones. In 2011, Xiang *et al.* [95] propose a "hash-based method" which takes the HTML of the phishing site, removes the spaces and default values from it, and hashes the resulting HTML. Phishing



(a) DOM of page p_1 (b) DOM of page p_2

Figure 2.1: DOM trees, taken from [29]

.

sites with the same hash are considered similar. They found that 72.67% of their phishing site database were replicas from at least one other site in that same database. However this method is very strict and fails to match pages with small variations from each other. In 2017, Cui *et al.* [29] propose a "vector method" which takes the DOM of the phishing site, and creates a tag vector based on a list of tags. Using distance calculations between the vectors, they define a threshold which considers phishing sites to be similar. In this way, their vector method provides more flexibility in the definition of similarity. The vector method also uses criteria that are fast to compute, with time complexity $O(n)$. They report that 91.53% of their phishing sites are replicas of already known phishing sites, with a false positive rate of 0.08%, using 19,066 phishing sites and 24,800 legitimate sites. In the next section we further describe the vector method.

Vector Method

The "vector method" proposed by Cui *et al.* [29] in 2017, uses *tag vectors* to compare the similarity of the DOM of phishing attacks. A *tag vector* is based on an ordered list of 107 possible HTML tags, from which some of the more common tags have been removed, such as <body>, <head> and <html>. Note that their method only considers tags within the body of the DOM. The *tag vector* of a given DOM is a vector of size 107, and each element of the vector is the number of occurrences of the corresponding HTML tag in the DOM. To compare the distance between tag vectors, Cui *et al.* propose using the *Proportional Distance* (PD), which divides the Hamming Distance of the vectors by the number of tags that appear in at least one of the two DOMs.

Formally, for all integers x and y, the Hamming Distance is defined as $D(x, y) = 1$ if $x \neq y$ and $D(x, y) = 0$ otherwise. $L(x, y) = 1$ is defined if $x \neq 0$ OR $y \neq 0$ and $L(x, y) = 0$ otherwise. Let t_1 and t_2 be two non-null tag vectors over the same corpus of size n. The proportional distance $PD(t_1, t_2)$ between t_1 and t_2 as follows:

$$PD(t_1, t_2) = \frac{\sum_{i=1}^{n} D(t_1[i], t_2[i])}{\sum_{i=1}^{n} L(t_1[i], t_2[i])} \tag{2.1}$$

As an example, consider Figure 2.1 taken from [29], where two simple DOMs are provided. Assuming that the corpus consists only of the following nine html tags in this order:

`<form>, <b>, <p>, <h1>, <button>, <video>, <input>, <iframe>, <div>.`

The respective tag vectors for pages p_1 and p_2 are then:

```
<1, 0 ,2 ,3, 1, 1, 2, 0, 4>
<1, 0 ,0 ,4, 0, 0, 0, 0, 6>.
```

Seven of the nine possible tags of the corpus are used by at least one of the two vectors. Only the first one, <form>, appears the same number of times in both vectors, so $PD(p_1, p_2) = 6/7 = 0.86$. This indicates a large difference between the two pages.

However, the proportional distance PD as defined in [29] does not emphasize on the "amount" of differences between each HTML tag, and simply focuses on whether the number of tags is the same. For example, the vector $t_1 = \{1, 2, 5, 6\}$ and $t_2 = \{99, 2, 5, 6\}$ both have the same distance to the vector $t_3 = \{2, 2, 5, 6\}$, that is, $PD(t_1, t_3) = PD(t_2, t_3)$. To capture that t_2 is more different from t_3 than t_1 is, Cui et al. [30] define a new distance, called the *Weighted Proportional Distance*. Instead of using the Hamming Distance as the numerator, the sum of the *Weighted Differences* (WD) is used, defined by the following formula:

$$WD(t_1, t_2) = \sum_{i=1}^{n} \frac{|t_1[i] - t_2[i]|}{max(t_1[i], t_2[i])} \tag{2.2}$$

Whereas the value of PD for a given tag was boolean (0 or 1), for tags that are used in both vectors, WD will be in the range $[0, 1)$.The larger the difference between the number of tags, the larger WD.

S is defined as follows:

$$\begin{gathered} S(t_1, t_2) = \sum_{i=1}^{n} EQU(t_1[i], t_2[i]) \\ \text{where } EQU(t_1[i], t_2[i]) = \begin{cases} 1 & \text{if } t_1[i] = t_2[i] \text{ AND } t_1[i] \neq 0 \\ 0 & \text{otherwise} \end{cases} \end{gathered} \tag{2.3}$$

Finally, the *Weighted Proportional Distance* (WPD) is defined as follows:

$$WPD(t_1, t_2) = \frac{WD(t_1, t_2)}{WD(t_1, t_2) + S(t_1, t_2)} \tag{2.4}$$

Other distance metrics could be used with probably similar results, however the authors in [30] argue that WPD is fast to compute.

Problems with Similarity Methods

The problem with the above studies is that we have no basis for comparison between other similarity methods in order to confirm whether the existing method is efficient and accurate. For example, since the vector method does not capture the structure of the DOM, one can question whether the method is failing to catch some of the similar attacks. Moreover, this method is brittle in that small changes to the HTML DOM can create very different vectors.

2.3.5 Domain Classification

In this section, we look at research trying to determine how attackers commit their crimes based on whether they hacked a domain to host a phishing site, or whether they created their own domain. We first discuss different types of hosting for phishing websites, followed by discussing the work related to identifying various types of hosting for phishing websites.

Types of Hosting for Phishing

The types of hosting for phishing websites are identified by Moore *et al.* [71] as free web-hosting services, compromised machines, rock phish and fast-flux attacks. In [71] the authors analyze the different "notice and take-down" strategies for each case.

As an example, a typical URL for a website that has been set up at a free web-hosting provider would be http://www.brand.freehostsite.com/login, where the brand name is chosen to match or closely resemble the domain name of the brand being attacked. It is usually sufficient to compile a list of known free web-hosting domains, and then use this list to determine which websites are hosted on free space. In this case, to get the phishing website removed it is necessary to contact the webspace provider and draw their attention to the fraudulent site.

For compromised machines, attackers may have restricted permissions, and are limited on where files can be placed. They add their own web pages within an existing

structure, leading to URLs for their websites that have the typical form http://www.example.com/user/www.brand.com/ where the brand name lends legitimacy. The attacker may also find that the existing DNS configuration permits URLs of the form www.brand.com.example.com. In this case, in order to get a website removed from a compromised machine it is generally necessary to get in touch with the sysadmin who looks after it.

To further hide suspicion, attackers sometimes go through the effort of registering their own domain name. The domain names are usually chosen to be a variation of brand.com such as "brand-usa.com", or they will use the brand name as a subdomain of a misleading domain such as "brand.verysecuresite.com". We refer to these domains as maliciously registered. If a domain name has been registered by an attacker, the defenders will ask the domain-name registrar to suspend the offending domain.

Rock phish and fast-flux attacks require the attackers to purchase a number of cheap or free domains with meaningless names such as "vbe10.info", with unique identifiers in order to evade spam filters. We also consider these domains to be maliciously registered domains.

In this way we distinguish three types of hosting for phishing websites: free web-hosting domains, compromised domains, and malicious domains. Since free web-hosting identification simply requires a list of known hosting websites, we are left with the problem of classifying between compromised and malicious domains.

Identifying Types of Hosting for Phishing

In 2009, Moore *et al.* [70] worked on so-called "evil searching" of servers running known vulnerable software to compromise them and upload phishing sites. The authors mentioned that 75.8% of their database is made of compromised servers with no explanation about how they reached this number. More recently, in 2017 Corona *et al.* [28] proposed a method to detect phishing sites by evaluating the visual differences between the phishing page and the other pages hosted on the same domain. The authors suggest that 71% of the domains hosting phishing sites are compromised. However the authors used manual checking and did not provide any reusable method.

In 2016, Catakoglu *et al.* [22] use honey pots to lure attackers to compromise their server, and propose an automated technique to extract and validate indicators of compromise (IOCs) for web applications. In 2016, Hao *et al.* [45] detected malicious domains upon registration, for the purpose of phishing as well as spamming. Their strategy also uses machine learning in combination with designed features derived primarily from infor-

mation known by registrars or registries, as well as lexical patterns of the domain name. Our approach includes most of the lexical pattern features from [45]. In 2017, Lin *et al.* [61] detected domain shadowing, which are compromised domains with malicious subdomains. The authors find that instead of generating subdomain names, several domain shadowing cases exploit the wildcard DNS records.

A 2016 [4] study done by The Anti-Phishing Work Group (APWG) reports on phishing trends for malicious and compromised domains. For phishing attacks launched only in 2016, APWG report that almost 49% are malicious domains, while the rest are compromised. Their strategy is to identify malicious domains by checking 1) short timeframe from domain registration to phishing report, 2) brand name or misleading string in the domain names and 3) batch domain names registration.

Problems with Domain Classification

One common problem with the above studies is that their strategies focus on criteria that indicate malicious domain cases and otherwise simply considers that the domain is compromised. These type of one-sided criteria may be too aggressive and will for example classify any site that is hacked almost immediately upon creation as malicious. Second, their solution relies upon registration information which is not publicly available. This is especially relevant due to the recent adoption of General Data Protection Regulation (GDPR), which prevents certain registration information to be made publicly available. For example, registration information which can be used to determine which domains relate to a person are no longer publicly available, such as name, email or phone number.

2.4 Conclusion

In this chapter, we compared the amount of research done in phishing detection techniques *vs* the research looking at understanding the phishing ecosystem overall, pointing out that there should be more focus on the latter. We then give an overview of three areas in understanding the phishing ecosystem: phishing click through and timespan analysis, phishing similarity methods, and phishing domain classification. We discuss these areas in detail and their limitations: the analysis of phishing click through and timespan lifecycle has not been compared to other types of attacks. With regard to similarity methods to detect phishing variations, there has been no alternative method proposed to compare efficiency and accuracy tradeoffs. Lastly the solution of classifying phishing websites as being hosted on either a compromised or malicious domain does not balance criteria for

compromised cases and does not use publicly available information for a widely usable solution.

Our research goal is to fill these gaps. We perform a comparative analysis of phishing and malware attacks using the public analytics of URL shortening services which provide richer data with regard to HTTP referrers and country referrers. We also present a novel similarity method using the tree edit distance as a way to compare efficiency and accuracy tradeoffs against other methods. Lastly we propose a domain classifier balancing both compromised and malicious criteria using only publicly available information. We will discuss the details of these approaches in the rest of this thesis.

Chapter 3

An Analysis of Phishing Attack Click through and Timespan

3.1 Introduction

In this chapter, we try to determine how successful phishing attacks are, and to compare phishing attack success with malware attacks. We define success based on the number of victims who click on a phishing link and based on the length of time a phishing link is active.

Specifically, we also look at what URL shortening analytics can tell us about the life cycle of phishing and malware attacks. We start by introducing URL shortening services and why services such as Bitly provide public analytics (Section 3.2). We then discuss how we identify malicious Bitly URLs that lead to phishing and malware attacks and the type of analytics we fetch once a malicious Bitly URL is identified (Section 3.3). We then show our analysis of the malicious Bitly URL analytics, including our analysis of flagged short URL clicks (Section 3.4.1), click through (Section 3.4.2), click timespan (Section 3.4.3), click distribution by reported date (Section 3.4.4), HTTP referrer clicks (Section 3.4.5), and country referrer clicks (Section 3.4.6), before concluding at the end of this chapter.

3.2 URL Shortening Services

The concept behind URL shortening services is to assist in sharing URLs by providing a short equivalent. URL shortening services provide users with a smaller equivalent of any provided long URL, and redirects to the corresponding long URL by the service provider

through a "HTTP 301 Moved Permanently" response. Shortened URLs first appeared in 2001 [68]. Initially, the concept was used to prevent breaking of complex URLs while copying text, and to prevent email clients from inserting line breaks in the links which rendered them unclickable. However, its adoption was slow until they became popular in online social networks. Now URL shorteners are almost a requirement due to character limitations in some social media, and due to mobile devices, where space is always at a premium.

Over the years URL shorteners such as Bitly have evolved to provide increased utility to users, such as providing analytics to track clicks. Bitly is one of the most popular URL shortening services, as a few studies have found [73], [92]. Each short URL Bitly issues is unique and will not be re-used, so the Bitly short URL will always direct to the same long URL. When a registered user shortens the same long URL, each instance gets a unique short URL. This way users can keep track of their own click analytics. A long URL may have many short URLs, shortened by different registered users, but an aggregate shortened URL keeps cumulative count of statistics for every click on the long URL through Bitly (see Figure 3.1).

A short URL is uniquely identified by what Bitly calls a hash. For example, if the following URL is submitted to Bitly "https://www.theweathernetwork.com/ca/hourly-weather-forecast/ontario/ottawa", the corresponding short URL is "http://bit.ly/2lFK4BS" which consists of Bitly's default domain name "bit.ly", and the hash "2lFK4BS" as the backhalf. A hash only contains the characters "a-z,A-Z,0-9", and is a simple iteration across every permutation of the available characters as the URLs come in [27]. For example, given a limit of three character representations, each new URL gets a unique three character combination until no more unique combinations are left, at which point the limit is increased to four character representations.

Users can also choose to create a customized backhalf which is easier to remember than random letters and numbers, but the backhalf is only accepted if it has not already been used. A customized backhalf can further contain characters "-" and "_" [19]. For example, the short URL "bit.ly/SuperBowl" is a customized backhalf. Further customization of short URLs include Branded Short Domain (BSD), such as "nyti.ms/SuperBowl", where "nyti.ms" is the BSD for New York Times, which allows large organizations to maintain their brand identity while using URL shorteners.

Unfortunately URL shortening services provide attackers with a convenient and free tool to obfuscate their URLs. Through the use of customized backhalfs, attackers can even craft a short URL so as to fit the target's profile (e.g., bit.ly/freemoviesfast). In the case of blacklisting malicious short URLs, for blacklists based only on domains rather than

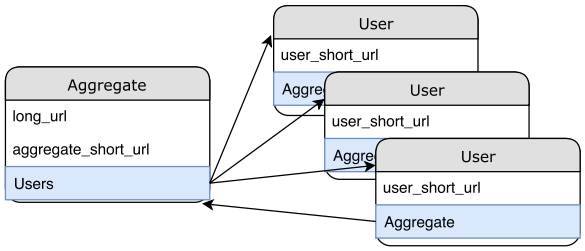


Figure 3.1: Bitly mapping of long URL, user short URL and aggregate short URL. A long URL may have many short URLs, shortened by different registered users, but the aggregate shortened URL keeps cumulative count of statistics for every click on the long URL.

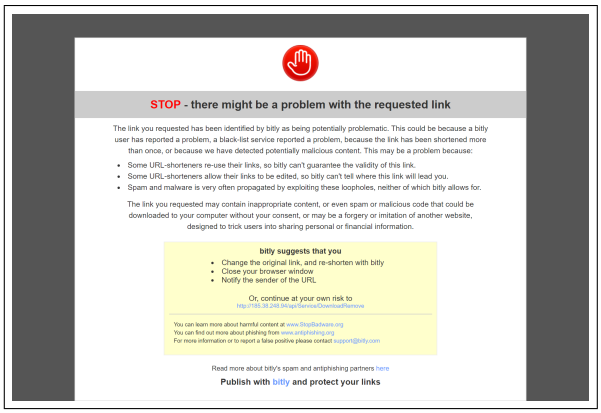


Figure 3.2: Bitly warning page.

full URLs, false positives pose a threat of blacklisting entire sites such as URL shorteners. This resulted in blacklists having to use crawling in order to resolve shortened URLs, and blacklist the long URL. Nikiforakis *et al.* [74] also give a good overview of other dangers of URL shortening, such as linkrot and hijacking.

Pressure was also put on URL shortening services to put in place countermeasures for spam. For example as shown in Figure 3.2, Bitly flags malicious short URLs and displays a warning page upon clicking the malicious short URL. Bitly also provides a "preview" of a short URL by allowing users to append a '+' to the end of a short URL when entering it into the browser, as shown in Figure 3.3. This preview shows the long URL as well as detailed analytics such as number of clicks and percentage of referrer and country clicks. Note that the preview shows a brief overview of the analytics, and in order to see more details such as hourly number of clicks, one needs to use the Bitly API. It is through these public analytics that we propose to compare the life cycle of phishing and malware URLs.

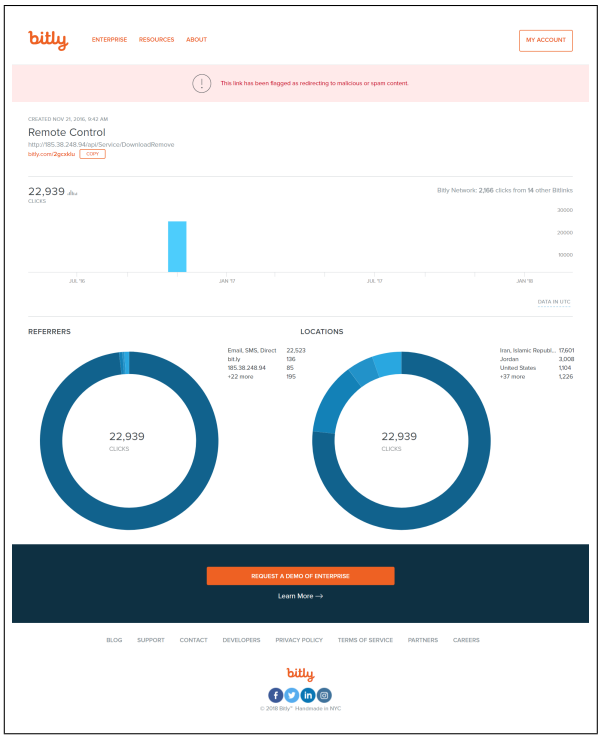


Figure 3.3: Bitly short URL preview page.

3.3 Collecting Malicious Short URL Data

To collect malicious short URLs we use three steps. The first step is to collect malicious URLs from each source, the second step is to identify short URLs from the malicious URLs, and the third step is to fetch Bitly analytics on the short URLs.

3.3.1 Collecting Malicious URLs

To collect malicious URLs, we looked for three things from our sources: a large database of verified malicious URLs, malicious URLs categorized as phishing or malware, and a report date. With these requirements in mind, we use the following sources to collect malicious URLs:

PhishTank

A community-driven portal. PhishTank deals with phishing reports verified by users, so all malicious URLs from this source are for category phishing. For PhishTank, we consider the submission date of a malicious URL as the report date.

