Phishing Vs. Legit: Comparative Analysis of Client-Side Resources of Phishing and Target Brand Websites

Kyungchan Lim
University of Tennessee
Knoxville, TN, USA
klim7@utk.edu

Jaehwan Park
University of Tennessee
Knoxville, TN, USA
jpark127@utk.edu

Doowon Kim
University of Tennessee
Knoxville, TN, USA
doowon@utk.edu

ABSTRACT

Phishing attacks have persistently remained a prevalent and widespread cybersecurity threat for several years. This leads to numerous endeavors aimed at comprehensively understanding the phishing attack ecosystem, with a specific focus on presenting new attack tactics and defense mechanisms against phishing attacks. Unfortunately, little is known about how client-side resources (e.g., JavaScript libraries) are used in phishing websites, compared to those in their corresponding legitimate target brand websites. This understanding can help us gain insights into the construction and techniques of phishing websites and phishing attackers' behaviors when building phishing websites.

In this paper, we gain a deeper understanding of how client-side resources (especially, JavaScript libraries) are used in phishing websites by comparing them with the resources used in the legitimate target websites. For our study, we collect both client-side resources from phishing websites and their corresponding legitimate target brand websites for 25 months: 3.4M phishing websites (1.1M distinct phishing domains). Our study reveals that phishing websites tend to employ more diverse JavaScript libraries than their legitimate websites do. However, these libraries in phishing websites are older (nearly 21.2 months) and distinct in comparison. For example, `Socket.IO` is uniquely used in phishing websites to send victims' information to an external server in real time. Furthermore, we find that a considerable portion of them still maintain a basic and simplistic structure (e.g., simply displaying a login form or image), while phishing websites have significantly evolved to bypass anti-phishing measures. Finally, through HTML structure and style similarities, we can identify specific target webpages of legitimate brands that phishing attackers reference and use to mimic for their phishing attacks.

CCS CONCEPTS

• **Security and privacy → Web application security**.

KEYWORDS

Web Security, JavaScript Library, Phishing

WWW '24, May 13–17, 2024, Singapore, Singapore.

ACM ISBN 979-8-4007-0171-9/24/05…$15.00
https://doi.org/10.1145/3589334.3645535

ACM Reference Format:
Kyungchan Lim, Jaehwan Park, and Doowon Kim. 2024. Phishing Vs. Legit: Comparative Analysis of Client-Side Resources of Phishing and Target Brand Websites. In *Proceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589334.3645535

1 INTRODUCTION

Phishing attacks aim to lure benign users (*i.e.*, potential victims) into divulging sensitive personal information (*e.g.*, login credentials). To accomplish this, phishing attackers meticulously construct deceptive websites that closely mimic legitimate target brand websites. Accordingly, similar to typical modern websites, phishing websites employ various client-side techniques, such as client-side scripting (JavaScript), Cascading Style Sheets (CSS), and more, all aimed at creating an appearance that is highly convincing and closely mirrors the genuine target brand websites.

Phishing attacks have long been a dominant and widespread cybersecurity threat for many years [40], leading to many attempts to conduct a comprehensive understanding of the phishing ecosystem and present new effective defense (or detection) mechanisms using machine learning (or deep learning) [25, 27, 41, 63, 65, 78, 96]. Particularly, for tactics, prior work mainly focused on how new evasion techniques (*e.g.*, cloaking or domain squatting) were used in the wild [29, 56, 72, 75, 88, 91, 96]. As the defense mechanisms, new effective phishing detection techniques were presented using machine learning (or deep learning); these detection techniques relied on screenshots (*e.g.*, login forms and target brand logos) and URLs [25, 27, 41, 63–65]. Also, the effectiveness of the current phishing blocklists (*e.g.*, Google Safe Browsing) was well understood [74, 75].

Although there has been significant progress in understanding phishing attacks, the client-side resources used in phishing websites (*e.g.*, how they are used) remain understudied. By understanding client-side resources used in phishing attacks, we can gain insights into the construction and techniques of phishing websites. To this end, we raise the following research question: **Main RQ:** "*How do phishing websites employ client-side resources (especially JavaScript libraries), in comparison to their corresponding legitimate target brand websites?*" Specifically, we raise the follow-up research questions: **RQ1)** What kind of client-side resources are employed in phishing websites? **RQ2)** Which JavaScript libraries are widely prevalent in phishing websites in terms of popularity, version, uniqueness, and inclusive type, as compared to their legitimate counterparts? **RQ3)** Why do a smaller percentage of phishing websites use JavaScript, compared to the legitimate target ones? **RQ4)** How similar are phishing websites and their corresponding legitimate target brand websites in terms of HTML structures?

To answer the research questions, we systematically measure the client-side resources of phishing websites by comparing ones of their legitimate target brand websites to better understand the phishing ecosystem, with an emphasis on JavaScript libraries as it is the most prevalent resource in phishing websites. Specifically, as shown in Figure 1, we first design a web crawler using Chrome Selenium WebDriver [9] to collect client-side resources of phishing websites and take screenshots of phishing websites; the phishing URLs are fed by APWG eCX [33] – one of the largest phishing blocklist repository. This helps us successfully collect 7.1M phishing websites (1.1M distinct phishing domains) for 25 months (July 10th, 2021 to July 31st, 2023). After refining our collected dataset (*e.g.*, filtering out inaccessible websites through clustering screenshots), we select the top 100 target brand websites and collect their client-side resources of landing pages and login pages from the Internet archive's wayback machine service (archive.org). Then, we compare the client-side resources between phishing websites and their target brand websites, with a focus on the dominant libraries, their versions, HTML structure similarity, and unique libraries not typically found in legitimate websites.

Our study reveals that phishing websites generally employ more diverse JavaScript libraries than legitimate target websites do, but these libraries are often older (nearly 21.2 months) and distinct in comparison. Certain libraries, such as `Socket.IO`, are rarely found in legitimate websites but serve specific purposes in the context of phishing attacks. Moreover, 22.8% of our collected phishing websites are still basic and rudimentary without JavaScript libraries (*i.e.*, they simply contain a single login form, an image, etc.), even though phishing websites have been advanced to defeat (or evade) anti-phishing mechanisms according to prior studies [57, 70]. Finally, our assessment involves gauging the similarities between phishing websites and their legitimate counterparts by comparing both the HTML structure (*i.e.*, structural similarity) and CSS classes (*i.e.*, style similarity). This analysis helps us to identify the authentic webpages that phishing attackers mimic for their phishing attacks. Our contributions are summarized as follows:

- We conduct a longitudinal, comparative analysis of client-side resources of phishing websites and their corresponding legitimate target brand websites collected for 25 months (July 10, 2021 to July 31, 2023).
- We reveal that phishing websites use a greater variety of JavaScript libraries than legitimate target brand websites, but the older versions are used for phishing websites. Moreover, certain libraries (*e.g.*, Socket.IO) are used only for phishing websites.
- We also find that a considerable number of phishing websites still maintain a basic and simplistic structure (*e.g.*, simply displaying a login form or image).
- We are able to identify specific target webpages of legitimate brands used to mimic phishing attacks using HTML structure similarity and style similarity.
- We discuss potential recommendations against phishing attacks, and we publicly share our source code and the collected two-year client-side resources (and screenshots) of phishing websites to facilitate future research in the community.

2 BACKGROUND

2.1 Phishing Attack

A phishing attack is a type of social engineering attack in which malicious actors build deceptive websites meticulously crafted to mimic legitimate websites, with the primary goal of enticing benign users (*i.e.*, potential victims) to divulge their personal information (*e.g.*, credentials). These pernicious social engineering tactics have affected billions of Internet users [49, 95].

2.2 JavaScript Library

Modern websites (even phishing websites) employ JavaScript libraries [37, 58, 62, 81, 86] that are embedded in HTML documents to interact with the Document Object Model (DOM) and support dynamic and interactive features in web pages (*e.g.*, interactive maps and dynamically updating content). All modern web browsers are built with JavaScript engines; *e.g.*, Google Chrome uses V8 [42].

Inclusion Option. To include a JavaScript library, web developers use a '`<script>`' tag and specify the URL of the library in the '`src`' attribute. The URL may point to either (1) a local JavaScript library file or (2) an external JavaScript library file. The *first* option is that they copy JavaScript libraries to their own web servers. This provides more control over the libraries than externally hosted libraries for web developers. In this option, the libraries are loaded from the same domain; for example, '<script src="./example.js"></script>'.

On the other hand, the *second* option is to load externally-hosted JavaScript libraries; for example, '<script src="https://example.com/example.js"></script>'. Using externally hosted libraries is a convenient option, as the burden of hosting and maintenance can be avoided. In this option, content delivery networks (CDN) are widely used to efficiently deliver externally hosted JavaScript libraries to clients. As CDNs ensure that edge servers are geographically dispersed to be closer to clients, the clients will be delivered contents (*e.g.*, libraries) from the nearest edge server, which can significantly reduce the delivery delay.

Library Versioning. JavaScript library projects commonly adopt Semantic Versioning [93], where a version number comprises three components: MAJOR.MINOR.PATCH (*e.g.*, 3.7.1). MINOR versions increase when new features are added and PATCH versions do when bugs are addressed; both do not change the public APIs. MAJOR versions, on the other hand, are for significant changes to libraries (*e.g.*, modifications to the public API that could lead to compatibility issues). The version information is typically found in the library's URL or its file name (*e.g.*, 'https://example.com/jquery-**3.7.1**.js').

