To the Graduate Council:
I am submitting herewith a thesis written by Kyungchan Lim entitled "Understanding Social Engineering Attacks from Attacker Behaviors to Defensive Strategies." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
Degree, with a major in Computer Science.

Dr. Doowon Kim, Major Professor

We have read this thesis
and recommend its acceptance:

Dr. Doowon Kim

Dr. Jian Liu

Dr. Jinyuan Sun

Dr. Yonghwi Kwon

Accepted for the Council:

Amber Williams

Vice Provost and Dean of the Graduate School

To the Graduate Council:
I am submitting herewith a thesis written by Kyungchan Lim entitled "Understanding Social Engineering Attacks from Attacker Behaviors to Defensive Strategies." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
Degree, with a major in Computer Science.

Dr. Doowon Kim, Major Professor

We have read this thesis
and recommend its acceptance:

Dr. Doowon Kim

Dr. Jian Liu

Dr. Jinyuan Sun

Dr. Yonghwi Kwon

Accepted for the Council:

Amber Williams

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

# Understanding Social Engineering Attacks from Attacker Behaviors to Defensive Strategies

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Kyungchan Lim

August 2025

# Acknowledgements

I would like to express my deepest gratitude to my advisor and committee chair, Dr. Doowon Kim, for his unwavering support and mentorship throughout my Ph.D. journey. His guidance has been invaluable at every stage of my research, helping me refine my critical thinking, communication, and technical skills. His encouragement and insightful advice have shaped both my academic and professional growth, and I am incredibly grateful for his dedication and patience. This dissertation would not have been possible without his support.

I am also sincerely thankful to my committee members, Dr. Jinyuan "Stella" Sun, Dr. Jian Liu, and Dr. Yonghwi Kwon, for their thoughtful feedback and guidance, which have significantly enriched my research. Their insights have been instrumental in shaping the direction of my work.

I would like to extend my appreciation to my collaborators, whose expertise and contributions have been invaluable. I am also grateful to my lab mates, for their support, discussions, and collaboration. Their camaraderie made this journey both productive and enjoyable.

Finally, I am deeply thankful to my family for their unconditional love, encouragement, and support. Their belief in me has been my greatest source of strength, and I could not have completed this journey without them.

# Abstract

Phishing attacks continue to grow in prevalence and sophistication, creating fake websites that mimic legitimate services to steal credentials from victims. This "cat-and-mouse game" sees attackers continuously evolving with new evasion techniques, making quick detection the key challenge. This defense presents a comprehensive analysis of the phishing attack lifecycle across five domains: website creation, domain registration, regional attack variations, defensive mechanisms, and post-detection behaviors. Our research spans multiple years and millions of phishing URLs. Our findings reveal that phishing websites typically use outdated resources and lack security protections. Most phishing domains are maliciously registered rather than compromised, with attackers preferring cheaper TLDs and targeting specific brands. Global detection mechanisms often miss regionally targeted campaigns, particularly those in non-English contexts. Defensive measures show significant gaps, with widely used blocklists exhibiting substantial detection delays compared to specialized services. Website security practices remain inadequate, with virtually no sites following all recommended security guidelines. After detection, phishing websites typically remain operational for approximately two days, with those employing frequent visual changes persisting longer. This research contributes actionable insights for improving phishing defenses through earlier detection, vulnerability exploitation, and enhanced security practices.

# Table of Contents

# Chapter 1

# Introduction

Phishing attacks aim to lure unsuspecting users into divulging sensitive personal information, such as login credentials and financial information. To achieve this, attackers meticulously construct deceptive websites that closely mimic legitimate brand websites. By replicating the appearance and interactive elements of trusted platforms, phishing websites deceive victims into believing they are accessing genuine services. This manipulation makes phishing one of the most widespread and persistent cybersecurity threats, continuously evolving to evade detection mechanisms.

Similar to modern web applications, phishing websites employ various client-side techniques, such as JavaScript, Cascading Style Sheets (CSS), and other client-side resources, to convincingly mirror their target websites. However, unlike legitimate websites that prioritize security and user protection, phishing websites often exhibit distinctive characteristics in their JavaScript library usage, server configurations, and domain registrations. Attackers deploy phishing websites using pre-configured phishing kits, which simplify website creation by bundling cloned HTML templates, JavaScript scripts, and credential collection mechanisms. These kits significantly lower the technical barrier for attackers, enabling even inexperienced individuals to launch phishing campaigns at scale.