We have been collecting phishing URLs from PhishTank daily since January 1st, 2016. PhishTank is a community-based phish verification system where users submit suspected phishes and other users vote if it is a phish or not. PhishTank uses an adaptive cut-off for number of votes required for a submitted URL to be declared verified. When collecting phishing URLs from PhishTank, we filter those URLs that are verified, and manually verify some of them ourselves, since PhishTank's voting system can sometimes lead to false positives. From the verified URLs, we remove those URLs that are not reachable or are blocked, such as URLs that return a 404 Not Found status code. This is because in our previous research, we were particularly interested in retrieving the DOM of the landing page (final URL). More about our PhishTank dataset and methodology can be found in [29]. From January 1st, 2016 to December 31st, 2017, our PhishTank dataset consists of 51,516 unique phishing URLs.

X-Force

An IBM enterprise security analysis platform. X-Force maintains reports for verified malicious URLs from several categories, and allows querying for specific date ranges for URLs. For X-Force, we consider the created date of a malicious URL as the report date.

We queried using the X-Force endpoint "Get URLs by Category" for categories phishing and malware from January 1st 2016 to December 31st 2017. This resulted in 247,105 unique phishing URLs and 72,681 unique malware URLs in our X-Force dataset.

3.3.2 Identifying Bitly Short and Long URLs

Given a list of malicious URLs, we describe the methods we used to identify those that are Bitly short URLs and those that are long URLs which have been shortened using Bitly's services. From a Bitly long URL, one can recover the corresponding short URL.

Identify Bitly Short URLs

Bitly Domain Name: Bitly's default domain name is "bit.ly". Other Bitly domain names are "bitly.com" and "j.mp". The domain name "j.mp" is a domain that Bitly offers to users who prefer a shorter domain name. This method simply checks if a given domain name is equal to either of the three domains mentioned above. Although this method does not identify branded short URLs, this is the easiest and fastest method to identify the majority of Bitly short URLs from a large list of URLs, since the method only requires string-matching operations.

Bitly Branded Short Domain: When shortening a URL, users can create a Branded Short Domain (BSD) that takes the place of "bit.ly". Since 2011, adding a BSD to a Bitly account is free, although a domain from a third party domain registrar must be purchased and linked to Bitly [33]. This method makes use of Bitly API endpoint "Pro domain" to query whether a given domain is a valid BSD. This method is relatively scalable for a large number of requests, and provides more coverage when used along with identifying by *Identify Bitly Short URLs*. Note that a BSD must be less than 15 characters, including the dot, so we check the length of the domain before checking whether it is a BSD [18].

Bitly ASN: An Autonomous System Number (ASN) uniquely identifies the organization that is responsible for a block of IP addresses. This method identifies Bitly short URLs by checking whether a given domain is part of Bitly's ASN, which is "395224". This method will identify all Bitly domain names, including the branded ones. However checking the ASN of a large list of URLs is usually only feasible with third party resources.

Identify Bitly Long URLs

Bitly URL Lookup: Since it is problematic to blacklist the URL shortening service itself, some sources do not blacklist Bitly short URLs but only the redirected URL (long URL). For this reason it is necessary to do a URL lookup to check whether a malicious URL has been shortened, and to record the corresponding short URL. This method identifies a Bitly long URL by checking the Bitly API "LookUp" endpoint, which returns the aggregate shortened URL (http://bit.ly/[aggregate_hash]) for a given long URL if there is any. The aggregate shortened URL keeps cumulative count of statistics for every click on the long URL through Bitly.

3.3.3 Determining Unique URLs

We define a unique URL (e.g. http://example.com/path/) by removing duplicates that only add '/' to the end of the URL. In addition, URLs that only add '/' are considered the same URL when shortened using Bitly. There are other cases that Bitly considers as the same URL when shortened, however we do not take into consideration these cases. For example, a URL that begins with "http://www." and one that begins with just "www." are considered to be the same by Bitly when shortened.

3.3.4 Short URL Identification Results

PhishTank

From our PhishTank dataset, we first tried to identify short URLs by *Identify Bitly Short URLs*, but only found 61 URLs in this way. This is probably because we removed submissions with blocked final URLs, and since Bitly short URLs were often blocked by the time we crawled the URL, they were not included in our PhishTank dataset. As a result we searched by *Identify Bitly Short URLs* directly through PhishTank and identified 1,048 short URLs, 8 of which used a branded domain. Returning to our PhishTank dataset, we used *Identify Bitly Long URLs* and identified 1,275 long URLs, recording the corresponding aggregate short URL for each. At the end of this process, we identified 2,207 malicious short URLs from PhishTank for category phishing.

X-Force

From our X-Force dataset, we first tried to identify short URLs by *Identify Bitly Short URLs*, but only found 4 short URLs for phishing and 1 short URL for malware. We concluded that X-Force avoids blacklisting Bitly short URLs and only blacklists the redirected URL (long URL). Next we used *Identify Bitly Long URLs* to identify Bitly long URLs and recorded the corresponding aggregate short URL. Note that X-Force returns malicious URLs without HTTP, so we tried to attach both "http://" and "https://" before looking up the URL. At the end of this process, we identified 5,855 malicious short URLs from X-Force, 2,532 for category phishing and 3,363 for category malware.

3.3.5 Fetching Short URL Analytics

As mentioned in Section 3.2, Bitly maintains metrics for each created short URL. After we identified Bitly short URLs from our collected list of malicious URLs, we fetched the following Bitly metrics for each of these short URLs:

- *Clicks*: Returns the number of clicks on a single short URL. Data can be as detailed as hourly number of clicks.
- *Countries*: Returns the number of clicks for each country referring click traffic to a single short URL.
- *Referrers*: Returns the number of clicks for each HTTP referrer referring click traffic to a single short URL. Even the click count for each full referrer URL is available.

- *Information*: Returns the date the short URL was created, as well as a reference to the aggregate short URL.
- *Expand*: Returns the long URL, as well as a reference to the aggregate short URL.
- *Encoders*: Returns the number of users who have shortened a single long URL.

For the sake of completeness, in our analysis we use metrics for the aggregate short URL, which aggregates the metrics from each registered user who shortened the same long URL. Analyzing the aggregate short URLs, we found that over 90% of short URLs were shortened by only one user, *vs* several registered users. This shows that analyzing the aggregate short URLs is effectively the same as analyzing the user short URLs.

Note that for this work, the last day we fetched Bitly metrics was on February 10th 2018.

3.4 Analysis and Results

In this section we perform an analysis on our previously discussed dataset. We start our analysis by looking at whether clicks are recorded for short URLs flagged by Bitly. With this understanding, we then analyze phishing and malware click through, click timespan, click distribution against reported date, and click sources from HTTP referrers and countries.

3.4.1 Flagged Short URL Clicks

Bitly short URL clicks are counted in near real time. One can simply test this by navigating to a short URL, and then previewing the URL by appending '+' to the end of it, to see the click count increased by one. However we found that clicks are not counted for short URLs that Bitly has flagged as malicious, which means that click analysis cannot be continued once a URL has been flagged. For example research has shown that after a URL has been blacklisted, visits continue to be logged up until the URL is taken down [69]. However based on our findings, this type of analysis would not be observed using short URLs, since clicks are no longer recorded when a short URL is flagged and a warning page is shown, even if a victim continues through the warning page to the malicious site. Therefore this should be taken into consideration when doing click analysis for malicious short URLs.

To check whether a click is counted for flagged short URLs, we gathered flagged short URLs by searching for the string "This link has been flagged as redirecting to malicious or

spam content" in Google. This search results in a few hundred flagged Bitly short URLs. We then entered these flagged URLs into the web browser, and after continuing through the warning page to the malicious site, we found that our clicks were not counted. It is unclear why clicks are not counted for flagged short URLs. We emailed Bitly to ask for clarifications, but did not receive a reply.

To verify our findings, we grabbed from our dataset the short URLs that were submitted on PhishTank as "warning" links along with their submission date. For example some of the short URLs were submitted as follows:

```
https://bitly.com/a/warning?hash=1WT7pjU...
```

Knowing when a short URL is submitted as a warning link gives us an indication as to when a short URL was flagged. Next we checked the short URL's latest click date against the submission date. If the clicks of a flagged short URL are no longer recorded, then we should observe that there are no clicks after the submission date.

We only have 16 of these "warning" short URLs in our dataset, which were almost all submitted by Veriform [91] to PhishTank. Of the 16 short URLs, 14 are still flagged by Bitly. Of these 14 URLs, 7 have their latest click before the submission date, which follows our understanding that flagged short URLs no longer record clicks. However the other 7 URLs have their latest click after the submission date. Taking a closer look at these 7 short URLs, we find a clear indication of the clicks stopping before the submission date, but then reappearing momentarily after a few more days, before stopping completely.

An example of this case is given in Figure 3.4, of a short URL "bit.ly/1WT6ZtL" which was created on June 18th 2016 at 9:20 AM, and was submitted on PhishTank as a warning link on June 20th 2016 at 7:57 PM. The graph shows the number of clicks every hour, and we see the clicks stop on June 20th 2016 at 4:00 PM, before the PhishTank submission date. This indicates that the short URL was flagged by Bitly on June 20th at 4:00 PM, at which point clicks are no longer counted. However, we see the clicks continue about 3 days later until stopping on June 25th 2016. After this point no new clicks have been recorded. This indicates that the short URL was flagged, then somehow unflagged by Bitly, before being flagged a final time.

We further manually checked these 7 cases exhibiting this behavior and found 5 of the cases were on compromised domains, hosted on a 3D printing shop, a robotics equipment shop, a kite surfing site, a Christian community radio site, and a travel site. Attacks hosted on a compromised domain would explain why the short URL might have been unflagged momentarily.

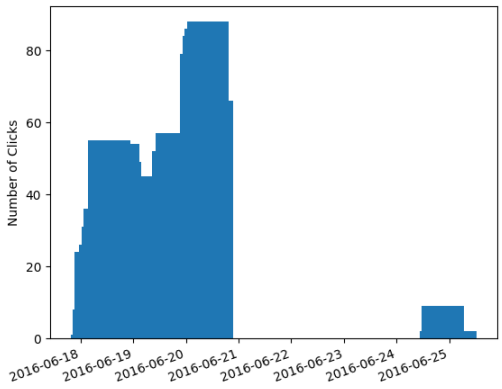


Figure 3.4: Hourly clicks of a short URL flagged by Bitly.

Table 3.1: Number of Short URLs with Click Through.

	Phishing		**Malware**	
	Size	Percent	Size	Percent
Number of unique short URLs	4,324	100.00%	3,363	100.00%
... with clicks	3,535	81.75%	1,042	30.98%
... ... within 1 year of report	3,425	79.21%	520	15.46%
... ... within 3 months of report	3,259	75.37%	313	9.31%
... ... within 50 days of report	3,151	72.87%	264	7.85%
... ... within 48 hours of report	2,346	54.25%	79	2.35%

As far as we know, none of the previous research on malicious short URL analysis has mentioned clicks no longer being counted for flagged URLs. This may be because Bitly might have brought changes regarding clicks being counted. Based on our analysis we find that within the past two years, Bitly does not record new clicks for flagged short URLs.

3.4.2 Click Through

From our 7,647 collected short URLs, only 51.3% generated click traffic within 1 year of reported date, accumulating to over 11.8 million clicks. Of the short URLs generating click traffic, we find that 10% of URLs make up 90% of the accumulated clicks. This shows that malicious URLs generate heavy traffic using URL shortening services and that only a few URLs make up the majority of clicks.

Of the short URLs not generating any clicks, most of these are malware URLs, as can be seen in Table 3.1, where numbers drop by two thirds when considering click traffic. This may be because some of these short URLs may not have been used in actual attacks. Therefore these short URLs do not necessarily need clicks to be reported since it suffices that the long URL be reported. To confirm this, if we look at only the short URLs reported on PhishTank, we find that 99.98% of these URLs have click traffic. This indicates that when short URLs are reported they are more likely to have been clicked on, *vs* when the long URL is reported.

Table 3.1 shows the number of short URLs as we restrict click-through traffic closer to the reported date. When analyzing URL clicks, we restrict our analysis to short URLs with clicks within at least 1 year of reported date since some of the URLs were created as far back as 2009, and do not have any recent click traffic. We consider clicks only occurring years before the reported date are not relevant in our analysis. Looking at Table 3.1, we notice that when we restrict URLs with clicks within 48 hours, 54% of phishing URLs are active while 46% are inactive, and 2% of malware URLs are active while 98% are inactive. The large percentage of inactive URLs suggests that our sources are late at reporting the attacks.

Looking at Figure 3.5, of the URLs that generate any traffic within 1 year of report, 20% of the phishing URLs and 60% of malware URLs receive fewer than 27 clicks. This shows that malware URLs have less click activity than phishing URLs. This may be because it has been found that phishing attacks work well at getting victims to click on their links [32] [2].

Looking at previous research, Grier *et al.* [40] reported that 2.3% of malicious short URLs have click-through traffic, which is much lower than our 51.3%. This may be because their dataset consists of malicious short URLs from Twitter over a period of 3 months, in which 95% of their dataset were scam attacks, and only 5% were phishing and malware attacks. To take this into consideration, of our short URLs generating any traffic within 3 months of reported date, 5.04% of these URLs have clicks from Twitter. This indicates that there is a similar number of malicious short URLs being clicked on from Twitter over the years. Grier *et al.* [40] also showed that 50% of URLs received less than 10 clicks. If we consider only short URLs generating any traffic within 3 months of reported date, looking only at clicks from Twitter, 50% of our URLs receive fewer than 13 clicks. This indicates that malicious short URLs from Twitter have been receiving about the same number of clicks over the years. More results about Twitter and other referrers for phishing and malware URLs is analyzed in Section 3.4.5.

Overall from our click-through analysis, we find that phishing receives more click-

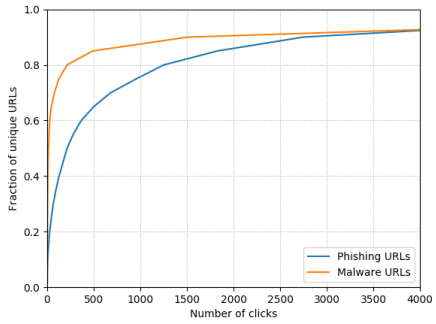


Figure 3.5: Click through for phishing and malware URLs. Only the 79% of phishing URLs and 15% of malware URLs generating any traffic within 1 year of report are shown.

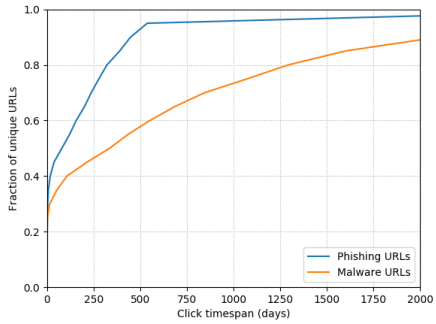


Figure 3.6: Click timespan in days from first to last click for phishing and malware URLs. Only the 79% of phishing URLs and 15% of malware URLs generating any traffic within 1 year of report are shown.

through than malware.

3.4.3 Click Timespan

Here we consider the timespan of clicks for each short URL, where timespan is measured from first to last click. We restrict our analysis to short URLs that have clicks within at least 1 year of reported date. Following our findings in Figure 3.6, we see that 50% of phishing URLs last less than 80 days, while 50% of malware URLs last fewer than 340 days, just under 1 year. This shows that malware URLs have a longer timespan than phishing URLs.

We also notice that as the graph continues, the difference between the click timespan of phishing and malware URLs increases. At the extremes, 90% of phishing URLs last up to 1 year while 90% of malware URLs last less than 5 years. This shows that the timespan of

the upper 50% of malware URLs is several years, while that of the upper 50% of phishing URLs is between 3 months and a year. As we look at the portion of the graph beyond 95% of phishing URLs, we notice a horizontal line in click timespan. This line is due to our dataset not having any phishing URLs with a timespan between 500 and 2,000 days, which again highlights the difference in timespan between malware and phishing URLs.

As far as we know, no previous research has looked at the timespan of malicious short URLs. In 2011 Antoniades *et al.* [6] studied the timespan of benign short URLs which revealed that 50% of short URLs exceeded 100 days. This shows that benign short URLs are active longer than phishing URLs, but that malware URLs are actually active three times longer than benign short URLs. Overall from our click timespan analysis, we find that malware lasts for longer timespan than phishing.

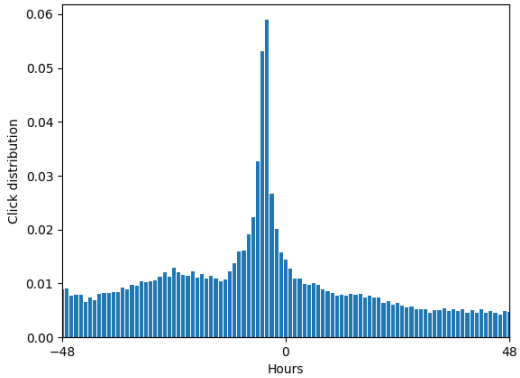
3.4.4 Click Distribution by Reported Date

We want to look at the click distribution of phishing and malware URLs before and after the reported date. To do this we aggregated the click data for every short URL. The most detailed we can aggregate our clicks is hourly, using Bitly's API. For URLs reported multiple times, we consider each report as a unique, independent event. For example, if we want to aggregate our clicks by hour, we create bins relative to each hour before and after a reported date. Then for each short URL, we normalize the number of clicks so that the hour with the maximum clicks has a weight of 1. This is to avoid short URLs with exceptionally large number of clicks to skew the results. Next we add these normalized values to their respective hour bins. Once we have aggregated all short URLs into bins, we normalize the bins so that their sum is equal to 1. Figure 3.7 shows the result of our analysis, looking at the distribution of clicks encountered 48 hours, 50 days and 1 year before and after the reported date. The number of URLs for each time frame can be found in Table 3.1.

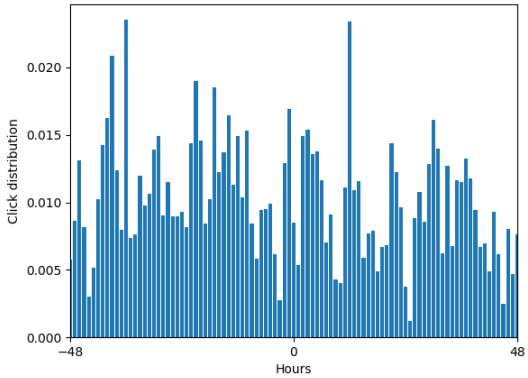
Looking at Figures 3.7 (a) and (b), we find a peak in the distribution of clicks for phishing about 4 hours before the reported date. However for malware we find that within 48 hours there is no clear peak in the distribution.