3 MOTIVATION

While there has been notable advancement in comprehending phishing attacks, there is still limited knowledge about the client-side resources employed in phishing websites and how they are utilized. Understanding the client-side resources utilization in phishing attacks can help us (1) gain insights into the construction and techniques of phishing websites, and (2) suggest potential recommendations or mitigation against phishing attacks. To this end, we raise a main research question: *How do phishing websites employ client-side resources (especially JavaScript libraries), in comparison to their corresponding legitimate target brand websites?* In this study, we address the research question by (1) collecting the client-side resources of phishing and legitimate target brand websites for nearly

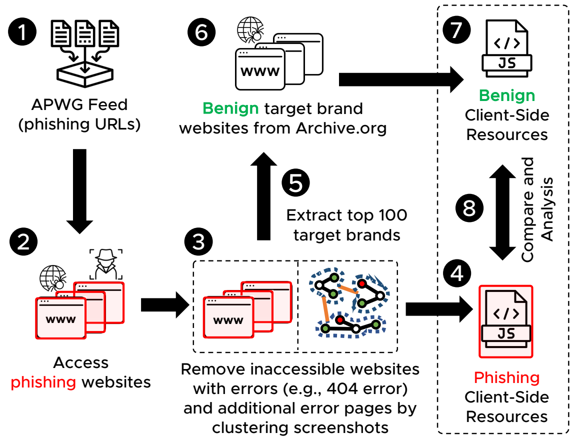

Figure 1: Overview of Data Collection. 1) Collect phishing URLs from APWG eCX. 2) Access each phishing URL. 3) Remove inaccessible websites with errors and by clustering screenshots. 4) Collect phishing client-side resources. 5), 6), and 7) Extract the top 100 target brands and collect client-side resources of these benign websites, and 8) Compare and analyze the benign and phishing resources.

two years and (2) conducting a comparative analysis of the resources of phishing and legitimate websites.

Research Focus on Legitimate Target Brands. We mainly focus on the top 100 target brands of our phishing websites. As described, the top 100 target brands account for 90.5% of our collected phishing websites. Moreover, as phishing websites typically mimic the login pages or landing pages (*i.e.*, index files) and obtain victims' credentials, we mainly focus on the landing pages and login pages of target brands.

4 DATASET COLLECTION

Our aim is to gain a deeper understanding of the phishing ecosystem, with a focus on client-side resources (*e.g.*, JavaScript libraries) by comparing them to the legitimate websites of the target brands. In this section, We describe our newly designed web crawler that collects the client-side resources (as well as screenshots) of phishing websites and their corresponding legitimate target brand websites.

4.1 Phishing Client-side Resource Collection

Phishing Website Crawler Design. We design a web crawler that collects the client-side web resources of phishing websites; the client-side resources are HTML pages, JavaScript libraries (*i.e.*, embedded JavaScript code snippets, internally-hosted JavaScript library files, and externally-hosted JavaScript library files), CSS files, and images. Moreover, the crawler captures screenshots of phishing websites after fully loading and executing client-side resources (*e.g.*, JavaScript libraries). The screenshots are used to serve the purpose of verifying the authenticity of reported phishing URLs and identifying any potential access errors (*e.g.*, internal DB connection errors).

We utilize APWG eCrime Exchange (`eCX`) [33] to obtain reliable phishing URLs because `eCX` is one of the most trusted and widely used repositories for phishing URLs used for real phishing attacks in the wild[47, 74–77, 90, 96, 97]. Our crawler is periodically (every 10 minutes) fed the most recently reported phishing URLs from APWG `eCX` and proceeds to visit these phishing URLs to collect client-side web resources and take screenshots of the phishing websites. The crawler is implemented with Google Selenium ChromeDriver [14] because ChromeDriver can help simulate real users' interactions with phishing websites since it fully loads and executes all client-side resources, such as JavaScript, CSS, and images on the webpages. Also, ChromeDriver could help circumvent certain phishing evasion techniques that scrutinize whether genuine web browsers actually access the phishing websites [67].

Type	# of URLs (Domains)
APWG Phishing URLs	15,747,193 (1,545,253)
Accessed URLs	7,067,778 (1,135,264)
Screenshots	6,125,810 (939,103)
Refined Dataset	3,388,997 (757,421)
# of Clusters	519,210
Collection Period	July '21 – July '23 (25 month)

Table 1: Overview of Our Collected Dataset.

Collecting and Refining Our Dataset. Our crawler runs every 10 minutes from July 10th, 2021 to July 31st, 2023 (for 25 months) and is fed a total of 15,747,193 (15.7M) phishing URLs from APWG `eCX`. As described in Table 1, out of 15.7M phishing URLs, it successfully accesses only 7,067,778 URLs (44.9%); in other words, the rest (8,679,415 URLs, 55.1%), are inaccessible as the web servers are unreachable due to offline web servers, DNS errors, etc. Even after successfully accessing each phishing URL and its web server, our crawler occasionally experiences a number of access errors due to web server internal errors (*e.g.*, 404 errors or internal DB connection errors) or blocking (or evasion) techniques (*e.g.*, CAPTCHAs). As these errors may introduce bias into our analysis of the collected dataset (for example, the CAPTCHAs pages may have different HTML code with different JavaScript libraries than the original phishing attack pages), we thoroughly filter out these error pages from our collected dataset using a clustering technique.

Clustering Screenshots. Recall that our crawler also takes screenshots of phishing websites using Selenium ChromeDriver. As these screenshots can be used to identify such errors to remove and phishing target brands, we cluster our collected screenshots by utilizing `Fastdup` [31], an unsupervised open-source tool for image dataset analysis. This tool is widely used for finding duplicates, outliers, and clusters of related images in a corpus of images, and works well on high contamination rates datasets [46]. Specifically, we have a total number of 6,125,810 image screenshots of phishing websites to cluster.[1] We run the tool with the all screenshots, and then we have 519,210 clusters (if the cluster only has 1 image, also called a cluster), 94.2% of screenshots are clustered. The max, min, mean, and median cluster sizes are 1,404,569, 1, 14.97, and 1 screenshots, respectively. As shown in Figure 2, 6,152 clusters (1.2% out of 519,210) account for 70% and 48,809 clusters (9.4% out of 519,210) account for 90% of our collected screenshots.

Our Final Refined Dataset. We manually take a look at each cluster and mainly focused on the clusters with more than 1000 phishing domains for validation. This accounted for more than 90% of our datasets and conservatively filters out all phishing websites

[1]Note that out of a total of 7,067,778 phishing URLs, only 6,125,810 (86.7%) phishing URLs have been successfully taken screenshots.

if a cluster has screenshots of errors or evasions (*e.g.*, CAPTCHA). 3,388,997 (47.9% out of 7.1M accessible phishing URLs) phishing webpages remain after removing all clusters that have error pages. Due to the nature of phishing websites [87], a number of removed pages take up 52.1% of our crawled initial phishing URLs. Finally, we obtain 757,421 distinct domains that are used for the analysis in this study. Note that our focus is on phishing domains, not individual phishing URLs. This is because of the nature of phishing campaigns, which usually operate under a single phishing domain with multiple URLs. This would be facilitated by the dynamically-generated URL feature that helps evade the anti-phishing techniques.

4.2 Target Brand's Resource Collection

Identifying Legitimate Target Brands. Recall that our main goal is to compare and analyze phishing client-side resources with those of legitimate target brands. We first identify the legitimate target brands of our collected phishing websites by leveraging both APWG `eCX` brand information (specified as metadata along with phishing URLs) and our clustering approach. As phishing websites typically resemble login or main pages, their appearance can look very similar unless they contain unique appearance features (*e.g.*, unpopular target brand or unique error pages). This allows us to identify a total of 4,606 target brands. Of these, for our in-depth analysis, we mainly focus on the top 100 target brands as they account for 90.5% of the phishing attacks in our dataset. We believe that this extensive coverage would provide a comprehensive perspective of the phishing ecosystem.

Collecting Client-side Resources of Target Brand Websites. We leverage the Internet Archive's Wayback Machine [2] to collect the client-side resources of the legitimate target brand websites. This archive service provides the archived versions of webpages dating back to 1996 and client-side resources (*e.g.*, HTML files, JavaScript files, CSS, and any image files included in the webpages). This service is widely used in prior work to better understand the web ecosystem [28, 30, 51, 60, 71, 85]. From the archive service, we attempt to gather the main webpages (*i.e.*, index pages) and login webpages (if separately available) of the top 100 target brands that have been collected by the archive service during our phishing dataset collection period (July 10, 2021 – July 31st, 2023). This is because phishing websites often mimic and display main or login webpages to deceive victims into divulging their credentials.

As shown in Table 5, we collect the main webpages from all 100 target brands, and separate login pages from only 80 brands as the remaining 20 brands have login forms on their main pages. During our phishing dataset collection period (751 days), a total of 108,343 webpages (67,482 main pages and 40,861 login pages) of the top 100 target brands are successfully collected. Note that *not all* brands (especially lower-ranked brands) are collected on a daily basis. In other words, the websites are archived at varying frequencies by the service. However, in our dataset, as the target brands are typically top-ranked in the wild, they are archived, on average, approximately once every 1.24 days, which is nearly once a day.

4.3 Identifying Resources and Versions

To identify client-side resources and their versions from both collected phishing and target brand client-side resources (HTML files, JavaScript library files, etc.), we utilize a website profiling tool, called `Wappalyzer` [23]. This profiling tool has been considered reliable and widely used in prior work to identify client-side resources and their versions on webpages [32, 37, 38, 48, 68, 80, 89]. Specifically, the tool employs regular expressions to extract the various types of client-side resources, including JavaScript libraries, CSS, and Content Management Systems (*e.g.*, WordPress), along with their respective versions from HTML and JavaScript files. Moreover, to verify the results of `Wappalyzer`, we also run our own Python script to identify resources and versions using regular expression.

5 OVERVIEW OF CLIENT-SIDE RESOURCE

Our study involves the quantitative assessment of client-side resources found on phishing websites. In this section, we aim to provide a general overview of various types of client-side resources employed in phishing attacks. Our first step involves quantifying the number and types of client-side resources utilized in phishing websites. We find that 95.3% of phishing websites (721,822, out of 757,421) use at least one client-side resource. Specifically, 626,719 (82.7%, out of 757,421) phishing websites contain one or more embedded internal JavaScript codes in their HTML or URLs of external JavaScript files. Interestingly, in contrast to previous studies of measuring client-side resources in benign websites [58], a smaller percentage of phishing websites utilize JavaScript libraries; in the benign websites, 97% Alexa's top sites contain JavaScript. This observation motivates us to raise a research question; "*Why do the smaller percentage of phishing websites use JavaScript, compared to the legitimate target brand websites?*" We seek to answer the research question in Section 6.2. Meanwhile, CSS is the second most frequently utilized resource at 72.3% (547,660), followed by Favicon (35.0%, 265,182) and SVG (Scalable Vector Graphics) at 16.5% (124,734). CMS (Content Management System) accounts for 7.3% (55,135), while XML collectively amounts to 1.5%.