To effectively distribute phishing websites, attackers adopt different hosting strategies. Some leverage web content publishing services (e.g., Blogger, Wix) to host phishing content, while others deploy standalone web servers using technologies such as Apache [80] and Nginx [199]. Attackers may also register malicious domains that resemble legitimate websites, employing benign brand names in domains, squatted domains, and bulk registration to mislead victims. Alternatively, they may compromise already established, legitimate websites and inject phishing content to bypass domain-based detection mechanisms.

Phishing defense mechanisms primarily rely on blocklisting services, such as Google Safe Browsing (GSB) [66], APWG [100], OpenPhish [208], and PhishTank [216], to identify and block malicious URLs. These blocklists integrate into modern web browsers, warning users or preventing access to known phishing websites. However, blocklist-based defenses are inherently reactive, as they rely on the detection and reporting of phishing URLs after attacks have already been launched. The update latency of blocklists – the time gap between a phishing site's deployment and its inclusion in blocklists – creates a critical window of opportunity during which phishing websites remain active and accessible to victims [206]. Research indicates that up to 75% of phishing victims interact with a phishing website before it is blacklisted, highlighting the inadequacy of purely reactive defense mechanisms.

Given the limitations of blocklist-based approaches, understanding the fundamental construction and deployment strategies of phishing websites is crucial for developing more proactive phishing detection mechanisms. Phishing domains serve as an ideal intervention point since they act as the gateway between users and malicious content. Attackers generally follow two strategies in domain usage: (1) registering new domains specifically for phishing attacks, or (2) compromising legitimate websites and injecting phishing pages. While maliciously registered domains are intentionally created for phishing, compromised websites allow attackers to exploit existing reputations to evade detection.

Although prior research has made progress in understanding phishing techniques, gaps remain in comprehensively analyzing the behavior of phishing websites, the effectiveness of blocklist-based defenses, and the lifecycle of phishing domains [206]. Understanding phishing websites at a deeper level is essential to advancing detection mechanisms beyond reactive blocklisting. As phishing tactics evolve, attackers continually develop new evasion techniques, such as CAPTCHAs, cloaking, dynamic content manipulation, and fast-flux DNS techniques, to bypass detection systems. These detection systems often rely on static URL-based signatures, which, while effective to some extent, fail to capture more sophisticated phishing techniques. Prior studies [237, 183, 184, 184] have examined various aspects of phishing, such as visual similarities to legitimate websites, JavaScript and HTML usage, evasion techniques, and the role of phishing kits in automated attacks. However, a more holistic and large-scale measurement of phishing website behaviors, DNS abuse, and the persistence of phishing domains post-detection is needed.

**Research Objectives and Questions** To systematically investigate phishing attacks, we formulate the following research questions (RQs):

- RQ1: How do phishing attackers construct and deploy phishing websites? What distinctive characteristics do phishing websites exhibit in client-side resources, server configurations, and domain registration behaviors? How do phishing websites differ from legitimate websites in terms of JavaScript library usage and security policies?

- RQ2: How effective are current phishing defense mechanisms, and how do attackers adapt to them? How does GSB perform in detecting phishing websites compared to other blocklists (e.g., APWG, OpenPhish, PhishTank, PhishStats)? What is the detection delay of blocklists, and how does this affect phishing site persistence? How long do phishing websites remain operational after detection?

To answer these questions, this dissertation conducts a comprehensive, large-scale analysis of phishing websites, focusing on the following aspects:

**Understanding Attacker Behavior**

- Measuring client-side resource usage: We analyze JavaScript dependencies, third-party scripts, and security configurations to compare phishing websites with their legitimate counterparts.
- Analyzing server-side configurations and phishing kits: We investigate HTTP response headers, PHP configurations, and phishing kit usage, as these elements influence how phishing websites behave post-deployment.
- Examining phishing domain registration patterns: We study how attackers register and configure malicious domains, identifying trends in TLD usage, WHOIS records, and DNS configurations.

**Evaluating Phishing Defense Mechanisms**

- Measuring the effectiveness of blocklists: Since GSB is one of the most widely used blocklists, we analyze its detection accuracy and latency, comparing it with other blocklists such as APWG, OpenPhish, PhishTank, and PhishStats.
- Examining phishing website lifespan after detection: We measure how long phishing websites remain operational post-detection, identifying factors that contribute to delays in domain suspension.

By combining client-side analysis, server-side analysis, and DNS analysis, we aim to provide a holistic understanding of phishing attacker behaviors and the effectiveness of current anti-phishing mechanisms. Our findings will help bridge the gap between attacker strategies and detection limitations, leading to the development of more proactive phishing detection models.

**Contributions of This Dissertation** This dissertation makes the following key contributions:

- Comprehensive measurement of phishing websites, analyzing client-side resources, security policies, and domain registration trends.