Looking at Figures 3.7 (c) and (d), as expected from the previous figure, we find a peak in the distribution of clicks for phishing on the day of the reported date. For malware we now see a more clear trend where the clicks peak about 4 days before the reported date, and then decline.

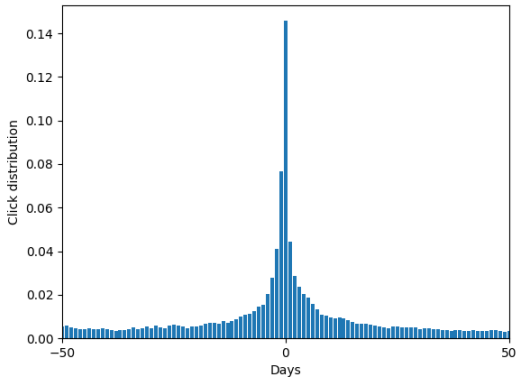
Looking at Figures 3.7 (e) and (f), here we see a bigger picture of the click distribution over a full year on either side of the reported date. Again as expected from the previous



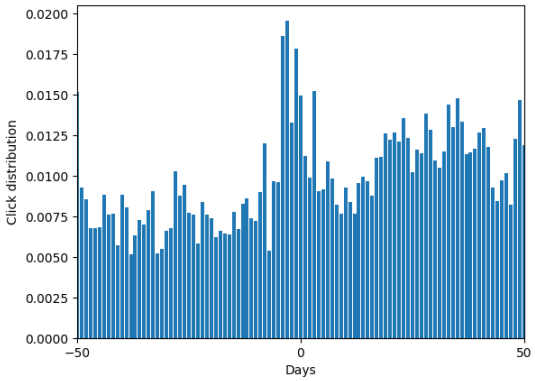
(a) Phishing - 48 Hours



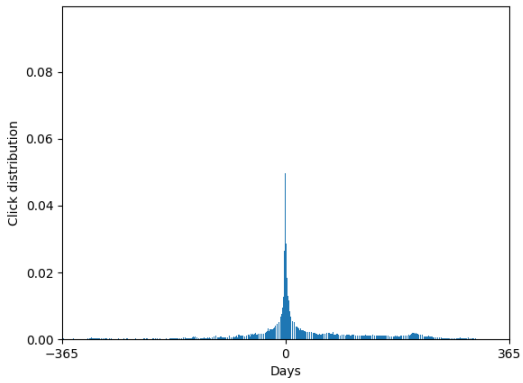
(b) Malware - 48 Hours



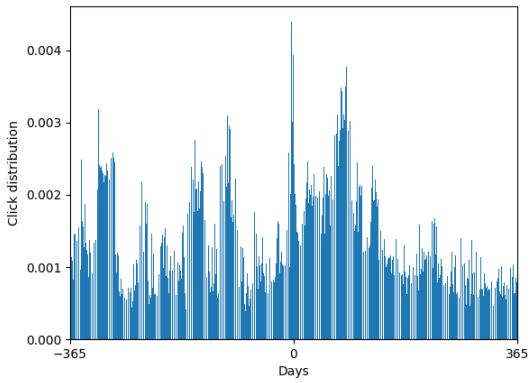
(c) Phishing - 50 Days



(d) Malware - 50 Days



(e) Phishing - 365 Days (1 Year)



(f) Malware - 365 Days (1 Year)

Figure 3.7: Click distribution of phishing and malware URLs encountered 48 hours, 50 days and 1 year before and after reported date.

figures, for phishing we see one clear peak, with a shorter tail before the peak and a longer tail after the peak. The shorter tail before the peak indicates that when phishing URL click activity starts, the clicks grow quickly. The longer tail after the reported date indicates that phishing attacks are slow to be completely removed, and we even see a slight bump in clicks at around the half year mark. This bump may indicate an attempt for phishing attacks to resurface. For malware we still see the highest peak near the reported date, but we also see several peaks before the reported date, and surprisingly the second highest peak is after the reported date, after which the clicks slowly start to reduce. The peaks before the reported date show that our sources for reporting malicious URLs are lagging and are not catching active malware URLs. The peak after the reported date indicates that malware attacks are successfully resurfacing and continuing to receive click activity, even after they have been reported. With the click activity remaining constant over a timespan of several years, this view also confirms our findings in Section 3.4.3, which found that malware attacks last longer than phishing attacks.

From previous research, Grier *et al.* [40] looked at the performance of several blacklists from when a malicious Twitter URL was posted. They found that there was a 20 day lag for malware URLs to be blacklisted and an 8 day lag for phishing URLs to be blacklisted. This shows that blacklists have a larger lag when detecting malware on Twitter. Overall our click distribution analysis also shows that based on the point at which click activity peaks, our reports have a larger lag when detecting malware URLs.

3.4.5 HTTP Referrer Clicks

In this section we look at the HTTP referrers referring the most number of clicks. We restrict our analysis to short URLs with clicks within at least 1 year of reported date. We find that phishing URLs were accessed from 5,480 referrer domains, while malware URLs were accessed from 1,468 referrer domains. This makes sense since we have more phishing URLs in our dataset, and since phishing URLs have more click activity, as discussed in Section 3.4.2.

As shown in Table 3.3, we find that short URLs are most commonly accessed "directly", that is, from sources including email clients, instant messages, and applications. This also matches other findings [6, 92]. We notice that although "direct" is the most common source for both attacks, it is much more common for phishing URLs. For phishing, the next most common referrer is Facebook. This includes referrers such as m.facebook.com, l.facebook.com, and facebook.com. Interestingly for phishing, Twitter is the source of only 0.25% of clicks *vs* 4.6% for malware. This indicates that Twitter's defense against malware URLs is not as strong as its defense against phishing URLs, as was already noted in [40].

Table 3.2: Top Referrer Categories With The Largest Number of Clicks.

	Phishing		Malware	
Rank	**Category**	**% Clicks**	**Category**	**% Clicks**
1	direct	61.93%	direct	33.07%
2	Social Media	12.14%	Blogs	29.85%
3	Social Networks	3.67%	Social Media	8.33%
4	Anonymisation	3.46%	Social Networks	8.32%
5	Webmail	2.31%	Search Engines	7.07%

Table 3.3: Top Referrers With The Largest Number of Clicks.

	Phishing		Malware	
Rank	**Referrer**	**% Clicks**	**Referrer**	**% Clicks**
1	direct	75.65%	direct	37.76%
2	facebook	10.94%	blogspot	33.30%
3	smikta	2.24%	google	7.43%
4	live	2.15%	twitter	4.60%
5	google	1.12%	facebook	4.08%

We also notice that a third of clicks for malware URLs are from Blogspot. For phishing we find smikta.net as a referrer with high click accesses; Smikta is an anonymisation service which advertises as a service allowing students to access any websites from school. We identified 11 other anonymisation services, which make up one of the top referrer categories for phishing, as shown in Table 3.2.

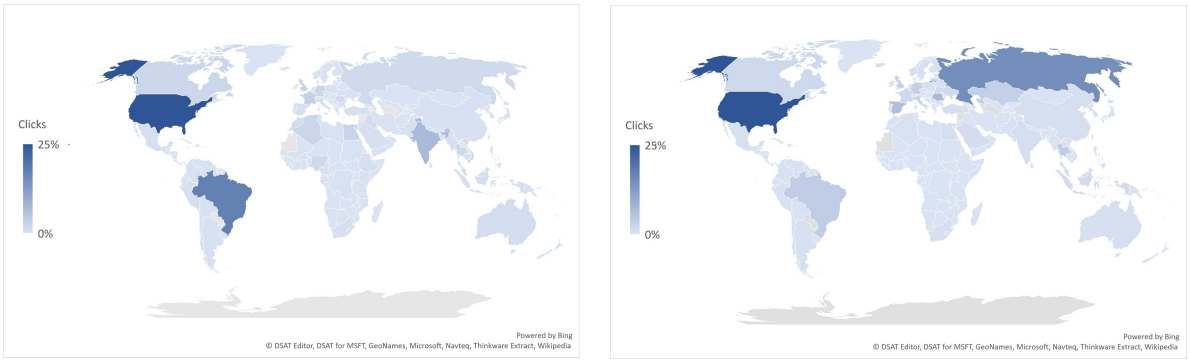
To get a bigger picture, as shown in Table 3.2, we grouped each referrer into categories using X-Force API endpoint "report". We were able to categorize 66% of phishing referrer domains and 72% of malware. For both phishing and malware URLs, we find that the majority of the clicks are from direct sources, social networks and social media, which together account for between 50 and 80% of clicks. This is probably because these sources allow short URLs to reach a large audience. Similar observations were made in 2013 by Wang *et al.* [92] who investigated Bitly spam short URLs from Twitter. Overall we find an increased use of social media to spread both phishing and malware.

Table 3.4: Top Countries With The Largest Number of Clicks.

	Phishing		Malware	
Rank	**Country**	**% Clicks**	**Country**	**% Clicks**
1	US (United States)	25.18%	US (United States)	24.57%
2	BR (Brazil)	17.40%	RU (Russia)	14.50%
3	IN (India)	7.13%	RO (Romania)	7.37%
4	FR (France)	3.25%	ES (Spain)	5.84%
5	GB (United Kingdom)	2.71%	TH (Thailand)	3.80%
6	EG (Egypt)	2.54%	BR (Brazil)	3.75%
7	DE (Germany)	2.27%	LT (Lithuania)	3.57%
8	DZ (Algeria)	2.25%	KZ (Kazakhstan)	3.31%
9	CA (Canada)	2.06%	DE (Germany)	3.30%
10	A1 (Anonymous Proxy)	1.72%	GB (United Kingdom)	3.22%

Table 3.5: Top Countries With The Largest Number of Clicks, Normalized by Population.

	Phishing		Malware	
Rank	**Country**	**% Clicks**	**Country**	**% Clicks**
1	CK (Cook Islands)	13.19%	SG (Singapore)	3.46%
2	TC (Turks & Caicos Isl.)	4.14%	JE (Jersey)	3.30%
3	BN (Brunei)	3.34%	GD (Grenada)	2.97%
4	SM (San Marino)	3.34%	EE (Estonia)	2.69%
5	BM (Bermuda)	2.75%	TC (Turks & Caicos Isl.)	2.64%
6	EE (Estonia)	2.65%	QA (Qatar)	2.37%
7	IL (Israel)	2.10%	IE (Ireland)	2.32%
8	SC (Seychelles)	2.00%	US (United States)	2.30%
9	QA (Qatar)	1.87%	NL (Netherlands)	2.21%
10	IS (Iceland)	1.73%	NZ (New Zealand)	2.15%



(a) Phishing (b) Malware

Figure 3.8: World maps showing countries referring click through for phishing and malware URLs by percentage of clicks. There are 236 countries referring click through for phishing URLs and 234 countries referring click through for malware URLs.

3.4.6 Country Referrer Clicks

In this section we look at the countries referring the most number of clicks. We restrict our analysis to short URLs with clicks within at least 1 year of reported date. We find that there are 236 countries referrering click through for phishing and 234 countries for malware. Unlike HTTP referrers, we find that phishing and malware have roughly the same number of countries referrering click through, perhaps because the number of possible countries is much less than the number of possible referrers. In total there are 254 possible country codes [54], which shows that nearly all countries are referrering click-through traffic for both phishing and malware URLs.

As shown in Table 3.4, we find that our malicious short URLs are most commonly accessed from US (United States), which matches several other findings [6], [52], [92]. This is probably due to a bias in our data: our data sources tend to be North-American centric, and Bitly is an organization from the United States which presumably has more users from a small pool of countries. We also find that for phishing, BR (Brazil) is the next most commonly accessed country, which may be because Brazil has anti-phishing organizations and are more involved in listing phishing attacks on public portals such as PhishTank. However, we find for phishing, China is not well represented in our database, even though the APWG shows China as one of the countries with the largest number of phishing attacks [4]. This may be because China does not interact as much with public portals and is more secluded when collaborating with other organizations in the United States.

Comparing phishing and malware URLs, we find that the top countries for both include

US (United States), BR (Brazil), GB (United Kingdom) and DE (Germany). We find the differences to be that phishing includes IN (India), FR (France), EG (Egypt), DZ (Algeria), CA (Canada) and A1 (Anonymous Proxy), while malware includes RU (Russia), RO (Romania), ES (Spain), TH (Thailand), LT (Lithuania) and KZ (Kazakhstan). Some of these similarities and differences can be seen more clearly in Figure 3.8 where we include all the country click percentages on a world map. Clicks from Brazil account for an especially high number of accesses for phishing. This follows the 1H 2017 APWG report [8], which shows Brazil as the second highest country reporting phishing incidents. Clicks from Russia account for an especially high number of accesses for malware. As the Q4 2016 APWG report shows [8], Russia is ranked #5 in the list of countries with the highest infection rate, right behind China, Turkey, Guatemala, and Ecuador. AWPG also report Scandinavian countries have the lowest infection rates. We find that for malware, Scandinavian countries account for 1.03% of clicks, whereas for phishing they only account for 0.43% of clicks.

In Table 3.5 we take into account the population of each country and normalize the click rate. The click rate for CK (Cook Islands) stands out for phishing cases, where as for malware there is no country that stands out in the top 10. Interestingly for malware, the US (United States) still appears in the top 10 despite its large population. The countries in common between phishing and malware taking into account population are EE (Estonia), TC (Turks and Caicos Islands), and QA (Qatar).

In 2013, Wang *et al.* [92] investigated Bitly spam short URLs from Twitter collected over a 4 month period, and found Thailand at the top of their list because Thailand was a referrer for one their short URLs which had exceptionally high number of clicks. We found a similar case in our dataset of a short URL with malicious domain "www.pizzahut1150.com", generating over 100,000 clicks from Thailand. This shows that malicious URLs may generate heavy traffic using URL shortening services and confirms our findings in Section 3.4.2 in which only a few URLs make up the majority of clicks.

Overall we find the United States as the top country referrer for both phishing and malware, where phishing clicks mainly come from USA and Brazil, while malware clicks mainly come from USA and Russia. We also find that Scandinavian countries have higher click percentage for malware than for phishing.

3.5 Conclusion

In this chapter we found several differences between phishing and malware attacks relative to click through, click timespan, report lag, and click sources. We found that phishing URLs receive more click-through traffic, where 60% of malware URLs receive fewer than

27 clicks while the same is true for only 20% of phishing URLs. We also found that malware URLs have longer timespan, where 50% of malware URLs are active for several years while less than 50% of phishing URLs are active past 3 months. Although phishing URLs have smaller timespan, those URLs that are active past 3 months have longer timespan than what has been reported in past research reporting that most phishing attack removal times were days and even hours. Our findings more closely follow what was found in [29], in which several phishing attacks remained active for up to 10 months.

Looking at the click distribution, for phishing we found the highest click activity to be 4 hours before the reported date, with clicks growing quickly before, and then slowly reducing afterwards. Clicks remained for up to half a year, with signs of the attacks attempting to resurface. Conversely, for malware we found the highest click activity to be 4 days before the reported date, with several other peaks before and after, indicating that our sources are not catching active malware URLs, and also indicating that malware URLs are successfully resurfacing even after being reported. Overall our click distribution analysis shows that, based on the point at which click activity peaks, our reports have a larger lag when detecting malware than phishing.

The results in our analysis of URL timespan and click distribution especially highlight that the efforts against malware attacks are not as strong when compared to phishing attacks. However, for phishing, the domain names might be more deceptive compared to Bitly short URLs. The use of short URLs for just a small window of time might be an attacker's strategy. Conversely, for malware, the timespan seems to be much longer than expected and it is surprising that the malicious domain is not taken down. It could be that Bitly counts the click even if the domain does not resolve anymore. We would be interested in exploring further to determine why malware URL efforts are not as strong.

During our analysis we also found evidence suggesting that Twitter spam click activity has remained consistent compared to research from 2010 [40]: We found that Twitter receives similar volume of click through traffic, where 5% of short URLs receive clicks from Twitter, as well as similar number of clicks per short URL, where 50% of short URLs receive fewer than 13 clicks. We also found that Twitter's defense against malware URLs is not as effective as its defense against phishing URLs since Twitter is the source of only 0.25% of clicks for phishing *vs* 4.6% for malware.

In regard to other HTTP referrers, we found that the majority of the clicks are from direct sources, social networks and social media, which all together account for 50% and 80% of clicks for phishing and malware URLs respectively. For country referrers, we found that United States was the top referrer for both phishing and malware URLs, both with similar click percentages. However, the rest of the top referrers were mostly different: Brazil

was notably the second top country for phishing, and Russia was the second top country for malware. We also found that Scandinavian countries have higher click percentage for malware than for phishing.

Lastly, we also observed the limitations of short URL click analytics, finding that flagged Bitly short URLs no longer record new clicks. This means that click analysis cannot be continued once a URL has been flagged. In analyzing country referrers we also acknowledge the limitation of Bitly short URLs since clicks from China are not prevalent in our dataset, in which case it would be interesting to pursue working on a larger and more complete dataset, such as shortening services from China.

Another limitation we were made aware of is that companies and products go on crawling phishing websites, sometimes through the Bitly link, as part of their anti-phishing strategy. This might affect the click analytics. This may explain the peak in click activity just before a phishing attack is reported. It may also explain the slight uptick in clicks several months later, since anti-phishing companies and products crawl previously known phishing attacks several months later to check whether the attack has resurfaced. It also makes clear that efforts in anti-phishing are stronger, since there are several companies and products that each try to solve the problem of phishing, particularly in the interest of protecting company brands. To take anti-phishing click activity into consideration, one could use user identifiers that match specific click activity to a user account or IP address for example. Then one could remove the clicks corresponding to identifiers that are common among a large majority of phishing attacks, since these identifiers are most likely anti-phishing companies and products. However another limitation of short URL analytics is that such identifier information is not made publicly available.

Chapter 4

A Comparison of Methods to Detect Phishing Attack Variations

4.1 Introduction

In this chapter we compare methods used to measure the similarity between phishing attacks as a way to detect replicas of known phishing attacks. Specifically we compare our novel tree method [55] against the vector method [29]. We first introduce our novel method to measure similarity between phishing attacks. Next we discuss our experimental setup, in particular how we handle the computational cost of the tree edit distance. We discuss our clustering results and comparisons of both the strengths and weaknesses of the methods, followed by a conclusion at the end of the chapter.