Our Research Focus on JavaScript Usage. In this study, our main focus is on the two prominent client-side resources: JavaScript libraries and CSS that play a critical role in the appearance of phishing websites. This focus is driven by the important role of appearance in phishing attacks, as phishing attackers typically spend most of their time on the visuals of phishing websites to mimic the legitimate target brand websites and lure victims.

6 JAVASCRIPT LIBRARY IN PHISHING

Consistent with previous studies [58, 86] indicating JavaScript as the most utilized client-side resource in benign websites, it also stands out as the prevalent client-side resource in phishing websites, being employed in 82.7% (626,719 of 757,421) of phishing websites. Particularly, out of 626,719 websites, 585,073 (93.4%) utilize at least one JavaScript library, whereas only 6.4% solely include embedded their own JavaScript code. Moreover, `jQuery` and `Bootstrap` are the dominant libraries in phishing attacks. Finally, we observe that unique JavaScript libraries (*e.g.*, `Clipboard.js` or `Socket.IO`) are found in our phishing dataset. These unique libraries are barely used in the wild according to our result (as shown in Table 2) and prior work [58, 86].

6.1 JavaScript Library Usage

In this section, we examine the prevalent JavaScript libraries in phishing websites, with a focus on the dominant libraries, their

Phishing Website						Legitimate Target Brand Website (Landing & Login Page)					
		Inclusion Type			Dominant			Inclusion Type			Dominant
Library	Usage (%)[1]	Int.[2]	Ext.[2]	CDN[2,3]	Version[4]	Library	Usage (%)[1]	Int.[2]	Ext.[2]	CDN[2,3]	Version[4]
jQuery [17]	436,832 (57.7%)	33.7%	66.3%	91.5%	v3.5.1 (26.9%)	jQuery [17]	52 (52%)	75.0%	25.0%	33.3%	v3.5.1 (29.4%)
Bootstrap [13]	236,056 (31.2%)	32.5%	67.5%	89.7%	v4.0.0 (40.1%)	Bootstrap [13]	26 (26%)	54.5%	45.5%	20%	v5.0.0 (76.8%)
Clipboard.js [15]	105,206 (13.9%)	0.9%	99.1%	0.3%	v1.5.15(43.1%)	core-js [98]	16 (16%)	0%	100%	-	v2.6.12 (72.3%)
core-js [98]	47,060 (6.2%)	4.9%	95.1%	98.8%	v3.0.0 (21.4%)	React [19]	15 (15%)	50.0%	50%	-	v17 (37.0%)
Vue.js [24]	39,496 (5.2%)	13.4%	86.6%	15.4%	v3.3.4 (30.2%)	Choices [36]	14 (14%)	100%	0%	-	N/A[5]
Modernizr [18]	29,317 (3.9%)	65.3%	34.7%	37.0%	v2.8.3 (78.2%)	Boomerang [8]	10 (10%)	100%	0%	-	N/A[5]
jQuery-UI [11]	21,204 (2.8%)	29.9%	70.1%	87.8%	v1.10.3 (25.5%)	jQuery-UI [11]	10 (10%)	75.0%	25.0%	-	v1.12.1 (45.1%)
React [19]	18,670 (2.5%)	2.2%	97.8%	86.2%	v16.14.0 (51.5%)	Modernizr [18]	10 (10%)	90.0%	10.0%	-	v2.6.2 (82.1%)
Slick [5]	14,616 (1.9%)	87.7%	12.3%	39.9%	v1.6.0 (33.3%)	Emotion [39]	8 (8%)	0%	100%	33.3%	v11.9.0 (75.6%)
Lodash [4]	11,163 (1.5%)	5.8%	94.2%	94.1%	v4.17.21 (38.9%)	jQuery Migrate [54]	7 (7%)	42.9%	57.1%	-	v3.3.2 (73.2%)
jQuery Migrate [54]	10,536 (1.4%)	17.2%	82.8%	7.2%	v3.3.2 (37.7%)	Lodash [4]	7 (7%)	66.7%	33.3%	-	v1.13.1 (91.3%)
Moment.js [12]	9,971 (1.3%)	19.0%	81.0%	90.7%	v2.24.0 (45.6%)	RequireJS [6]	5 (5%)	66.7%	33.3%	-	v2.2.0 (100%)
RequireJS [6]	8,814 (1.2%)	39.1%	60.9%	3.7%	v2.2.0 (60.7%)	Slick [5]	5 (5%)	50.0%	50.0%	-	v1.8.1 (20.0%)
Choices [36]	8,601 (1.1%)	44.8%	55.2%	0%	v9.0.1 (20.5%)	styled-comp. [21]	4 (4%)	0%	100%	-	v5.3.0 (35.9%)
Angular [10]	8,130 (1.1%)	73.8%	26.2%	94.1%	v1.6.4 (45.3%)	Underscore.js [92]	4 (4%)	66.7%	33.3%	-	v1.13.4 (100%)
web-vitals [43]	6,446 (0.9%)	1.8%	98.2%	98.9%	v2.1.0 (50.0%)	Polyfill [52]	3 (3%)	11.1%	88.9%	25.0%	v3 (100%)
Axios [34]	6,442 (0.9%)	20.9%	79.1%	95.5%	v0.19.0 (62.1%)	Clipboard.js [15]	3 (3%)	75.0%	25.0%	-	v1.0.0 (75.0%)
OWL Carousel [16]	6,276 (0.8%)	80.4%	19.6%	13.1%	v1.0.0 (37.4%)	Angular [10]	3 (3%)	100%	0%	-	v7.2.15 (27.6%)
Socket.io [20]	4,755 (0.6%)	7.1%	92.9%	99.2%	v2.1.0 (31.5%)	Vue.js [24]	3 (3%)	0%	100%	100%	v2.6.11 (84.6%)
Lightbox [66]	4,719 (0.6%)	44.2%	55.8%	3.7%	v1.0.0 (22.0%)	Backbone.js [53]	2 (2%)	100%	0%	-	v1.2.3 (100%)
styled-comp. [21]	3,405 (0.4%)	25.0%	75.0%	100%	v5.3.5 (23.6%)	GSAP [45]	2 (2%)	100%	0%	-	v2.0.2 (100%)
Select2 [7]	2,537 (0.3%)	78.8%	21.2%	38.6%	v4.0.3 (35.2%)	OWL Carousel [16]	2 (2%)	100%	0%	-	N/A[5]
SweetAlert2 [22]	2,357 (0.3%)	50.8%	49.2%	9.2%	v7.26.11 (61.2%)	Prototype [79]	2 (2%)	0%	100%	-	N/A[7]
Polyfill [52]	2,226 (0.3%)	6.6%	93.4%	49.5%	v3 (75.1%)	LazySizes [26]	2 (2%)	100%	0%	-	N/A[5]
Emotion [39]	2,025 (0.3%)	62.8%	37.2%	0%	v11.9.0 (24.0%)	Lightbox [66]	2 (2%)	100%	0%	-	v2.2.3 (50.0%)
LazySizes [26]	1,998 (0.3%)	45.6%	54.4%	48.8%	v2.9.5 (28.3%)	web-vitals [43]	2 (2%)	100%	0%	-	N/A[5]
Hammer.js [3]	1,771 (0.2%)	93.1%	6.9%	75.0%	v2.0.4 (50.4%)	Datatables [84]	1 (1%)	100%	0%	-	N/A[5]
FancyBox [1]	1,659 (0.2%)	52.6%	47.4%	68.9%	v2.1.5 (52.0%)	FancyBox [1]	1 (1%)	100%	0%	-	v3.0.0 (100%)
Boomerang [8]	1,645 (0.2%)	1.5%	98.5%	49.9%	v1.0.0 (34.5%)	Moment.js [12]	1 (1%)	0%	100%	-	N/A[5]
Total	757,421 (100%)	39.9%[7]	60.1%[7]	47.4%[7]		**Total**	100 (100%)	69.1%[7]	30.9%[7]	6.8%[7]	

1: Usage per domain. 2: Int.: Internally-hosted libraries (*i.e.*, local JavaScript library file) and Ext.: externally-hosted libraries (*i.e.*, external JavaScript link).
3: Out of externally-hosted JavaScript libraries. 4: Most dominated version. 5: Not able to determine version due to JavaScript being embedded within HTML code.
6: Not able to determine version due to version number not included when using an external library. 7: Average number of usage.
Orange-colored libraries are more used in phishing websites than the legitimate ones. **Cyan**-colored libraries are only used in phishing websites.

Table 2: Top 29 JavaScript Usage, Inclusive Type and Dominant Version of Phishing Websites and Target Brand Websites.

versions, and unique libraries not typically found as high-ranked ones in benign websites.

Popular JavaScript Library. A total of 132 distinct JavaScript libraries are identified in our phishing dataset, in contrast to the 41 distinct JavaScript libraries found in their corresponding legitimate target brand websites. This implies that phishing attackers might incorporate a greater variety of JavaScript libraries than those actually used by legitimate brands on their websites. Particularly, phishing attackers utilize certain libraries (*e.g.*, `Socket.IO` and `Clipboard.js`) for their malicious purposes; these certain libraries are barely used in the legitimate ones, or more used in phishing websites than the legitimate ones. Further analysis of these libraries will be conducted later in this section.

Out of the 132 distinct libraries, our result shows that `jQuery` (57.7%) and `Bootstrap` (30.7%) are most used in both phishing websites, similar to other JavaScript usage statistics of benign websites [58, 86]. This proportion is smaller than the `jQuery` usage (83.9%) reported in prior work [58], despite the fact that half of the phishing websites (57.7%) in our phishing dataset utilize `jQuery`. `Bootstrap` is the second most used library in both phishing (31.2%) and legitimate ones (26%). Interestingly, `Clipboard.js` ranks third in popularity among phishing websites, while it is only ranked 16th among legitimate ones (we will further analyze it later this section).