- Longitudinal analysis of phishing domain registration and DNS abuse, identifying patterns in TLD usage and domain lifespan.
- Evaluation of blocklist effectiveness, measuring detection delays and comparing Google Safe Browsing with other blocklists.
- Post-detection phishing website behavior analysis, studying why some phishing websites persist longer than others after detection.

Phishing remains a major cybersecurity challenge, with attackers constantly refining their techniques to evade detection. While blocklist-based defenses provide some protection, their reactive nature leaves a critical vulnerability window, allowing phishing websites to remain active for hours or days before being blacklisted. By analyzing how phishing websites are constructed, how they differ from legitimate websites, and how attackers manipulate DNS infrastructure, this dissertation seeks to provide a data-driven foundation for improving phishing detection and mitigation strategies.

By addressing phishing website behaviors, defense mechanisms, and attacker strategies, this dissertation contributes to the development of more proactive and adaptive phishing detection techniques that can reduce user exposure to phishing threats.

# Chapter 2

# Background

Phishing is one of the most persistent and evolving cybersecurity threats, exploiting social engineering techniques to deceive users into revealing sensitive information, such as login credentials, financial data, or personal identification details. Over the years, phishing attacks have adapted to advancements in cybersecurity, developing new methods to bypass detection and increase their success rates. Understanding phishing attacks from multiple perspectives – including their construction, hosting, and defense mechanisms – is essential for developing effective mitigation strategies.

This section provides an overview of phishing attacks and current defense mechanisms, particularly focusing on blocklist-based detection, client-side resource usage, domain registration, and phishing kits.

## 2.1 Phishing Attack

Phishing attacks aim to lure users into interacting with fraudulent websites that closely mimic legitimate brands. These websites often copy the appearance, structure, and functionality of legitimate sites to deceive victims into submitting sensitive data, such as passwords, credit card details, or authentication tokens. Attackers typically distribute phishing URLs via email, SMS (smishing), social media, or malicious advertisements, tricking users into clicking links that lead to these fraudulent pages [183].

Phishing websites employ client-side scripting, HTML, CSS, and backend technologies (e.g., PHP, SQL) to replicate the targeted website's behavior. Unlike benign developers who prioritize security and usability, phishing attackers focus on rapid deployment and minimal maintenance, often relying on pre-packaged phishing kits to generate deceptive webpages [183, 181].

To increase their success rates, phishing attackers use various evasion techniques, including:

- Cloaking – Displaying benign content to security crawlers while showing phishing content to real users.
- CAPTCHAs – Preventing automated systems from scanning the phishing page.

Despite efforts to combat phishing, attackers continuously refine their techniques, making it difficult for traditional detection mechanisms to keep pace.

## 2.2 Phishing Defense Using Blocklisting

One of the primary defenses against phishing attacks is the use of blocklists, which store known phishing URLs and prevent users from accessing them. Google Safe Browsing (GSB) [139] is one of the most widely used blocklist-based defenses, integrated into browsers like Google Chrome [15], Mozilla Firefox [198], and Safari [26] to warn users before they visit suspected phishing sites. Other widely used blocklists include APWG [100], OpenPhish [69], PhishTank [72], and PhishStats [71], which collect phishing URLs from community reports and automated crawlers.

However, blocklist-based defenses have limitations, primarily due to detection latency—the time delay between a phishing website's deployment and its inclusion in blocklists. Furthermore, phishing websites often change domains rapidly and use fast-flux DNS techniques, making it difficult for blocklists to keep pace. These findings indicate that blocklist-based defenses alone are insufficient, emphasizing the need for proactive phishing detection mechanisms.

## 2.3 Client-Side Resources in Phishing Websites

JavaScript plays a crucial role in modern web development, providing dynamic content, interactive forms, and real-time updates. Phishing websites, like legitimate websites, also rely on JavaScript for their functionality. However, research has shown that phishing websites demonstrate distinct JavaScript usage patterns compared to legitimate websites [180].

In addition to outdated libraries, some phishing websites use JavaScript for real-time credential theft. Given these findings, JavaScript dependency analysis could serve as an early detection mechanism for identifying phishing websites, as attackers consistently reuse outdated or insecure JavaScript frameworks [180].

## 2.4 Domain Registration and DNS Manipulation in Phishing Attacks

Phishing attackers often register new domains that closely resemble legitimate ones to deceive users. These domains may exploit squatted domains, homoglyph attacks, or brand impersonation. Alternatively, attackers may compromise legitimate websites and inject phishing pages [182].