4.2 Proposed novel tree-based approach

The problem that we wish to tackle here is, given a new DOM, find the closest DOM among a database of DOMs of known phishing attacks.

DOM files can be represented as parse trees. Several publications present frameworks to use edit distance as a way to compare trees [80]. We will evaluate the feasibility of applying the edit distance on trees as a way to compare DOMs and find phishing attack instances that are similar to one another, and then compare this evaluation against [29]. Comparison includes the flexibility, accuracy, space, and speed requirements of these two methods. We will analyze our results to see whether our tree method approach exposes different ways in which variations of phishing pages are created. We will also analyze

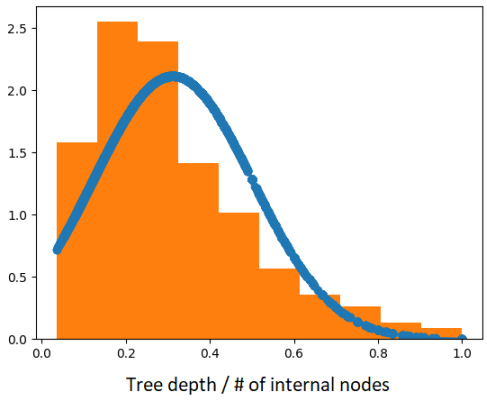


Figure 4.1: Distribution of DOM tree shapes in our database.

whether there are similar attacks that the vector method fails to catch since this method does not capture the structure of the DOM.

4.2.1 Tree Edit Distance

In this section we introduce and explain the concepts of our proposed tree method, which uses the Tree Edit Distance (TED) to measure the structural similarity between two DOMs.

The *Tree Edit Distance* (TED) is one of the most popular metrics for measuring the structural similarity between two DOMs. It represents the minimum number of operations (adding, removing and replacing) to convert one tree into the other.

The Tree Edit Distance (TED) is defined as the minimum-cost sequence of node edit operations[1] that transform one tree into another. This is a recursive problem with several sub-problems: a naive algorithm to calculate TED results in exponential runtime. Fortunately, several algorithms using dynamic programming achieve TED calculations in polynomial runtime.

The complexity of the best TED algorithm to date, *AP-TED* [76], is still $O(n^3)$, where n is the number of nodes in the DOM.

According to a recent result [21] it is unlikely that a truly subcubic TED solution exists. Although TED is cubic in the worst case, for many practical instances the runtime is much faster. It has been shown that some TED algorithms perform better for certain tree shapes [80]. For example, the algorithm by Chen [25] is efficient for thin and deep

[1]Deleting a node and connecting its children to its parent, inserting a node between an existing node and the children of this node, and renaming the label of a node.

trees, whereas the algorithm by Zhang and Shasha [99] is efficient for some interesting tree shapes such as balanced trees. The performance of the algorithm AP-TED by Pawlik and Augsten [76] however does not depend on the tree shape.

To explore what kind of TED algorithm to use, we tried to determine if there were any distinct tree shapes in our database. Figure 4.1 gives an idea of the tree shapes in our database, using a measure of tree depth divided by the number of internal nodes in the tree. It shows roughly that the distribution of tree shapes in our database is varied: most of the trees are not long and thin, since the number of nodes on the longest path is much less than the total number of internal nodes in the tree. As a result we chose the new, "state of the art" AP-TED [76] algorithm to calculate TED. The AP-TED algorithm has a time complexity of $O(n^3)$ and memory complexity $O(m * n)$, where n and m are the sizes of the two trees being compared. Here, tree size is defined as the number of nodes in the tree. To reduce the complexity of computing TED, some approaches based on fuzzy hash [65] or information retrieval [36, 39] have been proposed. These methods are however limited and cannot be used to find out the specific differences between the trees.

4.2.2 Tree Method

Since we are comparing our method with the vector method proposed by Cui *et al.* [29], we will extract the tag features from the DOM. For each DOM, we compute the corresponding tree by parsing the DOM in bracket notation. We consider tags within the entire DOM.

Consider Figure 2.1, where two simple DOMs are provided. The corresponding tag trees in bracket notation for pages p_1 and p_2 are as follows, respectively:

```
{body{div{div{p}{p}{form{input}{button}
  {input}}}{div}}{div{h1}{h1}{h1}{video}}}

{body{div{div{h1}{h1}{h1}}{div}}{div{div}
  {div}{form}{h1}}}.
```

To compare the distance between tag trees, we use the *Proportional Tree Edit Distance* ($PTED$), which divides the differences by the sum of differences and matches. Given two pages p_1 and p_2, the number of differences $diff(p_1, p_2)$ is based on the tree edit distance, which is the minimum cost of mapping the source tree into the destination tree. In our implementation, the default costs we use is 1 for deletion, insertion and renaming. The number of matches $matches(p_1, p_2)$ is defined as the number of nodes of the trees that are mapped from one to the other without edits.

Let t_1 and t_2 be two non-null tag trees. Formally, the *Proportional Tree Edit Distance* $PTED(t_1, t_2)$ between t_1 and t_2 is defined as follows:

$$PTED(t_1, t_2) = \frac{diff(t_1, t_2)}{diff(t_1, t_2) + matches(t_1, t_2)} \tag{4.1}$$

As an example, consider the two pages from Figure 2.1: $diff(t_1, t_2) = 10$, and $matches(t_1, t_2) = 6$, so $PTED(t_1, t_2) = 10/(10 + 6) = 0.625$. This value indicates a large difference between the two pages.

4.2.3 Clustering Algorithm

The clustering algorithm we use is similar to the one described in [29]. We used a minimum spanning tree approach, which sorts the distances between entities from smallest to largest. Entities are then grouped together in this order, where smallest distance indicates that they are the most similar to each other and so on.

This method is based on a threshold that yields clusters that are both compact and far away from each other. This is defined as the *quality of clustering* QC, using the average similarity distance of entities inside a cluster. We average this value, and divide it by the smallest similarity distance between two entities in any two clusters.

4.3 Experiments

4.3.1 Phishing Database

We compiled our phishing database by collecting URLs of phishing attack instances from the community-driven portal PhishTank and the enterprise security analysis platform IBM X-Force. A total of 64,790 "verified" phishing sites were collected by fetching the daily archive from PhishTank between January 1st, 2016 to December 31st, 2017 and from IBM X-Force between June 12th, 2017 to December 31st, 2017. For each phishing site, we fetched the DOM of the final URL using a crawler.

4.3.2 Optimal Threshold

In order to calculate the optimal threshold for our clustering algorithm, we need to perform distance calculations for each comparison. Our database of 64,790 phishing attack instances yield 11,092 different trees and 10,762 different vectors, as shown in Table 4.1. This means

that we will need to perform about 62 million pairwise comparisons for the tree method and about 58 million pairwise comparisons for the vector method. This poses a problem when calculating TED for the tree method, since these calculations are more computing intensive. To solve this problem, we decided to run TED pairwise comparisons on a subset of our database, to obtain the optimal threshold of the subset, and to then use this threshold on our whole database.

4.3.3 Running distance calculations on a subset

In order to run our experiments, we first generated the tag tree and tag vector representations of the phishing sites in our database. As several DOMs in the database yield the same tree or the same vector, from our database of 64,790 phishing sites, we only obtained 11,092 unique trees, and 10,762 unique vectors, as shown in Table 4.1. It is interesting to note that the number of vectors generated is less than the number of trees generated. The reason for this is that two instances with the same tag tree have the same tag vector, but the converse is not necessarily true: the vector method is only performing a simple count of the tags, while the tree method is capturing more details such as the structure of the DOMs. This reveals that the vector method has already found more replicas than the tree method, since the same number of phishing sites are represented using fewer tag vectors than tag trees.

The tree size is defined as the number of nodes in the tree, where the nodes have tag labels, and the TED runtime increases as the cube of the tree size. The tree size varies in our database, and the largest tree size is approximately 500,000. As we need to perform approximately 62 million pairwise calculations, as shown in Table 4.1, the runtime would be very long.

Fortunately, tree sizes as large as 500,000 are an anomaly in our database, since 99.77% of entries in our database have tree sizes less than 5,000, as shown in Table 4.1. Furthermore, 83.61% of the trees in our database have sizes less than 500. Figure 4.2 shows the distribution of trees less than 500, indicating that the majority of trees in our database are smaller in size. As a result, we only ran our calculations on smaller sized trees in a preliminary experiment. In this case it is reasonable to focus on groups of trees with similar tree size, since large differences in tree size result in large distances due to insertions and deletions. By focusing on smaller sized trees, the TED calculations run much faster. In addition, by focusing on a subset of smaller tree sizes, this also results in fewer pairwise calculations of TED.

In our experiment we therefore focused on trees with sizes between 100 and 150. This

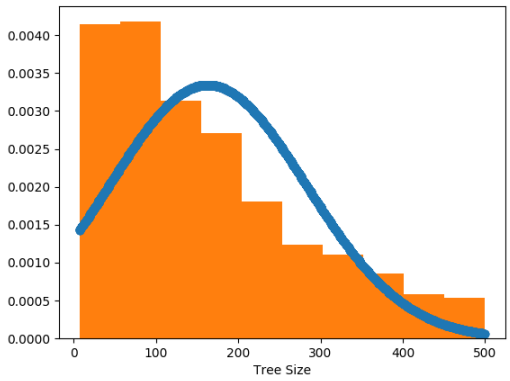


Figure 4.2: Distribution of trees in our database with tree size less than 500.

resulted in a reduced dataset of 6,271 phishing attacks, which only generated 758 unique trees, and 737 unique vectors as shown in Table 4.1. We did not elect to run calculations on trees with size 0 to 50, or 50 to 100 for example, because the phishing attacks with tree sizes this small are less interesting, and are usually just simple pages showing text without input fields or forms. For this reason we chose trees with sizes between 100 and 150 to focus on more complex phishing attacks, while still keeping the runtime manageable. The results of our distance calculations on trees with sizes between 100 and 150 are shown in the bottom portion of Table 4.1. Running all the calculations still took about 12 hours on a personal laptop using the tree method, but only took about 30 seconds using the vector method.

4.3.4 Optimal Threshold Results

To choose the value of the clustering threshold for the tree method and the vector method, we have computed the threshold that will provide the minimal (*i.e.* the best) quality of clustering (QC) as defined in Section 4.2.3. To get the optimal thresholds, we manually checked thresholds with step 0.01 from 0.1 to 0.4, and obtained the clustering results. We then calculated QC based on the clustering results, choosing the lowest QC value as the optimal threshold. We then calculated QC based on the clustering results, choosing the lowest QC value as the optimal threshold. Results are shown in Figure 4.3. For the tag trees, the optimal threshold was calculated to be 0.32, while for tag vectors, the optimal threshold was 0.35.

Table 4.1: Database (top), Tree size (middle) and Dataset distance calculation (bottom)

Phishing sites database	**Size**
# of phishing attack instances	64,790
# of trees	11,092
# of vectors	10,762

Tree size	**# of dataset**	**% of dataset**	**# of pair calculations**
All trees	11,092	100.00%	61,510,686
less than 5,000	11,066	99.77%	61,222,645
less than 500	9,274	83.61%	42,998,901
less than 50	1,640	14.78%	1,343,980

Dataset of DOMs where 100 <= tree size <= 150	**Tag Tree**	**Tag Vector**
# dataset	758	737
# pairwise calculations	286,930	271,216
# average runtime (seconds)	0.1477	0.00006130
# runtime (hours)	11.77	0.004618

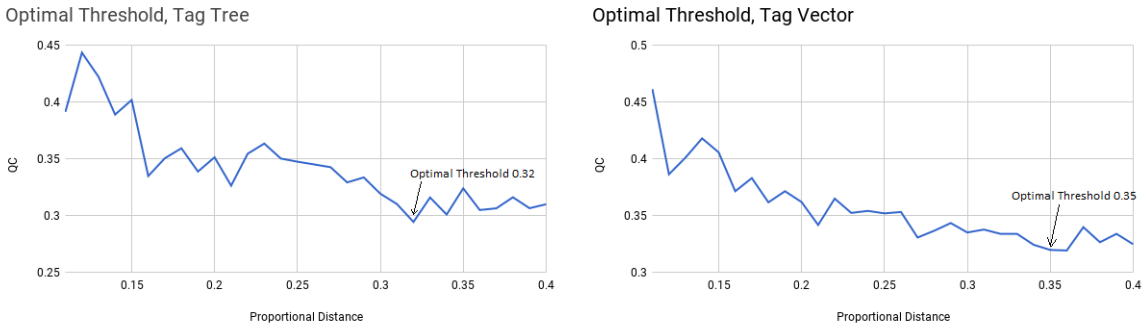


Figure 4.3: Optimal threshold computation for Tag Tree and Tag Vector.

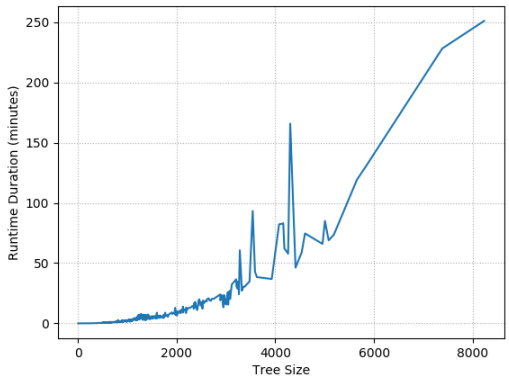


Figure 4.4: The average runtime to calculate TED against tree size, where tree size is defined as the number of nodes in the tree.

4.3.5 Running distance calculations on our database

In this section, we use the optimal threshold calculated above in Section 4.3.4, to reduce the complexity of computing TED on our entire database of 64,790 phishing attack instances, represented as 11,092 tag trees. We obtained this optimal threshold using a subset of our database, and found the optimal threshold to be 0.32 and 0.35 for the tree method and vector method respectively.

Using the threshold, we can perform quick calculations to decide whether or not to continue calculating TED based on the two discriminant quantities:

$$PSDiff(t_1, t_2) = \frac{abs(t_1, t_2)}{max(t_1, t_2)} \tag{4.2}$$

$$PTDiff(t_1, t_2) = \frac{max(t_1, t_2) - sim(t_1, t_2)}{max(t_1, t_2)} \tag{4.3}$$

In equations 4.2 and 4.3 above, $abs(t_1, t_2)$ is the absolute value of the difference of the sizes of the two trees, $max(t_1, t_2)$ is the size of the larger tree, and $sim(t_1, t_2)$ is the number of nodes that are similar in both trees.

It is straightforward to note that both $PSDiff$ and $PTDiff$ cannot be smaller than $PTED$ if all costs have the same value. In addition, both are straightforward to calculate (milliseconds on a laptop even for large tree sizes) while that of AP-TED can be computationally very intensive.

If both discriminant quantities $PSDiff$ and $PTDiff$ are below the threshold, then we continue with calculation of $PTED$ between the tree pairs. As will be shown below,

the above simple remark actually results in significant compute-time savings.

As an example, consider the two pages from Figure 2.1: $abs(t_1, t_2) = |15 - 12| = 3$, $max(t_1, t_2) = 15$, and $sim(t_1, t_2) = 9$, therefore $PSDiff(t_1, t_2) = 3/15 = 0.2$ and $PTDiff(t_1, t_2) = (15 - 9)/15 = 0.4$. Although $PSDiff$ is less than our threshold 0.32, $PTDiff$ is larger than our threshold, therefore the $PTED$ of these two trees will not have to be computed: the resulting $PTED$ would be larger than even $PTDiff$, and therefore larger than the threshold. Indeed Section 4.2.2 shows that $PTED$ does not consider these two trees as similar.

Using these two discriminant quantities, from our 11,092 trees, we found that this reduced the number of TED calculations on all pairwise comparisons from 61,510,686 calculations to 367,233 calculations, which means that we avoided in this way 99.4% of the calculations. After sorting our trees from smallest to largest, and starting with the smallest tree size, using 38 threads on a power machine with 64 cores (2.4GHZ per core), it took 113 hours to perform 367,220 calculations, in which only the last 13 calculations for our largest trees timed out. Therefore we find that computing TED calculations on 99.996% of the pairwise comparisons on our large database is feasible using our threshold method.

Figure 4.4 shows the average runtime taken by our machine to calculate TED against tree size. We find that trees of size 500 took more than 1 second each, and trees of size 5,000 took more than 1 hour each. The longest runtime was over 4 hours for trees of about 9,000 in size. Of the trees being compared in our 367,233 comparisons, we only have 6 trees larger than 9,000, the largest being about 50,000 in size. Timeout for our calculations was set to 8 hours. Only 13 comparison calculations of our largest trees took longer than 8 hours and were timed out.

In the next section, we use the optimal thresholds 0.32 and 0.35 for the tree method and vector method respectively, to cluster our tag trees and tag vectors. We do not recalculate the optimal threshold here because it is not comparable with the TED distribution of our sampling case of 367,233 comparisons. This is because we ignored many of the larger TED calculations using our discriminant quantities. Calculating a new threshold would necessarily yield a smaller threshold.

4.3.6 Clustering Results

Our main results are shown in Table 4.2: we find that our database of tag trees and tag vectors yields 5,243 clusters and 6,218 clusters (*i.e.* phishing classes) respectively. This

shows that the tree method was able to group more phishing attack instances into fewer phishing classes by finding more similarities between the phishing attack instances.

Clusters were categorized into three groups: "flagged clusters with more than one entity", "flagged clusters with one entity", and "singleton clusters". If we consider the tree method, the category of "flagged clusters with more than one entity" defines a cluster with more than one tag tree, where each tag tree represents one or more phishing instances. The category of "flagged clusters with one entity" defines a cluster with one tag tree, where the tag tree represents more than one phishing instance. Finally the category "singleton cluster" defines a "cluster" with a single phishing instance of a tag tree. Therefore we define flagged clusters as clusters with more than one phishing attack instance, and singleton clusters as clusters with only one phishing attack instance. It is interesting to note here that the tree method has more "flagged clusters with more than one entity" than the vector method. This means that the tree method flags more repeated attacks by finding similarities between different tag trees. Conversely, the vector method has more "flagged clusters with one entity" than the tree method. This means that the vector method flags more repeated attacks by representing several different phishing instances as one vector.

The most important result is that using our tree method, 61,955 of the 64,790 phishing attack instances end up in a cluster with more than one phishing attack instance (so-called *flagged cluster*). This means that the tree method finds that 95.62% of our database is made of repeated attacks. Similarly, using the vector method, 61,316 of the 64,790 phishing attack instances end up in a flagged cluster, which means that this method finds that 94.64% of our database is made of repeated attacks. Notice that this is larger than in [29] who reported 91.53% of their database is made of repeated attacks, using a database of 19,066 phishing sites. This is expected because chances of finding a replica in a database obviously increase with its size.