More Used Library in Phishing. We further analyze the libraries that are more used in phishing websites than their legitimate target websites. As shown in Table 2 (colored in orange), three unique JavaScript libraries (among the top 29) are more utilized in phishing websites than their legitimate target websites during the same observation period; `Cipboard.js`, `Select2`, and `SweetAlert2`. These libraries are used by 13.9%, 0.3%, and 0.3% of phishing websites, respectively. Particularly, `Clipboard.js` [15] is an open-source JavaScript library that simplifies the process of copying text to the clipboard (*i.e.*, copy-to-clipboard functionality) in websites, which can enhance the user experience (*i.e.*, improving usability) by enabling users to copy content with simply one click. In our phishing dataset, we observe that the phishing websites leverage the library to facilitate the straightforward copying of attackers' cryptocurrency wallet addresses, such as Bitcoin, as illustrated in Listing 1. Out of a total of our collected phishing websites using this library, 38.4% employ the library for copying Bitcoin addresses. For example, a phishing website impersonates a major cryptocurrency exchange platform (or Tesla), enticing potential victims with promises of double earnings.

Uniquely Used Library in Phishing. We also find that three libraries are used only in phishing websites: `Axios`(0.9%), `Socket.IO` (0.6%), and `Hammer.js` (0.2%), as shown in Table 2 (colored in cyan).

In other words, these libraries are not used in their target brand websites. Specifically, `Axios`(0.9%) is to fetch data from APIs by making HTTP requests (*e.g.*, GET requests). For example, a phishing website makes a GET request to a certain URL and receives a response from the URL. We manually analyze the phishing websites using this library and find that the library is used to exfiltrate victims' IDs (or email addresses) and passwords to a certain server, as illustrated in Listing 2. Finally, this library is also used to communicate with their self-hosted CAPTCHA JavaScript library as an evasion technique, in order to check if visitors are real humans, rather than relying on the Google CAPTCHA service, as shown in Listing 3. This implies that the phishing attackers want to avoid disclosing their information (*e.g.*, the hosting server's IP information) to Google.

`Socket.IO` [20] is for real-time and event-based communication between users (such as web browsers) and web servers. This library is typically used when real-time data exchange is required (*e.g.*, real-time chat applications). In our collected phishing attacks, the library is used to promptly transmit visitors' information (*i.e.*, potentially victims) to their external server in real-time when they visit the phishing website, as illustrated in Listing 4. To elaborate, the phishing website initially obtains a visitor's identification from the URL because this phishing attempt is specifically targeted and its URL is sent to a particular individual along with a victim's identification as a BASE64-encoded parameter. Then the phishing website decodes this parameter and promptly sends the identification to the external server in real-time. For example, in our dataset, the phishing URL is 'https://[redacted]/?q=aWQ9c2MwbV9sYW5nPWVzX3NjPTc3NV91c2VyPTYyMzc0NjE3NDY%3D.' The BASE64-encoded parameter is decoded into 'id=sc0m_lang=es_sc=775_user=62374617467.' The targeted user is '62374617467,' and the user ID is sent to the external server immediately after the victim visits the phishing website. This enables the phishing attackers to assess the success rate of their phishing attacks (*e.g.*, who visits, who is lured, etc.)

> **Takeaway:** The JavaScript libraries utilized in phishing websites often mirror those used in their corresponding target brand websites. However, three distinct libraries (`Axios`, `Socket.IO`, and `Hammer.js`) are exclusively employed in phishing websites. Additionally, three other libraries (`Clipboard.js`, `Select2`, and `SweetAlert2`) are more frequently utilized in phishing websites compared to their legitimate counterparts. These libraries serve specific purposes in the context of phishing attacks.

Dominant Version. Next, we measure the prevalent versions of each JavaScript library in phishing websites. The most dominant version of `jQuery` in phishing websites is v3.5.1. This version was released on May 4, 2020, which is more than three years old. After this version, this library has seven more versions. Moreover, there is a similar trend with `Bootstrap`. The phishing websites with `Bootstrap` also use the outdated version, v4.0.0, released on January 19, 2018 (more than five years ago). Interestingly, compared to the legitimate target brand websites (v5.0.0, released on May 5, 2021), the phishing websites use an older version of the library. Likewise, in general, phishing websites tend to employ older versions of JavaScript libraries. Specifically, out of the top 29 JavaScript libraries with identified versions (as shown in Table 2), 47.1.% of the JavaScript libraries used in phishing websites are older than those employed in the legitimate target brand websites. On average, phishing websites employ JavaScript libraries that are 646 days older, equivalent to nearly 21.2 months, than the versions utilized by legitimate websites. This observation implies that phishing websites contain different versions of JavaScript libraries, compared to legitimate websites even though their primary goal is to imitate the legitimate target websites. Also, the phishing JavaScript libraries are even older, meaning that a reluctance among phishing sites to adopt (or update to) newer versions of libraries.

Inclusive Type. Recall that two inclusive types (internal and external) are used to include JavaScript libraries. Table 2 lists the percentage of the inclusion types of phishing and legitimate target brand websites. In the phishing websites, 60.1% have externally-hosted libraries while 39.9% utilize internal libraries. Interestingly, the legitimate target brand websites have a different usage pattern; 69.1% have internal libraries while only 30.9% use externally-hosted ones. This suggests that phishing websites tend to favor externally-hosted libraries, whereas legitimate target brand websites lean towards utilizing internal libraries. Moreover, out of externally-hosted libraries, 47.7% of the phishing websites rely on the Content Delivery Network (CDN) services for their external libraries. Specifically, the Google-hosted library service (ajax.googleapis.com) is the most commonly used in phishing websites. In other words, the remaining 52.6% of the phishing websites use resources taken from the target brand websites, which is discussed in Section 7.

> **Takeaway**: Despite the primary goal of phishing attacks being to mimic legitimate websites, these fraudulent sites often utilize different and outdated versions of JavaScript libraries, compared to their legitimate websites.

6.2 Phishing without JavaScript Library

There are 22.8% (172,348 out of 757,421) of our collected phishing webpages that do not use JavaScript. Of these 172,348 websites without JavaScript, 99.0% (170,650) of websites simply have only CSS, and the rest 1.0% (1,698) do not have both JavaScript and CSS. This observation prompts us to pose a follow-up research question: "Why do these phishing websites abstain from using JavaScript?" To answer the research question, we manually analyze the randomly selected samples from websites that do not use JavaScript to see how the websites are built without JavaScript. We find that these phishing websites lack sophistication in their design and often feature very simplistic structures, as basic as featuring a single login form accompanied by a target brand logo image. This highlights the surprising fact that a considerable number of phishing websites still remain rudimentary, even as recent studies [57, 70] reveal that recent phishing websites are built with the significant advancement in bypassing anti-phishing mechanisms. We believe that because building such basic phishing websites is comparatively inexpensive when contrasted with more advanced phishing websites, such basic phishing websites are used for phishing attacks in the wild.

> **Takeaway:** Even though phishing websites have been advanced to defeat (or evade) anti-phishing mechanisms, the considerable number of phishing websites still remain basic and rudimentary.

CDN	#	Blog	#	Program Lang.	#
Google APIs	155,062	Blogger	134,745	Python	135748
jQuery-CDN	120,389	WordPress	11,744	PHP	44129
Cloudflare	88,233	Wix	3,291	Node-js	10651
JSDelivr	31,522	Tiki CMS	21	Typescript	8038
UNPKG	10,110	Ghost	10	Java	4390
Total	498,505	**Total**	149,822	**Total**	205,009
CMS	**#**	**UI Framework**	**#**	**DB**	**#**
Weebly	30,560	Bootstrap	236,056	MySQL	42,529
WordPress	11,744	Animate-CSS	38,570	Firebase	5,409
Adobe Experience Mgr.	3,949	Marko	2,510	PostgreSQL	36
Wix	3,291	UIkit	2,177	Redis	19
GoDaddy Web Builder	1,482	Zurb-Foundation	1,839	Percona	10
Total	55,135	**Total**	285,981	**Total**	48,004

Table 3: Top 6 Web Applications Used in Phishing Websites.

C[1]	# of D.[2]	Sim.[3]	Target Brand	First Seen[4]	Mimicked-Date[5]	Diff.[6]
C1	47,714	97.7%	Facebook	2021-07-11	2020-08-12	333
C2	19,710	96.4%	Microsoft	2021-07-11	2018-01-03	1,285
C3	15,756	98.1%	Instagram	2022-10-20	2022-05-10	163
C4	14,614	85.9%	AT&T	2022-09-11	2022-09-10	1
C5	10,018	98.6%	WhatsApp	2022-02-11	2021-10-08	116
C6	9,637	88.0%	DHL	2023-03-09	2020-03-31	1,073
C7	9,567	65.7%	Ozon	2021-09-30	2021-03-27	187
C8	9,431	85.0%	Yahoo	2021-10-08	2017-01-01	1,741
C9	7,342	99.3%	Wells Fargo	2021-11-08	2019-04-23	930
C10	7,173	78.1%	Adobe	2023-02-12	2023-01-17	26

1: Cluster ordered by the number of domains. 2: The number of phishing domains.
3: Similarity. HTML structure and CSS class similarity within each cluster.
4: The first-seen date within each cluster.
5: The earliest date of a certain webpage that was mimicked. 6: The date difference.

Table 4: Top 10 Cluster by the Number of Phishing Domains with Similarity, Target Brand, First Seen, Mimicked-Date, and Date Difference.

7 HTML STRUCTURE SIMILARITY

In this section, we seek to answer our RQ4: "How similar are phishing websites and their corresponding legitimate target brand websites in terms of HTML structures?" This analysis helps us gain insights into the malicious tactics employed by phishing attackers when building their deceptive phishing websites. Specifically, we aim to determine whether phishing attackers resort to copying and pasting code directly from legitimate target brand websites to create their phishing sites. Moreover, this analysis helps us to identify the specific webpages of the target brand websites that are being mimicked. For example, we can know that a certain phishing website, commonly found in the wild, is mimicked from a webpage of the target brand dated Jan 10th, 2018.

Matching HTML Structure Similarity. We utilize a tool, called `html-similarity` [69] to assess the similarities in HTML structures between our collected HTML files from phishing websites and the archived HTML files from the corresponding legitimate target websites. This tool uses (1) sequence comparison of HTML tags (*i.e.*, structural similarity) and (2) CSS classes (*i.e.*, style similarity) to calculate the similarity between two given HTML files, which is presented in prior work [44]. We first run this tool with all collected HTML files within the top 10 clusters (see the clustering in Table 4.1) based on the number of distinct phishing domains for a more rigorous analysis, as shown in Table 4. Each cluster has on average 89.3% similarity among phishing websites.

Identifying Mimicked Legit Webpage. We raise a follow-up research question; "*What specific legitimate target brand webpages are used to mimic for phishing attacks?*" To address this question, we first identify the target webpages using the Internet archive service (archive.org). We collect all HTML files of the archive target brand websites beyond our data collection period. Then, we again utilize the `html-similarity` tool to compute the similarity score between our phishing webpage that first appears in each cluster and all HTML files of the corresponding target brand websites. For example, in Cluster 1, a phishing webpage targeting `Facebook` first appeared on July 11, 2021. This webpage is used to compare all archived HTML files of `Facebook` in terms of HTML structure similarity and style similarity and identify the highest similarity score. Finally, a legitimate webpage of `Facebook` on Aug. 12, 2020 (almost one year old), was identified to be used to mimic for phishing attacks. This analysis reveals that on average, 585.5 days older versions of target brand webpages are referenced (*i.e.*, mimicked) by phishing attackers. This implies that phishing attackers may use or reference older versions of target brand websites when building their phishing websites.

Attacker's Behavior of Building Phishing Website. We identify three approaches attackers adopt when constructing phishing websites: (1) Exact Replication, where they clone both HTML structures and resources of target websites; (2) Selective Replication, where resources from the target are copied but are integrated into different HTML structures; and (3) Original Construction, where a phishing website uses entirely different resources, but looks similar to target websites. Regardless of the methods, the core objective remains: the phishing website must convincingly resemble the target websites for victims.

While the first method is identifiable through techniques like HTML structure similarity, our focus narrows on the latter two. In the Selective Replication approach, instances arise where resources, even from the target brand's CDN, are incorporated into a unique web layout, as seen with the 'idmsa-gsx2-new-apple.com' phishing website where sources from Apple's CDN yet diverge in design. Interestingly, Figure 3 combines resources from DHL and USPS target websites as shown in Listing 5. In the Original Construction method, attackers craft sites with entirely distinct resources that, to the untrained eye, mirror the target's appearance, a tactic evident in the 'datastreamfusion.com/Arlene/Harrington/index.html' shown in Figure 4, the website's close resemblance to its target despite its distinct resource use.

Our analysis of the similarities between different clusters revealed that there is an exact overlap of 2.5% in terms of code copying. Additionally, more than 21.5% of the clusters show an overlap exceeding 85%. This data suggests that 2.5% of the clusters are exactly duplicating the same code, while more than 21.5% of the clusters are creating versions based on the target brand's website, with only slight modifications. These adjustments ensure the phishing sites maintain a similar HTML style and structure to the original, legitimate sites.

8 OTHER CLIENT-SIDE RESOURCES

Cascading Style Sheets (CSS). CSS is prevalent, accounting for 72.3% of the examined domains' primary client-side resource utilization. CSS can be integrated directly within the HTML as embedded code or referenced externally, analogous to how JavaScript is implemented. When CSS embeds within the HTML, it offers the flexibility to shape the webpage's format in a myriad of ways. Consequently, the embedded approach to CSS is predominant: among the domains that do not employ JavaScript, 35.8% opt for embedded CSS exclusively, eschewing external JavaScript libraries. Additionally, image formats like PNG, JPG, and GIF are seen in widespread use with 91.2% of the total.

Favicon. A favicon is a small graphic or icon file representing a website, commonly displayed in browser tabs or used to identify websites in bookmark lists. Favicons appear in 35.0% of phishing sites. Due to its simplicity and public accessibility, the favicon primarily serves as a placeholder for browser tabs. However, from our observations, phishing websites often repurpose the favicon, replacing it with a logo image.

SVG. We find that SVG is used in 16.5% of domains; SVG is a vector image format file. Because it is in XML-based format, SVG file can contain HTML code which means that it can contain JavaScript. This means malicious code can also be in an SVG file in HTML and JavaScript format. Our analysis shows that in our collected dataset that uses SVG, 3.1% of domains include HTML code in SVG files for malicious use.

9 DISCUSSION

Suggested Mitigation. Our research contributes significantly to the understanding of phishing websites by unveiling key patterns and characteristics of client-side resources.

- *Library-Based Detection.* Browsers could proactively detect websites using JavaScript libraries uniquely found in phishing sites (such as `Axios`, `Socket.IO`, and `Hammer.js`) and alert users before they access these sites. This strategy requires careful examination of the libraries to minimize false positives.
- *HTML Code Analysis.* Browsers could analyze HTML codes for unusual patterns, such as overly simplistic designs. Since 97% of legitimate websites use dynamic JavaScript content, the absence of such content could be a red flag. However, this method also requires detailed monitoring to avoid false positives, especially in cases where JavaScript might not be used for legitimate reasons.
- *Comparison with Target Brand Websites.* Browsers can compare the code of a visited website with that of known brand websites. If the website's code closely resembles that of a target brand, especially if the URL seems suspicious (e.g., domain squatting), it can indicate a phishing attempt. This approach is more effective when the URL contains the target brand's name, allowing for easier comparison with a whitelist of popular domain names. A crucial aspect of this method is the comparison should not only be with the current HTML code of these popular domains but also with their historical versions since phishing websites replicate older versions of legitimate sites.

10 RELATED WORK

There has been no previous work on analyzing phishing websites' resources. By analyzing phishing website resources, we can find insights on how phishing websites are created and even further, how phishing websites are compared to benign target websites.

Phishing Ecosystem. The research in the field of phishing attacks has yielded a well-rounded understanding of this malicious ecosystem [25, 27, 29, 50, 55, 56, 63–65, 72, 74–77, 82, 88, 96]. It encompasses two significant areas: attack tactics and defenses against the attacks. *First*, in the phishing tactics, prior work attempted to better understand how phishing attackers circumvent currently existing phishing detection or defense mechanisms and lure more victims into their phishing campaigns. Particularly, it has been well understood how squatting techniques have been employed by attackers [29, 56, 72, 88, 91]. Moreover, Oest et al. and Zhang et al. measured evasion techniques (*e.g.*, cloaking) used in the wild [75, 77, 96]. *Second*, in the defense mechanisms against phishing attacks, previous studies presented new effective detection algorithms using machine learning techniques (or deep learning) [25, 27, 41, 63–65]. Also, Oest et al. also measured the effectiveness of the current phishing blocklists (*e.g.*, Google Safe Browsing) [74, 75]. Nonetheless, little has been studied on how client-side resources (*e.g.*, JavaScript libraries) are used in phishing attacks. Particularly, our study takes a novel approach by gathering both phishing websites and benign websites, addressing an overlooked aspect related to client-side resources in phishing websites.

Client-side Resource Measurement. Several measurement studies have aimed to gain a deeper understanding of the web ecosystem, with a particular focus on security practices of client-side resources used in typical (benign) websites, usually using Alexa 1M domains or Tranco 1M domains [35, 58, 59, 61, 71, 73, 83, 86, 94]. Particularly, prior work has also predominantly centered on JavaScript libraries of benign websites, given their prominent role as client-side resources [58, 71]. For instance, Demir et al. conducted a longitudinal study that examined updating behaviors, such as JavaScript library updates, and discovered that these libraries, even when vulnerable, were rarely updated [37]. These measurement studies provide a general overview of general trends of typical benign websites in JavaScript library usage, updates, vulnerabilities of outdated versions, and library inclusion types. However, our research delves deeper into the comparison between phishing websites' individual JavaScript libraries and their versions.

11 CONCLUSION

We study the client-side resources used in phishing websites by comparing them with the resources in the corresponding legitimate target brand websites. We discover that phishing sites often use a broader range of JavaScript libraries than legitimate sites, although these libraries are typically older by about 21.2 months. Despite advancements in phishing techniques, a large proportion of these sites still retain basic designs, like plain login forms. Our analysis also pinpoints the specific pages of legitimate brands that attackers frequently mimic in their phishing campaigns, identified through HTML and stylistic similarities.

ACKNOWLEDGMENTS

We thank the anonymous referees for their constructive feedback. We also thank APWG for sharing their valuable phishing dataset. The authors gratefully acknowledge the support of NSF (2210137 and 2335798). This research was also supported by Science Alliance's StART program and gifts from Google exploreCSR and TensorFlow. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] 2010. Fancybox - Fancy jQuery lightbox alternative. http://fancybox.net/. (Accessed on 09/27/2023).

[2] 2014. Internet Archive: Digital Library of Free & Borrowable Books, Movies, Music & Wayback Machine. https://archive.org/. (Accessed on 09/26/2023).

[3] 2016. Hammer.JS Hammer.js. https://hammerjs.github.io/. (Accessed on 09/27/2023).

[4] 2016. Lodash. https://lodash.com/. (Accessed on 09/27/2023).

[5] 2017. slick - the last carousel you'll ever need. https://kenwheeler.github.io/slick/. (Accessed on 09/27/2023).

[6] 2018. RequireJS. https://requirejs.org/. (Accessed on 09/27/2023).

[7] 2021. Getting Started: Select2 - The jQuery replacement for select boxes. https://select2.org/. (Accessed on 09/27/2023).

[8] 2021. GitHub Akamai Boomerang: End user oriented web performance testing and beaconing. https://github.com/akamai/boomerang. (Accessed on 09/27/2023).

[9] 2021. *Selenium.* https://www.selenium.dev/

[10] 2022. AngularJS Superheroic JavaScript MVW Framework. https://angularjs.org/. (Accessed on 09/27/2023).