For phishing instances flagged by both methods, 61,237 of the same phishing attack instances end up in a flagged cluster, which means that both methods agree that 94.52% of our database is made of repeated attacks. If we took the union of phishing attack instances flagged by either methods and combined them, we find that 62,034 phishing attacks are flagged, which means that both methods combined find that 95.75% of our database is made of repeated attacks. This leaves about 5% of phishing instances not flagged as replicas by either method. Although these instances may represent novel phishing attacks, some may still be replicas that are not flagged by our method.

At the bottom of Table 4.2, we break down the number of phishing attack instances flagged by both methods, by one and not the other method, and by neither of the two methods. In the next section, we will take a closer look at the differences between phishing

Table 4.2: Database details (top) and clustering results (bottom)

Phishing sites database	**Size**
# of phishing attack instances	64,790
# of trees	11,092
# of vectors	10,762

Cluster Results	**Tree Method**	**Vector Method**
# flagged clusters with > 1 entity	1190	1162
# flagged clusters with 1 entity	1218	1582
# singleton clusters	2835	3474
# total clusters	5243	6218

Phishing Instances	**Tree Method**		**Vector Method**	
	#	%	#	%
# phishing in flagged clusters	61,955	95.62%	61,316	94.64%
# phishing in singleton clusters	2,835	4.38%	3,474	5.36%

Flagged Phishing Instances	**Size**
# flagged by both methods	61,237
# flagged by tree method and not vector method	718
# flagged by vector method and not tree method	79
# flagged by neither methods	2,756

attack instances flagged by one method but not the other. Further analysis of the phishing attack instances flagged by both methods, and of those flagged by neither method would have been interesting as well. Unfortunately, the volume of phishing attack instances in these two categories was too large for manual checking within the scope of the present study. However further analysis is planned to examine in more detail the reason why the tree method extracted more clusters than the vector method (112 *vs.* 105) and whether there are any false negatives in the phishing sites flagged by neither method.

4.3.7 Similar Cases

As shown in Table 4.2, there are 718 phishing attack instances flagged by the tree method and not by the vector method, and 79 phishing attack instances flagged by the vector method and not by the tree method.

In this section we manually check a few of these cases and discuss why one method considers these cases to be similar while the other method does not. We focus on phishing attack instances that look visually similar.

Table 4.3: Phishing Variants found by Tree Method and not by Vector Method, Sorted by Difference in Distances.

Case	Phishing Instance, MD5 1	Phishing Instance, MD5 2	Tree Distance	Vector Distance	Difference
1	33e250cc00b376d5e89fff0a8bf8ab02	f94c3d7b14c7f06f28730b534b55fb8f	0.232	0.714	0.482
2	201374f7ed38425702dd76c0facc861f	8784ed420bf726eafb2624fc763c811f	0.046	0.500	0.454
3	272926527360613fd5b0bf085455af5d	1f8960f1e4e6d11e343b7ecc7ace447e	0.285	0.654	0.369
4	7b54ed0ffd2f1c342135a79191c9ce0a	6c3faa4c70c5f1cf776373d9f5ee2eff	0.071	0.407	0.336
5	abd6c895866b5cc370993554abb09308	81a319d9e412c458d0a19a1999b28edd	0.263	0.500	0.237
6	ea8458bda6c9301bb86257400aa63a6c	593b22127fbf91b6b3e3e30041c50ec0	0.272	0.476	0.204
7	f0a9981eefb420550d90e90e0123832e	a617945a1164e23ca5220c872956008a	0.165	0.333	0.168
8	796b477846d6af90cc03caf831b3f2d1	fee3166de1d1680ed8cf0b925e0af87b	0.190	0.333	0.143

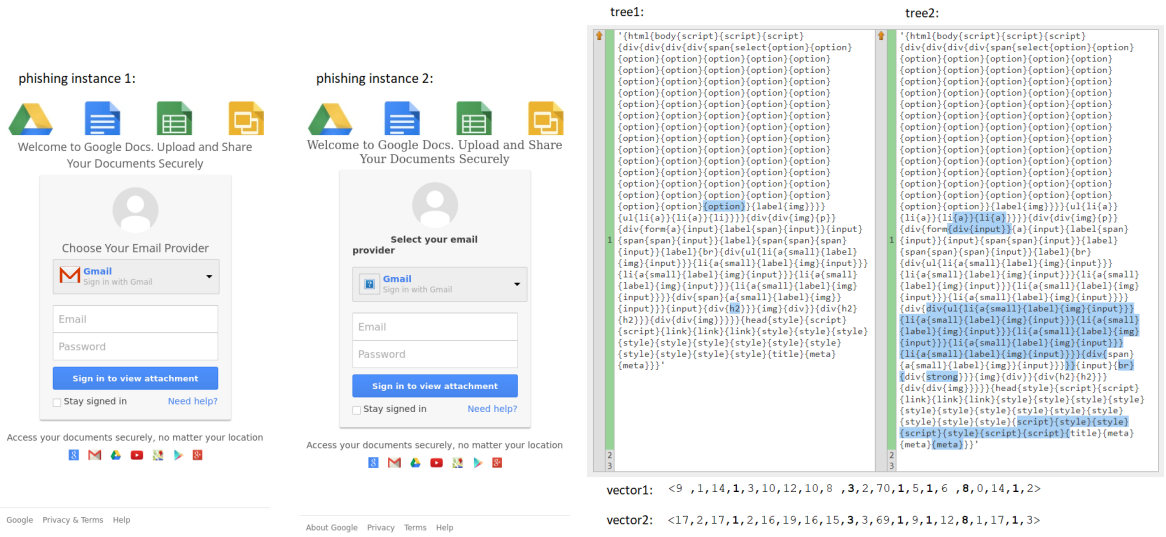


Figure 4.5: The following example is taken from Case 1 in Table 4.3. This is an example in our database of a phishing variant considered similar using the tree method but not using the vector method. A screen shot is provided for both phishing instances, as well as the tag trees and tag vectors. The differences between the trees have been highlighted, and the tags that have a count of zero for both vectors have been removed, so as to more easily spot the differences. Between the tag trees, there are 55 edits and 182 matches, so the $PTED(p1, p2) = 55/(55 + 182) = 0.232$, indicating that these cases are similar. Between the tag vectors, 21 of the 107 possible tags of the corpus are used by at least one of the two vectors, and 5 tags appear the same number of times in both vectors, so $PD(p_1, p_2) = (21 - 5)/21 = 0.714$, indicating that these cases are not similar.

Phishing flagged by the tree method but not by the vector method

Table 4.3 lists eight sample cases of phishing variations that do look visually similar and are considered similar by the tree method but not by the vector method. The cases are sorted by the difference in distance between the tree method and vector method. Figure 4.5

shows the screen shots, tag trees and tag vectors of Case 1 from Table 4.3, with largest difference in distance between the tree method and the vector method. This is an example of two phishing instances targeting Google credentials.

Even though very few changes were made to the DOMs, 16 out of 21 tag counts ended up being different. These changes have a significant impact on the vector distance calculation. On the other hand, these tag differences do not affect the structure of the DOM as much, so the tree distance remains small and the similarity is caught by the tree method.

Phishing flagged by the vector method but not by the tree method

While looking at phishing instances considered to be similar by the vector method and not by the tree method, we found several cases where many of the edits and differences made to the DOM where done using tags that are not included in the tag corpus of HTML elements provided by the World Wide Web Consortium [94] used by the vector method but not by the tree method. This is because the tag corpus is based on the list of HTML elements provided by the World Wide Web Consortium [94]. Tags such as <wml>, <template>, <do>, and <prev> are not included in the list. Therefore these differences in tags do not affect the similarity distance for the vector method. However our tree method considers any tag included in the DOM, so these differences in tags still affect the similarity distance for the tree method.

In addition we found cases where the majority of the changes to the DOM were additions or deletions of just one type of tag between the two phishing instances. This affected more the similarity distance of the tree method than the vector method.

4.4 Analysis and Discussion

4.4.1 Types of phishing attack variations

The results from the experiment in Section 4.3.6 reveal that different similarity methods work better to detect certain types of phishing attack variations. Even more, they reveal that phishing attacks are being modified in different ways. This type of insight informs us more about the ways attackers create variations.

For example, Figure 4.5 shows that similarity calculations for the vector method can be greatly affected if attackers introduce new tags *vs* introducing more of the same tags from the attack instance it is based upon. Figure 4.5 also shows that attackers can make

quite a few purposeful changes to the DOM, resulting in two variations that still appear very similar to the phishing victim.

As shown in Table 4.2, there are 718 phishing attack instances flagged by the tree method and not by the vector method, and 79 phishing attack instances flagged by the vector method and not by the tree method.

Phishing flagged by the tree method but not by the vector method

Table 4.3 lists eight sample cases of phishing variations that do look visually similar and are considered similar by the tree method but not by the vector method. The cases are sorted by the difference in distance between the tree method and vector method. Figure 4.5 shows the screen shots, tag trees and tag vectors of Case 1 from Table 4.3, for which the difference in distance between the tree method and the vector method is largest. This is an example of two phishing instances targeting Google credentials. Even though very few changes were made to the DOMs, 16 out of 21 tag counts end up being different. These changes have a significant impact on the vector distance calculation. On the other hand, these tag differences do not affect the structure of the DOM as much, so the tree distance remains small and the similarity is caught by the tree method.

Phishing flagged by the vector method but not by the tree method

We examined phishing instances considered to be similar by the vector method and not by the tree method. We found several cases where many of the edits and differences made to the DOM were done using tags that are not included in the tag corpus of HTML elements used by the vector method. In contrast, the tree method uses the full corpus provided by the World Wide Web Consortium [94]. We also found cases where the majority of the changes to the DOM were additions or deletions of just one type of tag between the two phishing instances. This affected more the similarity distance of the tree method than the vector method.

4.4.2 Comparison of the methods

Speed and memory

For the tree method, the time complexity of the AP-TED algorithm is $O(n^3)$ and the memory complexity is $O(m*n)$, which means that for some tree sizes, TED calculations are not feasible unless very powerful computers are used. In contrast, for the vector method

the time complexity is $O(n)$ and the memory complexity is constant since a predefined corpus is used.

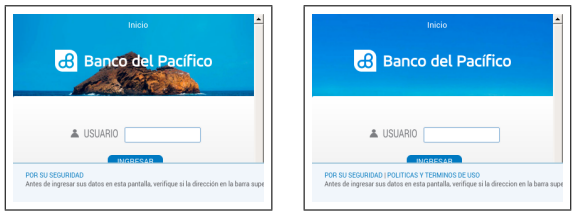
In terms of fast computations and small memory space, the vector method is clearly superior. On an average personal laptop, calculation of all pairwise comparisons with our current database using the vector method would take a few minutes. In contrast the tree method would take several years.

Flexibility and accuracy

The features extracted using the tree method provide quite a bit of flexibility. In our experiment we only extracted the tags from the DOMs, but it would be interesting to experiment by extracting the tree structure, or by extracting the attributes. The vector method is less flexible in this way, since the tag corpus is predefined, and does not include all possible tags. Another flexible feature that the tree method has is that the edit costs can be customizable. It may be interesting to experiment with different costs, where for example insertions and deletions would cost less whereas renaming from one tag to another would cost more. A downside to the flexibility of the tree method is that the DOM structure must be valid in order to parse the tree in bracket notation. On the other hand, for the vector method, the DOM structure does not need to be valid since we are simply counting the tags.

Comparing the accuracy between both methods, we do not know precisely how many of the phishing sites do have replicas in our database. We therefore cannot report the number of false negatives in our experiment, but this number is probably much lower than 4.38% and 5.36% (see Table 4.2) for tree method and vector method respectively. In fact if we assume all 718 phishing attack instances detected as replicas by the tree method and not the vector method are indeed replicas, then the false negatives for vector method is at least 718 of 64,790 phishing instances, or 1.11%. Conversely for our tree method the false negative would be at least 79 of 64,790 phishing instances, or 0.12%. This shows that the vector method has 9.25 times more false negatives than the tree method.

To evaluate the false positive rate, we looked at a random sampling of 6,239 legitimate sites taken from Alexa top 500 most popular sites by countries [1], as well as Alexa top 100,000 to 460,697 websites. Only 32 of these sites end up in one of the 5,243 phishing clusters for the tree method, a false positive rate of 0.52%. Conversely, 18 of these sites end up in one of the 6,218 phishing clusters for the vector method, a false positive rate of 0.29% (which is slightly higher than the rate of 0.08% found on [29], on a smaller set of phishing attacks). In [29], they found that their vector method has 0.08% false positives



(a) Legitimate site (b) Phishing site

Figure 4.6: Example of a false positive for tree method.

using 24,800 legitimate sites. We performed a similar experiment using 10,000 legitimate sites, collected from a series of Alexa top 500 most popular site by countries [1], as well as Alexa top 100,000 to 460,697 websites. Using the tree method to compare these legitimate sites to our phishing sites, after 10 days we were able to compare them to 6,239 legitimate sites. Only 32 of these sites end up in one of the 5,243 phishing clusters for the tree method, a false positive rate of 0.52%. Conversely, 18 of these sites end up in one of the 6,218 phishing clusters for the vector method, a false positive rate of 0.29%. We find the false positive rate for the vector method is higher than in [29], which makes sense since our database has a larger number of phishing attacks to which the legitimate sites are compared. We also find that the tree method has about 1.8 times more false positives than the vector method.

For all the cases in which the vector method considers a legitimate site similar to a phishing site, the tree method also considers them to be similar. This leaves 14 cases where the tree method considers a legitimate site similar to a phishing site, while the vector method does not consider them as similar. Looking closer at these 14 cases, 10 of the cases are considered similar due to the DOMs being small in size, where each case has a DOM with less than 30 tags. This shows that the tree method is more sensitive to small DOMs. Since the DOMs are smaller, this makes it more likely that the DOM is similar to other small DOMs. We also find that the reason these DOMs are small is because the sites are either error messages, or parked domains. Next, two of the cases are considered similar because the DOM is mostly made of repeated tags, such as the “options” tag and “image” tag. These cases show that if the majority of the DOM is made up of a single repeated tag, then this throws off the tree method more than the vector method. Finally, two of the cases are considered similar because the legitimate sites are actually a copy of the phishing site with some modifications (See for example Figure 4.6 (a) and (b)). In practice, we could reduce most of these false positive rates even further by simply ignoring small DOMs, for example ignoring DOMs with less than 30 tags, when building our database of phishing attacks.

In terms of the flexible features extracted from the DOM and customizable edit costs, as well as the likely improved accuracy in false negatives, we find that the tree method is somewhat better than the vector method. Although we found that the false positive rate is higher for the tree method, we determined that most of these cases can be reduced by ignoring very small DOMs. In addition, the tree method was able to find phishing sites making copies of legitimate sites, which the vector method did not find. Overall we find that the tree method better discriminates the similarity with known attacks since the tag tree is a more precise representation of the attack than the tag vector

4.5 Conclusion

In this chapter we have presented a new method to detect replicas of known phishing attacks. This new method applies the edit distance on trees as a way to compare DOMs and find phishing attack instances that are similar to one another. We also compared our tree method with an existing vector method, using a database of 64,790 phishing attack instances.

Overall we found that 95.62% of our database is made of repeated attacks using the tree method *vs* 94.64% using the vector method. We also found that our databases of tag trees and tag vectors yield 5,243 clusters and 6,218 clusters (*i.e.* phishing classes) respectively. This shows that the tree method was able to group more phishing attack instances into fewer phishing classes by finding more similarities between the phishing attack instances. This also shows that our tree method better discriminates the similarity with known attacks since the tag tree is a more precise representation of the attack. However, this also indicates that although the vector method does not capture the structure of the DOM, it still catches almost as many similarities.

We found that both methods detected the same 61,237 phishing attacks as replicas. This means that both methods agree that 94.52% of our database is made of repeated attacks. There were also some phishing attacks which were flagged as a replica by one method and not the other. We determined the main reason for these differences in flagged replicas to be that the vector method is more robust when the differences made between phishing instances are to the same tags. However if there are tags used in one phishing instance that are not used in the other, it was found that the tree method is more accurate at detecting similar cases.

In our comparison of these two methods we found that unless very powerful computers are available, the tree method is not practical to analyze an entire database with large trees. Taking this into consideration, we introduced a method to reduce the complexity by

99.4% when calculating pairwise edit distance on trees. We found that the advantage of using the tree method is its flexibility in defining what similar means, such as extracting different features of the phishing site, and adjusting the cost of some edits. We found that the disadvantage to the tag vector method is that the corpus of tags is predefined and set, and does not include necessarily all tags that are used in a given phishing page. We find that indeed a tree-based approach is able to catch more similarities between attacks, and can identify some of the small changes that are defeating the vector method. The tree-based method is fast enough to be usable, but remains much slower than the vector method even when the most efficient Tree Edit Distance algorithms are used.

This work shows that no single distance method will detect all types of phishing attack variations. Continued defense against phishing attacks will require a variety of different comparison methods in order to detect novel variations introduced by creative attackers.

We are starting to see attackers making variations between phishing instances by using dummy tags added to the DOM. This does not change the appearance of the phishing site, but it currently throws off the distance methods which conclude that the two instances are not similar. To improve the robustness of our tree method, we could consider using subtree pruning or local alignment. We could also consider integrating an image comparison algorithm. The goal of this idea would be that the dummy tags would not affect the screen appearance of the variations for phishing victims. Combinations of these three methods would be much stronger than using just one of the methods.

Chapter 5

A Multi-dataset Domain Classification of Phishing Attacks

5.1 Introduction

In this chapter we present a domain classifier which determines whether a phishing website is hosted on a legitimate but compromised server, in which case the owner is also a victim, or a maliciously registered domain. For simplicity we refer to compromised servers as compromised domains. Determining whether a phishing attack is hosted on either domain offers insight into how attackers commit their crimes and suggests different remediation options.

The remainder of this chapter is organized as follows. Section 5.3 describes our feature set and machine-learning algorithms. The experimental setup is described in Section 5.4, and the full experimental results are reported in Section 5.5. In Section 5.6, we present our analysis of classifying APWG's unlabeled phish feed. We conclude in Section 5.8.

5.2 System Architecture

5.2.1 Supervised, Semi-supervised, and Unsupervised Learning

Machine learning makes use of algorithms and statistical models to effectively perform a specific task without using explicit instructions, relying on patterns and inference through data analysis. The most common machine learning tasks are predictive, predicting a target variable from features using supervised, semi-supervised, or unsupervised learning. In

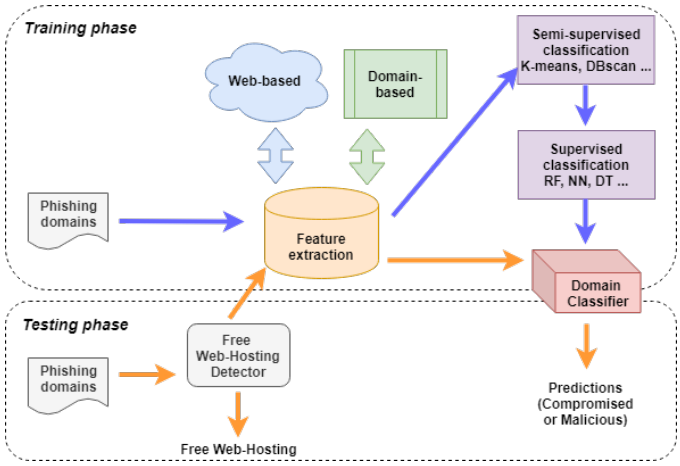


Figure 5.1: System architecture of our domain classifier. Types of hosting for phishing websites include: free web-hosting providers, compromised machines, and maliciously registered domains.

supervised learning, the model infers a function from labeled training data. Conversely, unsupervised learning does not require labeled data and instead models the probability density of inputs. Semi-supervised learning falls between supervised and unsupervised learning and typically makes use of a small amount of labeled data with a large amount of unlabeled data.

In this work, our machine learning task resembles semi-supervised learning, since we use a small amount of labeled data in order to cluster unlabeled data and increase the size of our training set.

5.2.2 Domain Classifier

Figure 5.1 shows the overall flow of our domain classifier. The feature extractor, shared by the training and testing phases, is the core of our system. It extracts automatically the values of the 19 features described below in Section 5.3.1. Specifically, the goal of the training phase is to obtain the feature values for each instance of the training domains. Those features are then used by semi-supervised classification algorithms: unlabeled domains that are clustering with labeled ones increase the size of our training set. Domains labeled in this way are then used by supervised classification algorithms to build classifiers. The goal of the testing phase is to label real phishing websites as either a compromised or a malicious domain.

In the testing phase, we first apply a free web-hosting detector using a list of known hosting websites. If the phishing website is not detected to be hosted on a free web-hosting

site, we move on to extract the 19 features from the domain using domain-based features as well as other publicly available web resources. Finally we apply a pre-trained model to classify a phishing domain as compromised or malicious.

5.3 Feature-based Machine Learning Framework

The present paper hinges on the set of high-level features detailed below in Section 5.3.1. Features 1 and 6 are inspired by work from APWG [4]. Features 2-10 are in common with [45]. Features 11-19 are the novel ones we propose here. Section 5.3.2 lists the machine learning algorithms we use.

5.3.1 Features

Features are organized into two categories. The first category (features 1 through 10) deals with the domain name of the phishing website. We deal with domain names rather than full URLs because we want our system to be usable widely and early, at domain registration time or by looking at the DNS traffic. The second category (features 11 through 19) exploits the history and internet presence of a domain and involves crawling the web for public information. In particular, we make use of The Wayback Machine, a digital archive of the Internet Archive[1] which allows retrieving the crawl history of a URL.

For domain based features, we make a further differentiation between which features are applied to the top level domain (TLD), primary domain, or subdomain. A domain name is presented in the structure of a hierarchical tree (*e.g.*, a.example.com). The top of the domain hierarchy is the root zone, which is usually represented as a dot. Below the root level is the top-level domain (TLD), a label after the rightmost dot in the domain name. Next to TLD is the second-level domain (2LD) (*e.g.*, .example.com), which can be directly registered from registrars such as GoDaddy if not yet occupied. In this work, we use TLDs to refer to the TLDs directly operated by registrars (like .com and .co.uk), and primary domains to refer to domains that can be obtained under TLDs. The registrant that owns the primary domain can also create subdomains, such as 3LDs and 4LDs, without asking permission from the registrar.

Below we also differentiate between continuous, ordinal and categorical features. Continuous features (also known as quantitative), most often involve a mapping into the reals, in which calculations such as mean require the full scale of the reals. Features with ordering

[1] http://archive.org/

but without scale are called ordinal features (also known as nominal). Features without ordering or scale are called categorical features.

Domain-based Features

TLD Features

(1) ***Freenom top level domain (TLD)*** (categorical): This feature checks whether the TLD belongs to Freenom[2]. Freenom provides free domain names on ".cf", ".gq", ".ml", ".tk" and ".ga" TLD. It is therefore not a surprise that several of these domains are abused [4].

(2) ***Previous malicious top level domain (TLD)*** (categorical): This feature checks the top three TLDs used to register malicious domains the month before, and checks whether the current TLD belongs to these top three. Price cuts happen periodically for groups of TLDs. It is then no surprise that they are periodically more abused.

Primary Domain Features

(3) ***Ratio of the longest English word*** (ordinal): This feature matches the longest English word that a domain name contains, and normalizes it by dividing by the length of the domain name. Attackers may generate pseudo-random names to avoid conflict with existing domains, or deliberately include readable words in the domain names to attract clicks from victims.

(4) ***Containing digits*** (categorical): This feature checks whether the domain name contains digits. Hao *et al.* [45] observe that spam and malicious phishing domains are more likely to use numerical characters than legitimate domains. Possible reasons may be that attackers add digits to generate several names from the same word or generate random names from a character set containing digits.

(5) ***Containing "-"*** (categorical): This feature checks whether the domain name contains any hyphens. Similarly, attackers can insert hyphens to break individual words or concatenate multiple words.

[2] https://www.freenom.com/

(6) ***Name length*** (ordinal): This feature checks the domain name length (number of characters). Attackers may create domains using a specific template, such as random strings of a given length.

(7) ***Edit distance to previous malicious domain*** (continuous): This feature examines the similarity of a domain with a set of known malicious domains. We derive a set of previously known malicious domains based on prior knowledge, compute the Levenshtein edit distances to the currently considered domain, and divide these edit distances by the length of the domain name for normalization. In our experiment, we use the data from a separate month to extract domains known to be malicious, and that do not overlap with any training or testing data.

(8) ***Partial match of brand name*** (ordinal): This feature matches the largest partial match of a brand name contained in a domain. We use a curated list of the top 50 brand names from Alexa. Given a brand name, we find the most similar substring in a domain name, and divide the number of similar characters by the length of the brand name. The ratio ranges from 0 to 1, where 0 indicates no match to any brand name and 1 indicates an exact match. Attackers may create domains which include the brand name or a variation to lend legitimacy and trick victims.

Subdomain Features

(9) ***Number of subdomain levels*** (ordinal): This feature counts the maximum number of subdomain levels of a domain. For example, if the domains "ebay.dll.verysecure.com" and "fedex.verysecure.com" host phishing attacks, where "ebay.dll" and "fedex" are subdomains of "verysecure.com", the maximum number of subdomain levels is two. Attackers may create domains with several misleading subdomain levels.

(10) ***Number of subdomains*** (ordinal): This feature counts the number of known subdomains belonging to a domain. In the above example, the number of known subdomains for "verysecure.com" is two ("ebay.dll" and "fedex"). Attackers may create several random subdomains under one domain in order to expand their attack.

Web-based Features

(11) ***Archived domain*** (categorical): This feature checks whether the domain has been archived on The Wayback Machine, which is part of The Internet Archive. An archived domain is more likely to be legitimately owned.

(12) ***Timespan in years the domain has been archived*** (ordinal): This feature checks the number of years the domain has been archived. A domain archived over a long period is more likely to be legitimately owned.

(13) ***Timespan in years of the domain's last archive before 2019*** (ordinal): This feature checks the time in years of the domain's last archive relative to 2019. For example if the domain's last archive was in 2017, then it has been 2 years since the last archive. A recently archived domain is more likely to be legitimately owned.

(14) ***Number of domain archive captures*** (ordinal): This feature checks the number of archive captures for a domain. A frequently captured domain is an indication of legitimate ownership.

(15) ***Reachable domain*** (categorical): This feature checks whether the domain returns a status code 200. Phishing sites usually last a short time. Most pages do not last more than a few days: attackers may avoid tracking by taking them down [10]. For this reason, legitimately owned domains are more likely to be reachable.

(16) ***Blocked domain*** (categorical): This feature checks whether access to the domain returns a status code 404, or whether the title or content of the page contains keywords such as "404", "down" or "under maintenance". Legitimately owned domains are less likely to be blocked. The reasoning is similar to the feature above.

(17) ***Redirected*** (categorical): This feature checks whether access to the domain redirects to a different domain. This constitutes a suspicious flag adding to other features.

(18) ***Alexa rank*** (ordinal): This feature checks whether the domain appears in Alexa[3] top 1 million, and if so records the rank. Domains in the Alexa ranking are more likely to be legitimately owned domains.

(19) ***Wildcard subdomain*** (categorical): This feature checks whether the domain is registered to accept all subdomains, known as a "wildcard" subdomain. A wildcard DNS record will match any request for an undefined domain name. It is then easy for attackers to advertise a working URL with a subdomain of their choosing despite not controlling the DNS entries of the domain itself. This may influence determination of whether a domain is compromised or malicious.

[3] http://www.alexa.com/

5.3.2 Machine Learning Algorithms

We compare five machine learning algorithms in training the domain classifier: K Neighbors (KN), Neural Networks (NN), Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), as well as a majority class baseline learner (DM) [17]. KN is a distance-based classifier and uses a distance measurement such as the Euclidean distance to predict a target based on the similarity with known targets. NN is a linear-based classifier and uses linear algebra and non-linear functions to build a model and predict a target. DT is a tree-based classifier and uses branching and thresholds on several features to predict a target. RF is an ensemble-based classifier which constructs a multitude of simple decision trees and predicts a target based on a voting scheme between the decision trees. Lastly NB is a probabilistic-based classifier and applies Bayes theorem with the "naive" assumption of conditional independence between every pair of features given the class label. All the machine learning algorithm implementations were taken from the *sklearn* package [4].

The reason we use a machine learning approach *vs* a heuristic approach is because machine learning algorithms find thresholds for classification automatically given large amounts of data. In addition, a machine learning approach is a more practical solution for real world problems, where data evolves continuously and where features are removed, added and modified. Machine learning also helps determine the strongest feature sets for faster classification. One common disadvantage to machine learning approaches is that some algorithms such as neural networks appear as a black box, making it hard to understand classification decisions. However other statistical and decision tree based algorithms are easier to understand and through careful analysis, classification decisions become more clear. Furthermore based on our experience, we find decision tree based algorithms work well in our classification domain. Lastly, through review feedback of our papers, we found reviewers expected more proof given a heuristic approach and often recommended the use of machine learning approaches instead.

5.4 Experimental setup

5.4.1 Phishing Domain Corpus

We have three sources for phishing websites: *APWG*, *PhishLabs* and *DeltaPhish*. Note only DeltaPhish provides an online source of their dataset. From each source, some of the phishing websites are labeled as being hosted on either a legitimate but compromised

[4] https://scikit-learn.org/

domain or on a maliciously registered domain. We give a brief description of the datasets from each of our sources below.

APWG

We have access to APWG's phish feed spanning several years and several thousand phishing websites. However APWG's phish feed are not labeled as compromised or malicious. APWG does provide phishing website confidence levels, and with such a large feed we focus on their phishing websites with 90% confidence or more. Note that although we have access to several years of phishing websites, we focus on recent phishing websites. This is because attacker strategies change overtime and we are interested in classifying current phishing domains. We also have access to APWG's malicious domain feed spanning about a 1 year period from January 2018 to the January 2019, resulting in about 25,000 malicious domains. This feed relies on both reporters submitting malicious domain URLs, as well as responders who monitor the list and act to suspend/take down the domains. Therefore we consider this list to be accurate. Analyzing these malicious domains, we find that most of the domains are malicious spam websites, such as fake stores and storefronts, rather than malicious phishing websites. By intersecting the malicious domains with domains in APWG's phish feed, we were left with 2,512 malicious phishing domains over a 1 year period.

PhishLabs

We received a list of 9,475 malicious domains collected from May 2018 to October 2018 from PhishLabs. The analysts reviewing phishing attacks at PhishLabs manually classify the domain as malicious if they believe it was created by the attackers for the purpose of phishing. Since this list was manually checked by the analysts at PhishLabs, we consider this list to be accurate. We also received a list of 17,427 confirmed phishing URLs from PhishLabs, collected from September 2018 to October 2018. As instructed by PhishLabs, intersecting the list of malicious domains and phishing URLs on "domain" gives the phishing URLs using malicious domains. The remaining phishing URLs are *most likely* those using compromised domains. Specifically, the 17,427 phishing URLs consist of 695 unique domains, 20 of which were in the list of malicious domains, resulting in a list of 675 likely compromised domains. However this list is less accurate, since the compromised domains were not manually inspected by the analysts at PhishLabs. Indeed, quick examination of the list of 675 likely compromised domains exposed a number of clearly malicious domains: "my-apple-id.cf", "doocs.gq", "my-sharepointofficedrive.tk", "outlook-livesl.cf". There are

Table 5.1: Train, Validate and Test sets.

Dataset Type	Dataset Name	Size	Distribution comp-mal	Timepan
Train set	*apwg-feed*	5744	3702-2042	Sept 2018-Oct 2018
Validate set	*apwg-mal*	495	0-495	Sept 2018-Oct 2018
	phishlabs-comp	669	669-0	Sept 2018-Oct 2018
Test set	*apwg-future-mal*	314	0-314	Nov 2018-Dec 2018
	phishlabs-mal	2804	0-2804	Sept 2018-Oct 2018
	deltaphish-comp	689	689-0	Dec 2015-Jan 2016

two reasons why these domains are malicious: 1) The TLDs belong to Freenom, a registry that provides free domains, so a lot of these domains are abused; 2) The domain name contains suspicious keywords such as "apple", "docs", "officedrive", and "outlook" mimicking legitimate brand names.

DeltaPhish

Since the list of compromised domains from PhishLabs contains some noise, we looked for other open source lists of compromised domains, and found a list shared by the authors of DeltaPhish [28]. The authors manually checked the list of 1,012 phishing websites hosted on 694 compromised domains that were retrieved from PhishTank from October 2015 to January 2016. Since this list was manually checked by the authors and used in their research, we consider this list to be accurate.

5.4.2 Evaluation Method, Target, and Metric

Evaluation Method

Our goal is to use APWG's large phish feed to train on, and to correctly classify future APWG malicious domains, while ensuring that compromised and malicious domains from other sources are also correctly classified. Our main issue with using APWG's large phish feed to train on is that these phishing websites are not labeled. To tackle this problem, we use semi-supervised learning to label some of the unlabeled cases. We explain our clustering approach in more detail below in Section 5.4.3.

A summary of our train, validate and test sets is shown in Table 5.1, where the distribution ordering is "compromised-malicious". Note that most of the dataset names represent datasets that are either only compromised or only malicious, denoted by the *comp* and *mal* postfix in the dataset name respectively, with the exception of the *apwg-feed* dataset. The *apwg-feed* dataset represents APWG's phish feed domains which were labeled using our clustering method.

Overall we try to keep our train, validate and test sets within the same timespan, since phishing attacks evolve over time. We find that we are constrained most by the *phishlabs-comp* dataset with timespan from September to October 2018, as discussed in Section 5.4.1. Therefore we kept all datasets within the timespan of September to October 2018, with two exceptions. The first exception is DeltaPhish's dataset which only spans from December 2015 to January 2016. The second exception is intended, since we want to test our classifier's predictions on APWG's future malicious domains. Therefore we include the *apwg-future-mal* dataset spanning 2 months from November to December 2018, which were reported directly after the phishing domains data our classifier is trained on. Note that all datasets shown in Table 5.1 are unique and that there are no common domains within datasets. We also use random sampling whenever we sample our data.

To optimize the algorithm parameters, we train on the train set and evaluate the results on the validate sets. We then train our model with the optimal parameters on the whole train set, and perform a final evaluation of our model with the test sets. Note that since our test sets are from three different sources, they have different distributions. This allows evaluation of the classifier robustness on a distribution of data it has not trained on, anticipating real world applications when the classifier will be deployed.

Evaluation Target

Our targets are compromised and malicious domains. Specifically, compromised domains are labeled as the positive class (1) while malicious domains are labeled as the negative class (-1). The reason for the use of 1 and -1 to label positive and negative classes rather than 1 and 0 is that we are interested in detecting both positive and negative classes, and not just the positive class. If we used the value 0 for our negative class, this would cause learning algorithms to have a harder time detecting malicious domains.

Evaluation Criteria

In our experiment, we use average recall as the evaluation criteria. Specifically average recall is an average of the true positive rate (TPR) and true negative rate (TNR): $AvgRecall = \frac{(TPR+TNR)}{2}$. The APWG's 2016 study [4] shows that both compromised and malicious cases were equally likely in their phishing database, therefore average recall is the evaluation measure of choice if all class distributions are equally likely. Note that if a balanced dataset is being evaluated, average recall is equivalent to accuracy.

5.4.3 Handling unlabeled data

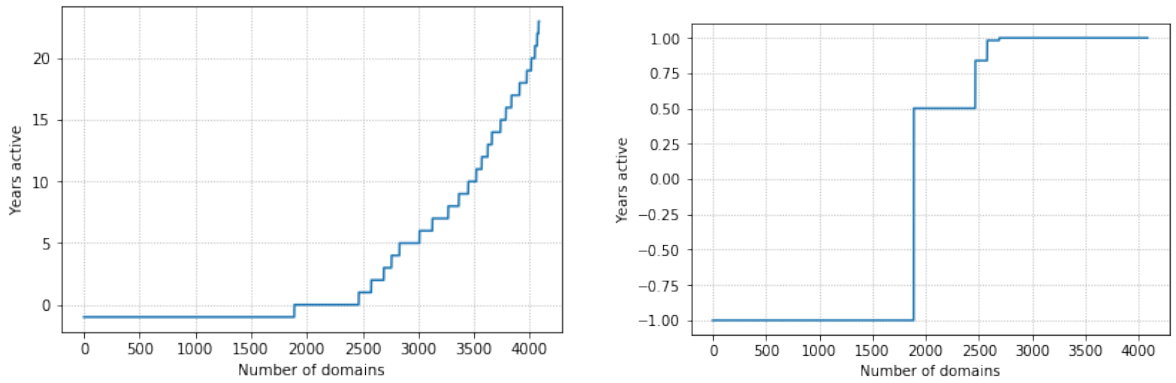
We use clustering in order to label some of APWG's unlabeled phish feed, referred to as *apwg-feed-unlabeled*. Specifically the *apwg-feed-unlabeled* dataset includes about 51,000 domains spanning from September 2018 to October 2018. We group this dataset with labeled domains from *apwg-mal* and *phishlabs-comp* datasets using clustering technique such as K-means, Birch and DBScan.