[11] 2022. jQuery UI. https://jqueryui.com/. (Accessed on 09/27/2023).

[12] 2022. Moment.js | Home. https://momentjs.com/. (Accessed on 09/27/2023).

[13] 2023. Bootstrap · The most popular HTML, CSS, and JS library in the world. https://getbootstrap.com/. (Accessed on 09/27/2023).

[14] 2023. ChromeDriver - WebDriver for Chrome - Getting started. https://chromedriver.chromium.org/getting-started. (Accessed on 09/21/2023).

[15] 2023. clipboard.js — Copy to clipboard without Flash. https://clipboardjs.com/. (Accessed on 09/27/2023).

[16] 2023. Home | Owl Carousel | 2.3.4. https://owlcarousel2.github.io/OwlCarousel2/. (Accessed on 09/27/2023).

[17] 2023. jQuery. https://jquery.com/. (Accessed on 09/27/2023).

[18] 2023. Modernizr: the feature detection library for HTML5/CSS3. https://modernizr.com/. (Accessed on 09/27/2023).

[19] 2023. React – A JavaScript library for building user interfaces. https://legacy.reactjs.org/. (Accessed on 09/27/2023).

[20] 2023. Socket.IO. https://socket.io/. (Accessed on 09/27/2023).

[21] 2023. styled-components. https://styled-components.com/. (Accessed on 09/27/2023).

[22] 2023. SweetAlert2 - a beautiful, responsive, customizable and accessible (WAI-ARIA) replacement for JavaScript's popup boxes. https://sweetalert2.github.io/. (Accessed on 09/27/2023).

[23] 2023. Technologies - Wappalyzer. https://www.wappalyzer.com/technologies/. (Accessed on 09/18/2023).

[24] 2023. Vue.js - The Progressive JavaScript Framework | Vue.js. https://vuejs.org/. (Accessed on 09/27/2023).

[25] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. 2020. VisualPhishNet: Zero-day phishing website detection by visual similarity. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security.* 1681–1698.

[26] aFarkas. 2021. GitHub - aFarkas/lazysizes: High performance and SEO friendly lazy loader for images (responsive and normal), iframes and more, that detects any visibility changes triggered through user interaction, CSS or JavaScript without configuration. https://github.com/aFarkas/lazysizes. (Accessed on 09/27/2023).

[27] Sadia Afroz and Rachel Greenstadt. 2011. PhishZoo: Detecting Phishing Websites by Looking at Them. In *2011 IEEE Fifth International Conference on Semantic Computing.* 368–375. https://doi.org/10.1109/ICSC.2011.52

[28] Vibhor Agarwal and Nishanth Sastry. 2022. "Way back then": A Data-driven View of 25+ years of Web Evolution. In *Proceedings of the ACM Web Conference 2022.* 3471–3479.

[29] Pieter Agten, Wouter Joosen, Frank Piessens, and Nick Nikiforakis. 2015. Seven months' worth of mistakes: A longitudinal study of typosquatting abuse. In *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society.

[30] Mshabab Alrizah, Sencun Zhu, Xinyu Xing, and Gang Wang. 2019. Errors, Misunderstandings, and Attacks: Analyzing the Crowdsourcing Process of Ad-Blocking Systems. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 230–244. https://doi.org/10.1145/3355369.3355588

[31] Amir Alush, Dickson Neoh, and Danny Bickson et al. 2023. Fastdup. GitHub.Note: https://github.com/visuallayer/fastdup.

[32] Danny E. Alvarez, Daniel B. Correa, and Fernando I. Arango. 2016. An analysis of XSS, CSRF and SQL injection in colombian software and web site development. In *2016 8th Euro American Conference on Telematics and Information Systems (EATIS).* 1–5. https://doi.org/10.1109/EATIS.2016.7520140

[33] APWG. 2023. The APWG eCrime Exchange (eCX). https://apwg.org/ecx/. (Accessed on 09/19/2023).

[34] axios. 2023. Getting Started | Axios Docs. https://axios-http.com/docs/intro. (Accessed on 09/27/2023).

[35] Adam Barth, Collin Jackson, and John C Mitchell. 2008. Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security.* 75–88.

[36] Choices-js. 2022. GitHub Choices-js/Choices: A vanilla JS customisable select box/text input plugin. https://github.com/Choices-js/Choices. (Accessed on 09/27/2023).

[37] Nurullah Demir, Tobias Urban, Kevin Wittek, and Norbert Pohlmann. 2021. Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security. In *Passive and Active Measurement*. Springer International Publishing, Cham, 76–92.

[38] Carlos Duarte, Inês Matos, João Vicente, Ana Salvado, Carlos M. Duarte, and Luís Carriço. 2016. Development Technologies Impact in Web Accessibility. In *Proceedings of the 13th International Web for All Conference* (Montreal, Canada) *(W4A '16)*. Association for Computing Machinery, New York, NY, USA, Article 6, 4 pages. https://doi.org/10.1145/2899475.2899498

[39] emotion js. 2023. GitHub emotion-js/emotion: CSS-in-JS library designed for high performance style composition. https://github.com/emotion-js/emotion. (Accessed on 09/27/2023).

[40] FBI. 2023. Internet Crime Complaint Center Releases 2022 Statistics — FBI. https://www.fbi.gov/contact-us/field-offices/springfield/news/internet-crime-complaint-center-releases-2022-statistics. (Accessed on 09/13/2023).

[41] Anthony Y. Fu, Liu Wenyin, and Xiaotie Deng. 2006. Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD). *IEEE Transactions on Dependable and Secure Computing* 3, 4 (2006), 301–311. https://doi.org/10.1109/TDSC.2006.50

[42] Google. 2023. v8/v8 - Git at Google. https://chromium.googlesource.com/v8/v8. (Accessed on 09/26/2023).

[43] GoogleChrome. 2023. GitHub GoogleChrome/web-vitals: Essential metrics for a healthy site. https://github.com/GoogleChrome/web-vitals. (Accessed on 09/27/2023).

[44] Thamme Gowda and Chris A Mattmann. 2016. Clustering web pages based on structure and style similarity (application paper). In *2016 IEEE 17th International conference on information reuse and integration (IRI)*. IEEE, 175–180.

[45] GreenSock. 2023. GSAP - GreenSock. https://greensock.com/gsap/. (Accessed on 10/10/2023).

[46] Fabian Gröger, Simone Lionetti, Philippe Gottfrois, Alvaro Gonzalez-Jimenez, Ludovic Amruthalingam, Labelling Consortium, Matthew Groh, Alexander A. Navarini, and Marc Pouly. 2023. SelfClean: A Self-Supervised Data Cleaning Strategy. arXiv:2305.17048 [cs.CV]

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

[48] Hao He, Lulu Chen, and Wenpu Guo. 2017/03. Research on Web Application Vulnerability Scanning System based on Fingerprint Feature. In *Proceedings of the 2017 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2017)*. Atlantis Press, 150–155. https://doi.org/10.2991/mecae-17.2017.27

[49] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M Voelker, and David Wagner. 2019. Detecting and characterizing lateral phishing at scale. In *28th USENIX Security Symposium (USENIX Security 19)*. 1273–1290.

[50] Hang Hu, Steve TK Jan, Yang Wang, and Gang Wang. 2021. Assessing Browser-level Defense against {IDN-based} Phishing. In *30th USENIX Security Symposium (USENIX Security 21)*. 3739–3756.

[51] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. 2017. The Ad Wars: Retrospective Measurement and Analysis of Anti-Adblock Filter Lists. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 171–183. https://doi.org/10.1145/3131365.3131387

[52] JakeChampion. 2023. Polyfill.io. https://polyfill.io/v3/. (Accessed on 09/27/2023).

[53] jashkenas. 2023. Backbone.js. https://backbonejs.org/. (Accessed on 10/10/2023).

[54] jquery. 2023. GitHub jquery-migrate: A development tool to help migrate away from APIs and features that have been or will be removed from jQuery core. https://github.com/jquery/jquery-migrate. (Accessed on 09/27/2023).

[55] Doowon Kim, Haehyun Cho, Yonghwi Kwon, Adam Doupé, Sooel Son, Gail-Joon Ahn, and Tudor Dumitras. 2021. Security Analysis on Practices of Certificate Authorities in the HTTPS Phishing Ecosystem. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security.* 407–420.

[56] Panagiotis Kintis, Najmeh Miramirkhani, Charles Lever, Yizheng Chen, Rosa Romero-Gómez, Nikolaos Pitropakis, Nick Nikiforakis, and Manos Antonakakis. 2017. Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. ACM, New York, NY, USA, 569–586.

[57] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. 2021. Catching Transparent Phish: Analyzing and Detecting MITM Phishing Toolkits. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security.* 36–50.

[58] Tobias Lauinger, Abdelberi Chaabane, Sajjad Arshad, William Robertson, Christo Wilson, and Engin Kirda. 2018. Thou shalt not depend on me: Analysing the use of outdated javascript libraries on the web. *arXiv preprint arXiv:1811.00918* (2018).

[59] Sebastian Lekies, Ben Stock, and Martin Johns. 2013. 25 million flows later: large-scale detection of DOM-based XSS. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 1193–1204.

[60] Ada Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 2016. Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016. In *25th USENIX Security Symposium (USENIX Security 16)*.

[61] Ada Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 2016. Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/lerner

[62] Kyungchan Lim, Yonghwi Kwon, and Doowon Kim. 2023. A Longitudinal Study of Vulnerable Client-side Resources and Web Developers' Updating Behaviors. In *Proceedings of the 2023 ACM on Internet Measurement Conference*. 162–180.

[63] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages.. In *USENIX Security Symposium*. 3793–3810.

[64] Ruofan Liu, Yun Lin, Xianglin Yang, Siang Hwee Ng, Dinil Mon Divakaran, and Jin Song Dong. 2022. Inferring Phishing Intention via Webpage Appearance and Dynamics: A Deep Vision Based Approach. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1633–1650. https://www.usenix.org/conference/usenixsecurity22/presentation/liu-ruofan

[65] Wenyin Liu, Xiaotie Deng, Guanglin Huang, and A.Y. Fu. 2006. An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing* 10, 2 (2006), 58–65. https://doi.org/10.1109/MIC.2006.23

[66] lokesh. 2023. GitHub lokesh lightbox2: THE original Lightbox script (v2). https://github.com/lokesh/lightbox2. (Accessed on 09/27/2023).

[67] Sourena Maroofi, Maciej Korczyński, and Andrzej Duda. 2020. Are you human? resilience of phishing detection to evasion techniques based on human verification. In *Proceedings of the ACM Internet Measurement Conference*. 78–86.

[68] Fabian Marquardt and Lennart Buhl. 2021. Déjà Vu? Client-Side Fingerprinting and Version Detection of Web Application Software. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. 81–89. https://doi.org/10.1109/LCN52139.2021.9524885

[69] matiskay. 2023. matiskay/html-similarity: Compare html similarity using structural and style metrics. https://github.com/matiskay/html-similarity. (Accessed on 10/11/2023).

[70] Elizabeth Montalbano. 2022. EvilProxy Commodifies Reverse-Proxy Tactic for Phishing, Bypassing 2FA. https://www.darkreading.com/vulnerabilities-threats/evilproxy-commodifies-reverse-proxy-tactic-phishing-bypassing-2fa. (Accessed on 10/11/2023).

[71] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2012. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 736–747.

[72] Nick Nikiforakis, Steven Van Acker, Wannes Meert, Lieven Desmet, Frank Piessens, and Wouter Joosen. 2013. Bitsquatting: Exploiting Bit-flips for Fun, or Profit?. In *Proceedings of the 22Nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13)*. ACM, New York, NY, USA, 989–998.

[73] Frolin Ocariza, Kartik Bajaj, Karthik Pattabiraman, and Ali Mesbah. 2013. An Empirical Study of Client-Side JavaScript Bugs. In *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. 55–64. https://doi.org/10.1109/ESEM.2013.18

[74] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. 2019. Phishfarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1344–1361.

[75] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. 2020. {PhishTime}: Continuous Longitudinal Measurement of the Effectiveness of Anti-phishing Blacklists. In *29th USENIX Security Symposium (USENIX Security 20)*. 379–396.

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

[77] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. 2020. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *29th USENIX Security Symposium (USENIX Security 20)*.

[78] Hyunjun Park, Kyungchan Lim, Doowon Kim, Donghyun Yu, and Hyungjoon Koo. 2023. Demystifying the Regional Phishing Landscape in South Korea. *IEEE Access* 11 (2023), 130131–130143.

[79] prototype. 2023. Prototype JavaScript framework: a foundation for ambitious web applications. http://prototypejs.org/. (Accessed on 10/10/2023).

[80] Nur Aini Rakhmawati, Sayekti Harits, Deny Hermansyah, and Muhammad Ariful Furqon. 2018. A Survey of Web Technologies Used in Indonesia Local Governments. *SISFO Vol 7 No 3* 7 (2018).

[81] Gregor Richards, Sylvain Lebresne, Brian Burg, and Jan Vitek. 2010. An analysis of the dynamic behavior of JavaScript programs. In *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*. 1–12.

[82] Richard Roberts, Yaelle Goldschlag, Rachel Walter, Taejoong Chung, Alan Mislove, and Dave Levin. 2019. You Are Who You Appear to Be: A Longitudinal Study of Domain Impersonation in TLS Certificates. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2489–2504.

[83] Prateek Saxena, Steve Hanna, Pongsin Poosankam, and Dawn Song. 2010. FLAX: Systematic Discovery of Client-side Validation Vulnerabilities in Rich Web Applications.. In *NDSS*.

[84] SpryMedia. 2023. DataTables | Table plug-in for jQuery. https://datatables.net/. (Accessed on 10/10/2023).

[85] Ben Stock, Martin Johns, Marius Steffens, and Michael Backes. 2017. How the Web Tangled Itself: Uncovering the History of {Client-Side} Web ({In) Security}. In *26th USENIX Security Symposium (USENIX Security 17)*. 971–987.

[86] Ben Stock, Martin Johns, Marius Steffens, and Michael Backes. 2017. How the Web Tangled Itself: Uncovering the History of Client-Side Web (In)Security. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 971–987. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/stock

[87] Karthika Subramani, William Melicher, Oleksii Starov, Phani Vadrevu, and Roberto Perdisci. 2022. PhishInPatterns: measuring elicited user interactions at scale on phishing websites. In *Proceedings of the 22nd ACM Internet Measurement Conference*. 589–604.

[88] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. 2014. The Long {"Taile"} of Typosquatting Domain Names. In *23rd USENIX Security Symposium (USENIX Security 14)*. 191–206.

[89] Yuta Takata, Hiroshi Kumagai, and Masaki Kamizono. 2021. The Uncontrolled Web: Measuring Security Governance on the Web. *IEICE Transactions on Information and Systems* 104, 11 (2021), 1828–1838.

[90] Bhaskar Tejaswi, Nayanamana Samarasinghe, Sajjad Pourali, Mohammad Mannan, and Amr Youssef. 2022. Leaky Kits: The Increased Risk of Data Exposure from Phishing Kits. In *2022 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 1–13.

[91] Ke Tian, Steve T. K. Jan, Hang Hu, Danfeng Yao, and Gang Wang. 2018. Needle in a Haystack: Tracking Down Elite Phishing Domains in the Wild. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18)*. New York, NY, USA, 429–442.

[92] Underscore. 2022. Underscore.js. https://underscorejs.org/. (Accessed on 05/26/2023).

[93] Semantic Versioning. 2023. Semantic Versioning 2.0.0. https://semver.org/. (Accessed on 09/26/2023).

[94] Philipp Vogt, Florian Nentwich, Nenad Jovanovic, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. 2007. Cross site scripting prevention with dynamic data tainting and static analysis.. In *NDSS*, Vol. 2007. 12.

[95] Suzanne Widup, Alex Pinto, David Hylender, Gabriel Bassett, and Philippe langlois. 2021. *2021 Verizon Data Breach Investigations Report*.

[96] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. 2021. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1109–1124.

[97] Penghui Zhang, Zhibo Sun, Sukwha Kyung, Hans Walter Behrens, Zion Leonahenahe Basque, Haehyun Cho, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, et al. 2022. I'm SPARTACUS, No, I'm SPARTACUS: Proactively Protecting Users from Phishing by Intentionally Triggering Cloaking Behavior. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 3165–3179.

[98] zloirock. 2023. GitHub zloirock core-js: Standard Library. https://github.com/zloirock/core-js. (Accessed on 09/27/2023).

A APPENDIX

A.1 In-Depth Analysis of Phishing Kits and Tools

Experiment Design. We further analyze the correlation between phishing kits, tools (*e.g.*, Gophish), and the JavaScript library usage with its versions. Specifically, we aim to better understand whether the phishing kits and tools influence the client-side resource usage on phishing websites. In this analysis, we utilize our own 4,707 phishing kits that we collected ourselves while collecting the client-side resources and screenshots for two years (Aug. 2022 – July

2023). Because multiple JavaScript libraries and versions can be found within a single phishing kit (*i.e.*, we do not know which library and version are used for phishing attacks), simply relying on extracting versions and names of JavaScript libraries from their file names may lead to bias. To address this issue, we deploy our collected phishing kits on our web server running Apache (on Ubuntu 18.04 and PHP version 7.2.24) and collect the phishing client-side resources by visiting the deployed phishing websites using Chrome Selenium WebDriver, which is the same approach as our client-side resource collection for the original experiments in this paper. After collecting these client-side resources, we employ the same analysis methodology used for the phishing website JavaScript library analysis, utilizing the Wappalyzer tool.

Phishing Kit Result. In this analysis, our findings reveal a notable similarity in the usage of JavaScript libraries between the phishing kits and the websites. Particularly, the 10 popular JavaScript libraries from our phishing websites are also in phishing kits. However, we observe a discrepancy in the versions of these libraries. Specifically, only two libraries (Modernizr [18] and Lightbox [66]) share identical versions across both phishing websites and phishing kits. Of the other eight libraries, 6 of them (jQuery [17], Core-js [98], jQuery-UI [11], Lodash [4], jQuery-Migrate [54], and Moment [12]) use older versions of libraries than those found on the phishing websites, and the other 2 libraries (Bootstrap [13] and Angular [10]) use newer versions of libraries than those found on the phishing websites. Specifically, for the most popular JavaScript library, jQuery, phishing kits provided version 2.2.4 with 34.7% while the phishing websites collected in the wild had version 3.5.1 with 26.9%.

Phishing Tool Result. For this analysis, we employ GoPhish, a well-known phishing tool, to create phishing websites targeting different brands (chase.com, yahoo.com, online.sbi, huntington.com, sbcgroup.com, usps.com, microsoft.com), which are among the most popular in our dataset. Specifically, we copy and paste the HTML code of 8 legitimate target brand websites to GoPhish, and GoPhish generates phishing websites mimicking the target brand websites. Then, we conduct a manual examination of the JavaScript library usage and its versions used in these phishing websites. We observe that no distinctions in the JavaScript libraries and versions between the websites created with GoPhish and the authentic target brand websites. This observation suggests that GoPhish effectively incorporates the same JavaScript libraries and versions as the target brand websites.

A.2 Limitation

In our collected phishing dataset, we identify a total of 4,606 target brands. We mainly focus on the top 100 target brands for a more in-depth analysis of more impactful phishing attacks. While we acknowledge that this approach may not encompass all phishing attacks, it is noteworthy that the top 100 target brands represent 90.5% of the phishing attacks within our dataset. This concentration allows for a more in-depth analysis, providing valuable insights into the phishing ecosystem.

Data Collection Challenges. From our initial list of URLs from APWG, we are able to collect only half of the initial list. We hypothesize two scenarios for this: one possibility is that these URLs were incorrectly reported; the other is that our crawler may have been blocked by the attackers using a cloaking technique. Determining the specific cloaking methods employed would require additional investigation. However, as our primary objective is to analyze client-side resources rather than delve into cloaking techniques, we mainly focus our analysis on the client-side resources of the URLs we successfully collected. Specifically, with the collected 7 million URLs, manually reviewing all 7 million entries is impractical, we establish specific criteria for URL removal. We cautiously filter out error pages to minimize the occurrence of false positives as much as possible. Although this approach significantly reduces the total number of URLs in our dataset, it enables us to achieve more accurate results with the remaining dataset.