An unlabeled domain is given the label of the majority class within its cluster. We also apply additional constraints based on a maximum cluster entropy and a minimum number of domains in a cluster. We use entropy constraints in order to focus on labeling domains which are clearly compromised or malicious. We also use the number of domains in a cluster as a constraint to avoid labeling several unknown cases based on only a small number of labeled domains.

Since we expect APWG's phish feed to have balanced numbers of compromised and malicious cases, as shown in [4], we try several constraint combinations and clustering techniques. We achieved the most balanced labeling results (equal labels of compromised and malicious), using K-means clustering with a maximum cluster entropy of 0.2 and a minimum number of domains of 10.

As an example, if a cluster contains 49 malicious, 1 compromised and 100 unlabeled domains, the entropy of malicious *vs* compromised cases is approximately 0.14 which is smaller than 0.2, and the number of the malicious domains is larger than 10. Therefore this cluster would be accepted and the resulting 100 unlabeled domains would be labeled as malicious. The resulting domains from *apwg-feed-unlabeled* that are labeled after applying our clustering method and constraints are referred to as *apwg-feed* in Table 5.1.

We cluster unlabeled APWG phishing domains with PhishLabs' compromised domains as opposed to those from DeltaPhish because we knew that PhishLabs' domains contained noisy labels, as discussed in Section 5.4.1. In this way we take advantage of the noise in the data in order to limit model over-fitting, ensuring better model generalization [53]. We also take advantage of our clustering method to remove some of the noisy cases, since the constraints we set should discourage cases that occur infrequently. Conversely, we kept the compromised domains from DeltaPhish [28] to test on because these domains were manually checked by the authors as discussed in Section 5.4.1, and are therefore presumably better suited for evaluation of our classifier.



(a) Without calibration

(b) With calibration

Figure 5.2: **Left**: Before calibrating *Years active* on train set. **Right**: After calibrating *Years active* on train set.

5.4.4 Data preprocessing, transformation and reduction

Data preprocessing

In our datasets we use *nan* values as placeholder values for features *Years active*, *Years inactive*, *Number of captures*, *Redirected*, *Blocked*, and *Alexa rank*. Note that our use of *nan* does not indicate that this value is missing, or was not checked. We simply use it as a placeholder to later decide how best to represent *nan* values (e.g. mean, median, mode, or constant). In this case, our first preprocessing step is to replace all *nan* values with constant -1. This is because we found that the constant value -1 clearly differentiates from other feature values, and found that mean, median or mode would not be appropriate since these strategies are more appropriate as a way to fill missing values. We perform this step on all train, validate and test sets. As our last preprocessing step we normalize the data with mean and standard deviation using the *StandardScalar* funtion in *sklearn*. Note that we use the mean and standard deviation of the train set to normalize the validation and test sets.

Data transformation

We use calibration as a feature transformation in order to adapt the scale of several of our quantitative features, including *Years active*, *Years inactive*, *Number of captures*, *Alexa rank*, *Number of subdomain levels*, *Number of subdomains*, *Match brand name*, and *Name length*. Feature calibration is understood as a supervised feature transformation adding a meaningful scale-carrying class information to arbitrary features. For example, we show the calibration results for *Years active* in Figure 5.2. Performing calibration has several

advantages, allowing the learning algorithms to choose whether to treat a feature as categorical, ordinal or quantitative. To perform calibration, we use the *IsotonicRegression* function in *sklearn*. Note, similar to normalizing in our preprocessing step, we use the *IsotonicRegression* function that was fitted on the training data to transform the validate and test sets. Out of bound cases are "clipped" so that the predicted y-values are set to the value corresponding to the nearest train interval endpoint.

Data reduction

Since our goal is to evaluate the effectiveness of our feature set, we did not use data reduction strategies such as PCA. In addition, since we only have 19 features, issues of run time or the "curse of dimensionality" is generally not a problem until the number of features is much larger. Note that the existence of highly correlated features such as *Archived*, *Years active*, *Years inactive*, and *Number of captures*, as well as features *Reachable*, *Blocked* and *Redirected*, should not affect the predictive accuracy of learning models.

5.4.5 Handling class imbalance

We used a balanced dataset for both training and validation (*i.e.* equal number of compromised and malicious domains), for two reasons. First, a balanced train set is more realistic since in the real-world scenario, the volume of malicious and compromised cases are similar. For example in 2016, APWG reports that almost 49% of domains are malicious, while the rest are compromised [4]. Second, machine-learning classifiers have difficulty coping with imbalanced train sets as they are sensitive to the proportions of the different classes. As a consequence, these algorithms tend to favor the majority class, creating misleading accuracy.

To handle class imbalance, we use random undersampling as a technique for both training and validation, since this limits the machine learning algorithm the most. Specifically, we remove a random number of the majority class until both classes are equally represented. By undersampling, we ensure a worst case scenario rather than the best case scenario of oversampling.

Table 5.2: Optimal *sklearn* model hyperparameters.

ML Algorithm	Param.	Tuned Param.	Train Acc. (%)	Val Acc. (%)
KN	n_neighbors	14	100.00	82.53
	weights	'distance'		
	p	2		
NN	hidden_layer_sizes	200	99.05	85.66
	activation	'identity'		
	solver	'sgd'		
	batch_size	100		
	learning_rate_init	0.0001		
DT	criterion	'gini'	100.00	86.26
	max_depth	None		
	max_features	'auto'		
RF	n_estimators	100	100.00	86.06
	max_depth	1		
	max_features	'auto'		
NB	-	-	100.00	52.93
DM	-	-	51.59	51.51

Table 5.3: Evaluation on test sets.

	apwg-future-mal	*phishlabs-mal*	*deltaphish-comp*	
Algorithm	Acc. (%)			Balanced Acc. (%)
KN	95.54	93.15	69.81	82.08
NN	95.54	90.87	88.97	91.09
DT	**95.85**	**91.98**	**91.73**	**92.82**
RF	94.90	88.02	**91.73**	91.59
NB	92.04	89.37	13.35	52.03
DM	52.23	50.36	49.35	50.32

5.5 Experimental evaluation

5.5.1 Multi-Dataset Evaluation

We tune the hyperparameters for each model by training on the train set and picking the hyperparameters that give the best accuracy on the validation set. Table 5.2 shows the optimal hyperparameters for each model.

We show our evaluation results in Table 5.3. Algorithm DT achieves the highest accuracy for all datasets. We also include an overall balanced accuracy result. It takes an average of the malicious datasets before averaging this result with the compromised dataset, ensuring equal values for compromised accuracy and malicious accuracy. Specifically, if the accuracy of *apwg-future-mal*, *phishlabs-mal*, and *deltaphish-comp* are represented as *mal1*, *mal2*, and *comp1* respectively, then $BalancedAcc = \frac{\frac{(mal1+mal2)}{2}+comp1}{2}$.

Appropriately, DT also has the highest *BalancedAcc*. However upon closer inspection, we found DT decision rules to be simple and not robust, and achieved high accuracy only by exploiting the *Archived* and *Previous malicious TLD* feature. We found that by removing *Previous malicious TLD* as a feature, DT performance on *deltaphish-comp* dropped to below 70%, whereas the RF classifier accuracy for example remained about the same. This may be because the RF classifier is an ensemble of several decision trees, and is more

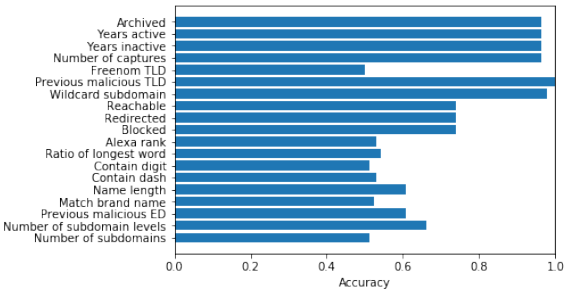


Figure 5.3: RF accuracy with individual features trained on a balanced *apwg-feed* dataset.

robust.

5.5.2 Learning with Individual Features

The evaluations in Section 5.5.1 were obtained by using the whole feature set (19 features in total). In this section, we evaluate the contribution of each individual feature to the overall accuracy. We only report the result with Random Forest classifier under a balanced *awpg-feed* train set in Figure 5.3.

From Figure 5.3 we find the strongest feature to be *Previous malicious TLD*. This is because *awpg-feed* was labeled using our clustering method discussed in Section 5.4.3. This resulted in only labeling clusters of malicious domains where TLDs were *Previous malicious TLDs*. This makes it trivial to separate the malicious and compromised cases in *awpg-feed* with just this feature, even though in practice not all malicious domains use previously malicious TLDs. The next strongest feature is *Wildcard subdomain* which, due to our clustering method, is very descriptive of our compromised domains. Although in practice not all compromised domains have a wildcard subdomain, it has been found in [61] that attackers exploit the wildcard DNS records of compromised domains. The next strongest features are the archive features *Archived*, *Years active*, *Years inactive*, and *Number of captures*, which are all highly correlated. After this are the features *Reachable*, *Redirected*, and *Blocked*, again highly correlated. Generally these two groups of features are descriptive of compromised cases, since they capture a domains history and internet presence.

It is interesting to note that *FreenomTLD*, *Contain digit*, and *Contain dash* do no better than random guessing. In our previous work we found that these features were stronger when trained on PhishLabs' malicious dataset. Indeed these characteristics of malicious domains are reported in other research as well [4, 45]. However APWG's malicious domain feed has very few domains with these characteristics. This may be because APWG use different criteria to consider a domain malicious.

Table 5.4: Top 10 Probable Malicious and Compromised Domains After Classification.

Malicious		**Compromised**	
Prob.	Domain	Prob.	Domain
0.9148	vacationrentalreview-g908802.top	0.8961	000a.biz
0.9148	radionovidade.top	0.8961	kop.kz
0.9148	poypol.top	0.8955	emk01.com
0.9148	sl1s2.top	0.8955	klittle.com
0.9148	j7g3.top	0.8955	kli.net
0.9148	2018sexuk.men	0.6937	emt.org
0.9148	atyha.men	0.6907	emv3.com
0.9148	removalmoval.org	0.6686	kl.com.ua
0.9148	inialive.tw	0.6686	eng.ph
0.9148	rental28009.live	0.6686	acts24.com

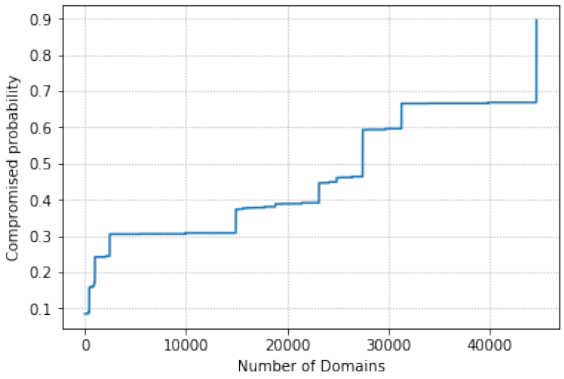


Figure 5.4: Sorted compromised probabilities of 44,668 phishing domains.

Despite these strength differences and over-representations of features, our RF classifier predicts several real world datasets from different sources with high accuracy, as shown in Section 5.5.1. The use of a validation set to tune our models and avoid overfitting the strongest features, as well as of ensemble methods such as RF, creating several simple decision trees using different features and a voting scheme, prevented overfitting.

5.6 Analysis

In this section, we present our analysis of the proportions and confidence of classifying unlabeled phishing domains as either compromised or malicious, using our RF domain classifier. Specifically we use the rest of the unlabeled APWG phish feed from September

to October 2018 which represents 50,412 domains, of which 5,744 were labeled during clustering as discussed in Section 5.4.3, leaving 44,668 still unlabeled.

After using our RF domain classifier to predict the unlabeled domains, we find that 38% of the domains are malicious while 62% are compromised. In contrast APWG's previous study of classifying phishing websites from 2015-2016 found that 49% of domains are malicious. There may be a few reasons that explain this difference: first we classify hosting sites as compromised or malicious while AWPG detects domains as malicious and assumes the rest are compromised. Second, we believe that the criteria used in [4] may be too aggressive and will for example classify any site that is hacked almost immediately upon creation as malicious.

Next we look at the confidence, or probability of our classification. In Table 5.4 we include the top 10 list of most probable compromised domains and malicious domains. We also include the full list of sorted compromised probabilities in Figure 5.4, where for example, the last domain (the 44,668th domain) with the highest probability of being compromised (about 0.9 probability) can equivalently be read as having 0.1 probability of being malicious. Overall we find that over 60% of domains have a probability greater than 65% of being correctly classified. Equivalently, less than 40% have a probability between 65% and 50% of being correctly classified.

5.7 Discussion

5.7.1 Runtime Performance

Our framework is composed of a labeling phase, training phase and a testing phase (Figure 5.1). The labeling and training phase can be done periodically offline. Users then experience no time delay in this stage. When deployed, the testing phase is conducted online as each phishing domain arrives.

All our experiments were conducted on a standard computer with a 2.10GHz processor and 14.7GB of available RAM. The free web-hosting detector caused no apparent delay in the work flow. The module with critical time issue is the feature extractor. For example fetching archive features take the longest time since we use a crawler and wait 10 seconds for the page to load. The average run time of the feature extractor module is 30 seconds per domain. Various measures exist for improving the time performance of the feature extraction module. For example we used 5 threads in the experiments in order to process 5 domains in parallel. This way we process 5 domains on average after 30 seconds (or 1

every 6 seconds). Another essential strategy we use is caching. For example caching the query and result web-based features improves runtime performance.

Once the feature values have been extracted, applying the pre-trained machine-learning model consumes a trivial amount of time.

5.7.2 Limitations

One limitation to our approach is that The Internet Archive is unintentionally internationally biased [87] towards North-American websites. For example, our approach may not work as well for Chinese phishing websites, since archive is less likely to have captured Chinese websites.

One way around this limitation would be to include search-engine results as an indicator of domain presence and reputation. For example, in the Google search engine, one can search for "inurl:example.com" which returns indexed URLs with the string "example.com". To address international bias, one could then use several international search engines. However this solution may not be scalable since search engines usually place a limit on the number of requests.

Another limitation to our approach would be if attackers intentionally register a domain with history. For example, an owner may have a domain for several years until dropping the domain, at which point an attacker may be ready to pick it up. However this also limits the attacker, since domains with history may be more expensive, and may take more time to identify and acquire. This also prevents attackers from choosing a domain name that is misleading, or from choosing a domain name resembling a brand name to lend legitimacy.

A previous way around this limitation would be to check registration history to see whether the domain has recently changed owners. However this approach has become increasingly difficult to automate, and the information may not be publicly available. Another option around this limitation would be to use the captures in archive and compare the HTML of a domain over time, to see whether there are any recent and sudden changes in the HTML. HTML that has suddenly changed and is not relevant to the previous content implies that the domain has probably passed on to a new owner.

5.7.3 Detecting Malicious Domains

Our domain classifier can also be used to detect malicious domains from normal domains, where normal domains are those which are not hosting phishing attacks. However our classifier cannot be used to detect compromised domains from normal domains: the features

we use to detect compromised domains are those that lend legitimacy to a domain such as domain history, which both compromised and normal domains share.

5.8 Conclusion

In this chapter, we classify domains of known phishing websites as either hosted on legitimately owned (compromised) domains or on maliciously registered domains. Our classifier uses only publicly available information and achieves acceptable runtime performance, making our solution widely usable.

We also present a novel semi-supervised method to label some of APWG's large phish feed thereby increasing the size of our training set. Our semi-supervised technique is not limited to phishing domain classification, and can be reused in other machine learning predictive tasks, even if the labeled data being used is noisy. Overall our solution achieves high levels of accuracy on three datasets from three different sources.

Based on our evaluation of five different machine learning algorithms, we find the Random Forest (RF) classifier to be the most robust. However, based on our analysis, we find our RF classifier has low confidence in its predictions, since 40% of the predictions have a probability between 50% and 65% of being correctly classified. To address this in future work, we would like to continue experimenting with our semi-supervised method to form clusters that include a more varied train set of APWG's phish feed. This may be most easily achieved by manual checking and labeling a few hundred of AWPG's phish feed.

Chapter 6

Conclusion

The problem of phishing attacks is a continued threat to society. Efforts in understanding the phishing ecosystem can offer insight into effective anti-phishing strategies. Analysis of click-through and timespan shows that although anti-phishing efforts are stronger than anti-malware efforts, the greatest click activity still occurs hours before the reported date. Our analysis also reveals that several phishing attacks are active past 3 months. This insight adds more weight to the argument that we are not as good at phishing detection as we assumed we were. For furture work it would be important to take anti-phishing click activity into consideration, since companies and products crawl phishing websites, sometimes through the Bitly link, as part of their anti-phishing strategy, affecting the click analytics.

Comparing similarity methods shows that upwards of 94% of phishing attacks can quickly be flagged as replicates. This insight helps anti-phishing efforts to automatically flag phishing attack replicas and focus on novel phishing attacks. For future work it would be interesting to study more similarity methods and comparing tradeoffs. For example there are algorithms which approximate the tree edit distance. We are also starting to see attackers making variations between phishing instances by using dummy tags added to the DOM. We could also consider integrating an image comparison algorithm. The goal of this idea would be that the dummy tags would not affect the screen appearance of the variations for phishing victims. In this way, a combination of several similarity methods would be much stronger than using just one of the methods.

Domain classification of large phish feeds shows that compromised cases are more common than malicious cases. This insight reveals that more efforts should be put on prevention and detection of compromised cases. For future work it would be interesting to analyze a large number of the domains that are classified as compromised as a way to find additional descriptive features to identify compromised cases. For example, from several

compromised phishing domains, one could analyze the phishing URL paths of the domains. From the analysis, one could determine if there are any keywords in the path which suggest that the server has been compromised, since attackers are usually limited in where they can host a phishing website on compromised servers.