Target Brand Website Archival Challenges. Moreover, in spite of our best efforts to collect the legitimate target brand websites from the Internet archive service (archive.org), we encounter a challenge related to dataset collection — the websites have been archived with varying frequencies by the service. However, given that these target brand websites are relatively the top-ranked ones in the wild, they tend to have a relatively short archival frequency. In our dataset, they have been archived by the Internet archive service, on average, approximately once every 1.24 days, which is nearly once a day. Note that for our analysis, we collect all 100 target brands that have been archived almost daily.

Focus on Popular JavaScript Libraries. while we have a more extensive list of such libraries, our work currently focuses on the most popular ones. Interestingly the AES library (aes.js) which is well-known to be used in phishing campaigns ranks 66th in our list, being used across 161 phishing domains. Future research that conducts a comprehensive examination of these types of libraries promises to yield intriguing insights.

A.3 Temporal Analysis

To evaluate whether the libraries used in phishing campaigns are outdated from their inception, we employ temporal analysis, creating a graph to examine the distribution of the most commonly used version of the library throughout our data collection period. The analysis with a focus on the jQuery library reveals a consistent pattern of using older library versions over this timeframe. This trend implies that phishing attackers are consistently opting for older versions of libraries when they start new phishing campaigns. For instance, jQuery version 3.5.1, which was new at the start of our data collection in July 2021, was still being used at the end of our collection period in July 2023.

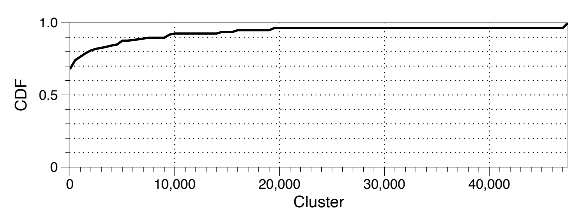

Figure 2: CDF of Clustered Screenshots.

```
<h1 class="f-24 mvn em-300">Transactions for address: <span id="
      trnsctin">0x1131b7355243aedd...</span></h1>
...
<button class="btn btn-warning mb-25" data-clipboard-target="#
      trnsctin" data-toggle="tooltip" data-placement="top" data-
      original-title="Copied!">
```

Rank	Brand	Page	#[1]	Rank	Brand	Page	#[1]	Rank	Brand	Page	#[1]	Rank	Brand	Page	#[1]	Rank	Brand	Page	#[1]
1	Facebook	M	751	21	IRS	M/L	751	41	Brooks Running	M/L	752	61	TD bank	M/L	693	81	Royal Bank of Canada	M/L	752
2	Microsoft	M/L	750	22	M&T bank	M/L	752	42	Commonwealth Bank	M/L	752	62	BT internet	M/L	752	82	Crocs	M/L	722
3	Instagram	M	727	23	Orange S.A.	M/L	752	43	Banco Itau	M	704	63	Rabobank	M/L	752	83	CaixaBank	M	752
4	AT&T	M/L	749	24	Santander	M/L	752	44	StarHub	M/L	752	64	Coinbase	M/L	752	84	BECU	M/L	752
5	WhatsApp	M	730	25	Swiss Post	M/L	751	45	Cox Communications	M/L	750	65	HSBC	M/L	752	85	Bank of Ireland	L	752
6	DHL	M/L	752	26	Bank BRI	M	738	46	Rakuten Card	L	752	66	Piraeus Bank	M	728	86	DocuSign	M/L	717
7	Ozon	M/L	705	27	eBay	M/L	752	47	Dr.Martens	M/L	752	67	Swisscom	M/L	751	87	American Express	M/L	748
8	Yahoo, Aol	M/L	692	28	Tesco	M/L	751	48	ICS - Intl. Card Services	M/L	751	68	Navy FCU	M/L	750	88	DenizBank	M/L	747
9	Wells Fargo	M	747	29	Sparkasse	M/L	752	49	Scotiabank	M/L	752	69	Deutsche Post	M/L	750	89	HDFC Bank	M/L	381
10	Adobe	M/L	748	30	Google	M/L	752	50	Wayfair	M/L	752	70	ACB	M/L	752	90	Square	M/L	704
11	Meta	M	740	31	Intesa Sanpaolo	M/L	752	51	Bank of America	M/L	752	71	DPD	M/L	752	91	Vietcombank	M/L	734
12	PayPal	M/L	751	32	Linkdin	M	735	52	Uniswap	M/L	743	72	Zimbra	M/L	750	92	LINE	M/L	752
13	USPS	M	751	33	Credit Agricole	M	743	53	Alaska USA FCU	M/L	752	73	Societe Generale	M/L	735	93	Roundcube	M/L	751
14	Apple	M/L	749	34	BT Group	M/L	752	54	BBVA	M/L	752	74	Paxful	M/L	463	94	Desjardins	M/L	752
15	Netflix	M/L	736	35	La Poste	M	731	55	T-Mobile	M/L	749	75	1&1	M/L	752	95	Regions	M/L	748
16	Amazon	M/L	687	36	Shopify	M/L	752	56	Citibank	M	749	76	Microsoft Office 365*	M/L	399	96	Nedbank	M/L	752
17	Chase	M	752	37	Plesk	M	747	57	WeTransfer	M	752	77	CommBank of Australia	M/L	747	97	Banca Monte	M/L	752
18	Rakuten	M/L	752	38	Credit Agricole CIB	M	743	58	Societe General Group	M/L	735	78	Virgin	M/L	744	98	Absa bank	M/L	752
19	State Bank of India	M/L	752	39	SMBC	M/L	752	59	Huntington Bank	M/L	752	79	Türkiye Gov	M/L	752	99	Robinhood	M	106
20	NAVER	M/L	751	40	Ing Groep	M/L	752	60	NAB	M/L	752	80	Dropbox	M/L	748	100	Interac	M	298

1: The number of webpages we have collected during our observation period. ∗: This page looks difference from the rank #2 Microsoft.
M indicates the main pages (*i.e.*, landing or index pages) are collected. L indicates the separate login pages are collected.

Table 5: Top 100 Target Brands of Our Collected Phishing Attacks. The main pages of all top 100 brands. The separate login pages of only 80 brands are collected as the rest (20 brands) have the login forms in their main pages.

```
<i class="fas fa-copy"></i>
        <span aria-hidden="true"></span>Click/tap here to copy the
              address!</button>
```

Listing 1: Example Code of Clipboard.JS Usage.

```
window.MAIL_URL = 'https://younteam.vip/link/mail.php';
window.FINAL_REDIRECT_URL = 'https://google.com';
async function sendMail(email, password) {
    const data = new FormData();
    data.append('email', email);
    data.append('password', password);
    return await axios.post(window.MAIL_URL, data);
}
```

Listing 2: Example Code of Axios Library for Exfiltrating Victims' Information.

```
if (link_click_fraud_mode > 0) {
  console.log("Testing humanity")
  grecaptcha.ready(function() {
    grecaptcha.execute(site_key, {action: 'http_ok_redirect'}).
         then(function(token) {
      axios.post("/js/captcha/verify",
        {click_id: 31406009, token: token, link_id: 9068409}
      ).then(function(response) {
        console.log("Humanity score " + response.data.score)
        if (response.data.score < 0.5 && true){
          not_found();
        } else {
          if (!link_cloaking) {
            redirect();}}
      }).catch(function(error) {
        console.log("Unable to test humanity.")
        redirect();})})});
}
```

Listing 3: Example Code of Axios Library for CAPTCHA.

```
<script>
const socket = io("wss://sc0m.herokuapp.com");
const queryStrings = window.location.search;
const urlParamss = new URLSearchParams(queryStrings);
const qs = urlParamss.get('q');
let rrss = atob(qs);
let users = rrss.split("_")[3];
users = users.split("=")[1];
function ss(){
  socket.emit('add', {nickname: "users",groupe: parseInt(users)});
};
</script>
```

Listing 4: Example Code of Socket.IO Library for Retrieving Visitors' Identifications from URLs and Sending Them to Their External Servers in Real Time.

```
<title>DHL_Tracking</title>
<script src="https://tools.usps.com/go/scripts/libs/jquery.min.js"
       ></script>
<link rel="stylesheet" href="https://tools.usps.com//go/css/
       tracking-cross-sell.css">
<script type="text/javascript" charset="utf-8" src="https://www.
       usps.com/global-elements/lib/script/require-jquery.js"></
       script>
```

Listing 5: Example Code of Selective Replication approach. This phishing website is targeting the DHL website however using resources from USPS's CDN.

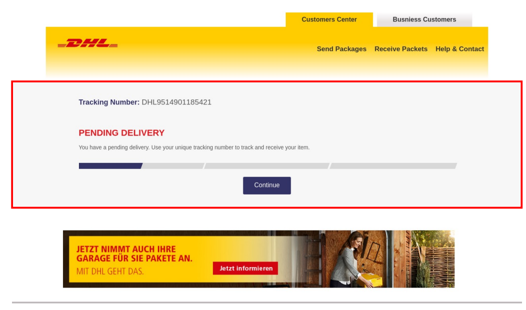

Figure 3: Example of Selective Replication of Using Mixed Resources and Target Brand. The red box has a different theme than other parts of the website due to USPS's resources.

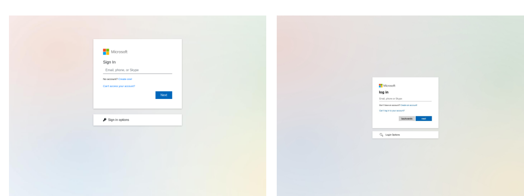

Figure 4: Example of Phishing Website Created with CSS. It contains a long list of CSS to make the website look legitimate. Whereas benign websites use the JavaScript library (either their own library or a 3rd-party library) to create a website.