Understanding the phishing ecosystem allows us to take a step back and understand what is working, but also what is lacking. It also allows us to discover opportunities to improve defenses against phishing attacks as well as other internet attacks through comparative analysis. Through our comparative click analysis of phishing and malware attacks, we discovered how much better anti-phishing efforts were. This may also extend to scam attacks for which efforts are also lacking [14]. These differences in efforts are likely because brands put a lot of effort into anti-phishing strategies, since phishing attacks affect the public trust and public image of brands by impersonating them, causing brands to lose money. Although malware and scam attacks may not cause brands to lose as much money, they are still just as important as phishing attacks to defend against so as to protect end users. Overall these findings offer some guidance for future effective anti-phishing strategies and present an opportunity to make big improvements in efforts against other internet attacks.

References

[1] Alexa. Top 500 Sites in Each Country. http://www.alexa.com/topsites/countries.

[2] Mohamed Alsharnouby, Furkan Alaca, and Sonia Chiasson. Why phishing still works: User strategies for combating phishing attacks. *International Journal of Human-Computer Studies*, 82:69–82, 2015.

[3] Anti-Phishing Working Group. Phishing Activity Trends Report 1st Quarter in 2016. http://docs.apwg.org/reports/apwg_trends_report_q1_2016.pdf.

[4] Anti-Phishing Working Group. Global Phishing Survey: Trends and Domain Name Use in 2016. http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf, 2017.

[5] Anti-Phishing Working Group. Phishing Activity Trends Report 1st Half 2017. http://docs.apwg.org/reports/apwg_trends_report_h1_2017.pdf, 2017.

[6] Demetris Antoniades, Iasonas Polakis, Georgios Kontaxis, Elias Athanasopoulos, Sotiris Ioannidis, Evangelos P Markatos, and Thomas Karagiannis. we. b: The web of short urls. In *Proceedings of the 20th international conference on World Wide Web*, pages 715–724. ACM, 2011.

[7] Simone Aonzo, Alessio Merlo, Giulio Tavella, and Yanick Fratantonio. Phishing attacks on modern android. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1788–1801. ACM, 2018.

[8] APWG. APWG Phishing Attack Trends Reports. https://www.antiphishing.org/resources/apwg-reports/.

[9] APWG. Phishing Activity Trends Report 1st Quarter in 2018. bit.ly/2HfK0Ik.

[10] APWG. Phishing Activity Trends Report 3rd Quarter in 2018. bit.ly/2VTVYuh.

[11] APWG. Workshop agenda 2017. https://www.apwg.eu/apwg-events/ecrime2017eu/agenda.

[12] Calvin Ardi and John Heidemann. Auntietuna: Personalized content-based phishing detection. In *NDSS Usable Security Workshop (USEC)*, 2016.

[13] Omid Asudeh and Mathew Wright. Poster: phishing website detection with a multi-phase framework to find visual similarity. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1790–1792. ACM, 2016.

[14] Emad Badawi, Guy-Vincent Jourdan, Gregor Bochmann, Iosif-Viorel Onut, and Jason Flood. The game hack scam. In *International Conference on Web Engineering*, pages 280–295. Springer, 2019.

[15] Alejandro Correa Bahnsen, Eduardo Contreras Bohorquez, Sergio Villegas, Javier Vargas, and Fabio A González. Classifying phishing urls using recurrent neural networks. In *2017 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–8. IEEE, 2017.

[16] Morvareed Bidgoli, Bart P Knijnenburg, and Jens Grossklags. When cybercrimes strike undergraduates. In *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–10. IEEE, 2016.

[17] Albert Bifet, Ricard Gavaldà, Geoff Holmes, and Bernhard Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT Press, 2018.

[18] Bitly Blog. How do I set up a Branded Short Domain (BSD). https://support.bitly.com/hc/en-us/articles/230898888-How-do-I-set-up-a-Branded-Short-Domain-BSD.

[19] Bitly Blog. How to Create And Customize Your Bitlinks. https://bitly.com/blog/bitly-basics-bitlinks/.

[20] Pranjal S Bogawar and KK Bhoyar. Soft computing approaches to classification of emails for sentiment analysis. In *Proceedings of the International Conference on Informatics and Analytics*, page 22. ACM, 2016.

[21] Karl Bringman, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Tree edit distance cannot be computed in strongly subcubic time (unless apsp can). In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1190–1206. SIAM, 2018.

[22] Onur Catakoglu, Marco Balduzzi, and Davide Balzarotti. Automatic extraction of indicators of compromise for web applications. In *Proceedings of the 25th International Conference on World Wide Web*, pages 333–343. International World Wide Web Conferences Steering Committee, 2016.

[23] Ee Hung Chang, Kang Leng Chiew, San Nah Sze, and Wei King Tiong. Phishing detection via identification of website identity. In *2013 International Conference on IT Convergence and Security, ICITCS 2013*, pages 1–4. IEEE, 2013.

[24] 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.

[25] Weimin Chen. New algorithm for ordered tree-to-tree correction problem. *Journal of Algorithms*, 40(2):135–158, 2001.

[26] Sidharth Chhabra, Anupama Aggarwal, Fabricio Benevenuto, and Ponnurangam Kumaraguru. Phish social: the phishing landscape through short urls. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, pages 92–101. ACM, 2011.

[27] Coding Horror. URL Shortening: Hashes In Practice. https://blog.codinghorror.com/url-shortening-hashes-in-practice/.

[28] Igino Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli. Deltaphish: Detecting phishing webpages in compromised websites. In *European Symposium on Research in Computer Security*, pages 370–388. Springer, 2017.

[29] Qian Cui, Guy-Vincent Jourdan, Gregor V Bochmann, Russell Couturier, and Iosif-Viorel Onut. Tracking phishing attacks over time. In *Proceedings of the 26th International Conference on World Wide Web*, pages 667–676. International World Wide Web Conferences Steering Committee, 2017.

[30] Qian Cui, Guy-Vincent Jourdan, Gregor V Bochmann, Iosif-Viorel Onut, and Jason Flood. Phishing attacks modifications and evolutions. In *European Symposium on Research in Computer Security*, pages 243–262. Springer, 2018.

[31] Rachna Dhamija and J Doug Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 77–88. ACM, 2005.

[32] Rachna Dhamija, J Doug Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM, 2006.

[33] Lauren Dugan. Bit.ly Pro Is Now Free! http://www.adweek.com/digital/bit-ly-pro-is-now-free-get-your-custom-short-domain-today/, Jun 2011.

[34] Yahia Elsayed and Ahmed Shosha. Large scale detection of idn domain name masquerading. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–11. IEEE, 2018.

[35] Anthony Y Fu, Liu Wenyin, and Xiaotie 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, 2006.

[36] Norbert Fuhr and Kai Großjohann. Xirql: A query language for information retrieval in xml documents. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 172–180. ACM, 2001.

[37] Guang-Gang Geng, Xiao-Dong Lee, Wei Wang, and Shian-Shyong Tseng. Favicon - a clue to phishing sites detection. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–10, Sept 2013.

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

[39] Torsten Grabs. Generating vector spaces on-thefly for flexible xml retrieval. In *In [1*. Citeseer, 2002.

[40] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @ spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 27–37. ACM, 2010.

[41] Neha Gupta, Anupama Aggarwal, and Ponnurangam Kumaraguru. bit. ly/malicious: Deep dive into short url based e-crime detection. In *Electronic Crime Research (eCrime), 2014 APWG Symposium on*, pages 14–24. IEEE, 2014.

[42] Srishti Gupta and Ponnurangam Kumaraguru. Emerging phishing trends and effectiveness of the anti-phishing landing page. In *Electronic Crime Research (eCrime), 2014 APWG Symposium on*, pages 36–47. IEEE, 2014.

[43] Xiao Han, Nizar Kheir, and Davide Balzarotti. Phisheye: Live monitoring of sandboxed phishing kits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1402–1413. ACM, 2016.

[44] Xiao Han, Nizar Kheir, and Davide Balzarotti. Phisheye: Live monitoring of sandboxed phishing kits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1402–1413. ACM, 2016.

[45] Shuang Hao, Alex Kantchelian, Brad Miller, Vern Paxson, and Nick Feamster. Predator: proactive recognition and elimination of domain abuse at time-of-registration. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1568–1579. ACM, 2016.

[46] Grant Ho, Aashish Sharma, Mobin Javed, Vern Paxson, and David Wagner. Detecting credential spearphishing in enterprise settings. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 469–485, 2017.

[47] Artsiom Holub and Jeremiah O'Connor. Coinhoarder: Tracking a ukrainian bitcoin phishing ring dns style. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–5. IEEE, 2018.

[48] Hang Hu and Gang Wang. End-to-end measurements of email spoofing attacks. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1095–1112, 2018.

[49] Ankit Kumar Jain and B. B. Gupta. Phishing detection: Analysis of visual similarity based approaches. *Security and Communication Networks*, 2017:20, 2017.

[50] Vasileios Kandylas and Ali Dasdan. The utility of tweeted urls for web search. In *Proceedings of the 19th international conference on World wide web*, pages 1127–1128. ACM, 2010.

[51] Panagiotis Kintis, Najmeh Miramirkhani, Charles Lever, Yizheng Chen, Rosa Romero-Gómez, Nikolaos Pitropakis, Nick Nikiforakis, and Manos Antonakakis. Hiding in plain sight: A longitudinal study of combosquatting abuse. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 569–586. ACM, 2017.

[52] Florian Klien and Markus Strohmaier. Short links under attack: geographical analysis of spam in a url shortener network. In *Proceedings of the 23rd ACM conference on Hypertext and social media*, pages 83–88. ACM, 2012.

[53] Anders Krogh and John A Hertz. Generalization in a linear perceptron in the presence of noise. *Journal of Physics A: Mathematical and General*, 25(5):1135, 1992.

[54] Laendercode. List: The two-letter country code / country abbreviation. https://laendercode.net/en/2-letter-list.html.

[55] Sophie Le Page, Gregor v Bochmann, Qian Cui, Jason Flood, Guy-Vincent Jourdan, and Iosif-Viorel Onut. Using ap-ted to detect phishing attack variations. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6. IEEE, 2018.

[56] Sophie Le Page and Guy-Vincent Jourdan. Domain classifier: Compromised machines vs malicious registrations. In *19th Internation Conference on Web Engineering*, 2019.

[57] Sophie Le Page and Guy-Vincent Jourdan. Victim or attacker? multi-dataset domain classification of phishing attacks. In *2019 17th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2019.

[58] Sophie Le Page, Guy-Vincent Jourdan, Gregor V Bochmann, Jason Flood, and Iosif-Viorel Onut. Using url shorteners to compare phishing and malware attacks. In *APWG Symposium on Electronic Crime Research (eCrime), 2018*, pages 1–13. IEEE, 2018.

[59] Sophie Le Page, Guy-Vincent Jourdan, Gregor V Bochmann, Iosif-Viorel Onut, and Jason Flood. Domain classifier: Compromised machines versus malicious registrations. In *International Conference on Web Engineering*, pages 265–279. Springer, 2019.

[60] Tongxin Li, Xueqiang Wang, Mingming Zha, Kai Chen, XiaoFeng Wang, Luyi Xing, Xiaolong Bai, Nan Zhang, and Xinhui Han. Unleashing the walking dead: Understanding cross-app remote infections on mobile webviews. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 829–844. ACM, 2017.

[61] Daiping Liu, Zhou Li, Kun Du, Haining Wang, Baojun Liu, and Haixin Duan. Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 537–552. ACM, 2017.

[62] Wenyin Liu, Gang Liu, Bite Qiu, and Xiaojun Quan. Antiphishing through Phishing Target Discovery. *IEEE Internet Computing*, 16(2):52–61, 2012.

[63] Christian Ludl, Sean McAllister, Engin Kirda, and Christopher Kruegel. On the effectiveness of techniques to detect phishing sites. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 20–39. Springer, 2007.

[64] Federico Maggi, Alessandro Frossi, Stefano Zanero, Gianluca Stringhini, Brett Stone-Gross, Christopher Kruegel, and Giovanni Vigna. Two years of short urls internet measurement: security threats and countermeasures. In *proceedings of the 22nd international conference on World Wide Web*, pages 861–872. ACM, 2013.

[65] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141–150. ACM, 2007.

[66] Samuel Marchal and N Asokan. On designing and evaluating phishing webpage detection techniques for the real world. In *11th {USENIX} Workshop on Cyber Security Experimentation and Test ({CSET} 18)*, 2018.

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

[68] MetaFilter. We want 'em shorter. https://www.metafilter.com/8916/We-want-em-shorter, 2001.

[69] Tyler Moore and Richard Clayton. An empirical analysis of the current state of phishing attack and defence. In *WEIS*, 2007.

[70] Tyler Moore and Richard Clayton. Evil searching: Compromise and recompromise of internet hosts for phishing. In *Financial Cryptography*, pages 256–272. Springer, 2009.

[71] Tyler Moore and Richard Clayton. The impact of incentives on notice and take-down. In *Managing Information Risk and the Economics of Security*, pages 199–223. Springer, 2009.

[72] Raj Kumar Nepali and Yong Wang. You look suspicious!!: Leveraging visible attributes to classify malicious short urls on twitter. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pages 2648–2655. IEEE, 2016.

[73] Alexander Neumann, Johannes Barnickel, and Ulrike Meyer. Security and privacy implications of url shortening services. In *Proceedings of the Workshop on Web 2.0 Security and Privacy*, 2010.

[74] Nick Nikiforakis, Federico Maggi, Gianluca Stringhini, M Zubair Rafique, Wouter Joosen, Christopher Kruegel, Frank Piessens, Giovanni Vigna, and Stefano Zanero. Stranger danger: exploring the ecosystem of ad-based url shortening services. In *Proceedings of the 23rd international conference on World wide web*, pages 51–62. ACM, 2014.

[75] Adam Oest, Yeganeh Safei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Gary Warner. Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–12. IEEE, 2018.

[76] Mateusz Pawlik and Nikolaus Augsten. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157–173, 2016.

[77] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, Nagendra Modadugu, et al. The ghost in the browser: Analysis of web-based malware. *HotBots*, 7:4–4, 2007.

[78] Gowtham Ramesh, Ilango Krishnamurthi, and K. Sampath Sree Kumar. An efficacious method for detecting phishing webpages through target domain identification. *Decision Support Systems*, 61(1):12–22, 2014.

[79] A. P E Rosiello, Engin Kirda, Christopher Kruegel, and Fabrizio Ferrandi. A layout-similarity-based approach for detecting phishing pages. In *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks, SecureComm*, pages 454–463, Nice, 2007.

[80] Stefan Schwarz, Mateusz Pawlik, and Nikolaus Augsten. A new perspective on the tree edit distance. In *International Conference on Similarity Search and Applications*, pages 156–170. Springer, 2017.

[81] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. In *Sixth conference on email and anti-spam (CEAS)*. California, USA, 2009.

[82] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. 2009.

[83] Gunikhan Sonowal and KS Kuppusamy. Masphid: a model to assist screen reader users for detecting phishing sites using aural and visual similarity measures. In *Proceedings of the International Conference on Informatics and Analytics*, page 87. ACM, 2016.

[84] Samaneh Tajalizadehkhoob, Tom Van Goethem, Maciej Korczyński, Arman Noroozian, Rainer Böhme, Tyler Moore, Wouter Joosen, and Michel van Eeten. Herding vulnerable cats: a statistical approach to disentangle joint responsibility for web security in shared hosting. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 553–567. ACM, 2017.

[85] Kshitij Tayal and Vadlamani Ravi. Particle swarm optimization trained class association rule mining: Application to phishing detection. In *Proceedings of the International Conference on Informatics and Analytics*, page 13. ACM, 2016.

[86] Joe Tekli, Richard Chbeir, and Kokou Yetongnon. An overview on xml similarity: Background, current trends and future directions. *Computer science review*, 3(3):151–173, 2009.

[87] Mike Thelwall and Liwen Vaughan. A fair history of the web? examining country balance in the internet archive. *Library & information science research*, 26(2):162–176, 2004.

[88] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, et al. Data breaches, phishing, or malware?: Understanding the risks of stolen credentials. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1421–1434. ACM, 2017.

[89] Ke Tian, Steve TK Jan, Hang Hu, Danfeng Yao, and Gang Wang. Needle in a haystack: tracking down elite phishing domains in the wild. In *Proceedings of the Internet Measurement Conference 2018*, pages 429–442. ACM, 2018.

[90] Javier Vargas, Alejandro Correa Bahnsen, Sergio Villegas, and Daniel Ingevaldson. Knowing your enemies: Leveraging data analysis to expose phishing patterns against a major us financial institution. In *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–10. IEEE, 2016.

[91] Veriform. VERIFROM - Keep Your Brand Safe. https://www.verifrom.com/.

[92] De Wang, Shamkant B Navathe, Ling Liu, Danesh Irani, Acar Tamersoy, and Calton Pu. Click traffic analysis of short url spam on twitter. In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, pages 250–259. IEEE, 2013.

[93] Maarten Wullink, Moritz Muller, Marco Davids, Giovane CM Moura, and Cristian Hesselman. Entrada: enabling dns big data applications. In *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–11. IEEE, 2016.

[94] WWW. HTML Tag Set. https://www.w3.org/TR/html-markup/elements.html.

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

[96] Guang Xiang and Jason I Hong. A hybrid phish detection approach by identity discovery and keywords retrieval. In *Proceedings of the 18th international conference on World wide web*, pages 571–580. ACM, 2009.

[97] Ting-Fang Yen, Victor Heorhiadi, Alina Oprea, Michael K Reiter, and Ari Juels. An epidemiological study of malware encounters in a large enterprise. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1117–1130. ACM, 2014.

[98] Soojin Yoon, Jeongeun Park, Changkuk Choi, and Seungjoo Kim. Shrt: New method of url shortening including relative word of target url. *The Journal of Korean Institute of Communications and Information Sciences*, 38(6):473–484, 2013.

[99] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

[100] Weifeng Zhang, Hua Lu, Baowen Xu, and Hongji Yang. Web phishing detection based on page spatial layout similarity. *Informatica*, 37(3), 2013.

[101] Yue Zhang, Jason Hong, and Cranor Lorrie. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web*, pages 639–648, Banff, AB, 2007.

[102] Leah Zhang-Kennedy, Elias Fares, Sonia Chiasson, and Robert Biddle. Geo-phisher: the design and evaluation of information visualizations about internet phishing trends. In *2016 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–12. IEEE, 2016.

[103] Chaoshun Zuo and Zhiqiang Lin. Smartgen: Exposing server urls of mobile apps with selective symbolic execution. In *Proceedings of the 26th International Conference*

on World Wide Web, pages 867–876. International World Wide Web Conferences Steering Committee, 2017.