## 2.5 Phishing Kits

Phishing kits are pre-packaged toolkits that simplify the process of creating phishing websites by including HTML templates, JavaScript scripts, credential collection mechanisms, and email distribution tools. These kits are widely used by attackers with little technical expertise, enabling large-scale phishing campaigns [181].

# Chapter 3

# Related Work

Understanding phishing attack strategies and defense mechanisms has been a central focus of cybersecurity research. Several studies have explored blocklist effectiveness, client-side resource usage, server configurations, and phishing website lifespan [203]. While these studies provide valuable insights, significant gaps remain, particularly in the areas of phishing detection delays, security vulnerabilities in phishing servers, and long-term phishing domain persistence.

## 3.1 Measuring Phishing Defense: Blocklist Detection of Phishing Websites

Blocklists such as GSB [139], APWG [100], OpenPhish [69], and PhishTank [72] are widely used to protect users from phishing websites by preventing access to known malicious domains. Several studies have evaluated the effectiveness of blocklists in detecting phishing websites, revealing that detection mechanisms often suffer from delays and limited coverage. Research [176] has found that GSB exhibits significant detection delays, with some phishing websites remaining active for hours or even days before being blocklisted. Additionally, less than 19% of phishing URLs are detected across multiple blocklists, suggesting that no single blocklist provides

comprehensive protection. Another major challenge is that most phishing victims interact with phishing websites before detection occurs, demonstrating the reactive nature of blocklists. While prior studies have passively collected phishing datasets to assess detection efficacy, our work advances this approach by actively measuring phishing defense mechanisms through the controlled deployment of phishing websites. This methodology enables us to quantify detection latency in real-time, compare multiple blocklists, and assess evasion techniques such as CAPTCHA protection, redirection chains, and fast-flux DNS. By providing a real-world assessment of phishing detection gaps, we reveal how attackers exploit delays in blocklisting mechanisms to target victims before their websites are flagged.

## 3.2 Measuring Client-Side Resources in Phishing Websites

Phishing websites, like legitimate websites, employ various JavaScript libraries, third-party scripts, and client-side frameworks to enhance their appearance and functionality. Prior research has explored how phishing websites utilize client-side resources, focusing on JavaScript dependencies and dynamic content injection [183]. Findings from previous studies indicate that phishing websites often use outdated JavaScript libraries, with an average library version age of 21.2 months older than legitimate websites [183]. Attackers frequently copy JavaScript files from archived versions of target websites, leading to structural inconsistencies in phishing websites. Additionally, certain JavaScript libraries, such as Socket.IO, are frequently used in phishing websites to enable real-time credential exfiltration. Our work is the first large-scale study to systematically measure the prevalence of JavaScript libraries in phishing websites, compare JavaScript dependencies between phishing and benign sites, and analyze third-party script usage to reveal attacker trends. This research provides new insights

into how attackers reuse outdated JavaScript frameworks, which could serve as an early detection signal for phishing detection.

## 3.3 Web Server Configurations and Security Vulnerabilities in Phishing Websites

Most prior research on server-side security configurations has focused on legitimate websites, analyzing how they implement security best practices such as Content Security Policy (CSP), secure cookies, and HTTP security headers. Some studies [226] have measured web server misconfigurations in high-profile enterprise websites, but little research has explored phishing website server configurations. Existing studies indicate that legitimate websites frequently implement security headers such as CSP and HSTS to prevent client-side attacks, while many fail to properly configure HTTP response headers, making them vulnerable to clickjacking and XSS attacks. However, these studies have primarily focused on security vulnerabilities in benign websites rather than phishing infrastructure. Unlike previous work, our study is the first large-scale measurement of phishing website server configurations, particularly focusing on security misconfigurations and vulnerabilities. By analyzing HTTP response headers, CSP adoption, and phishing kit configurations, we uncover how phishing websites lack security best practices, use misconfigured servers, and deploy pre-packaged phishing kits that contribute to insecure setups. This research bridges the gap between phishing detection and web server security, offering new insights into how attackers configure phishing sites to evade detection while minimizing maintenance efforts.

## 3.4 Phishing Lifecycle and Website Longevity

Several studies have analyzed the lifespan of phishing websites, investigating how long phishing sites remain active before being detected and taken down. Prior research [108]

has examined phishing website longevity using closed datasets, finding that phishing sites typically last between 3 to 5 days before removal. Other studies [191, 126] have analyzed phishing takedown processes, identifying factors that contribute to delays in site shutdown. However, these studies have primarily relied on limited datasets that do not reflect real-world phishing behavior. Unlike previous studies that use closed datasets [206, 108], our research leverages real-world, actively collected phishing URLs to provide a more accurate measurement of phishing site longevity. Our study measures the lifespan of phishing websites post-detection, identifies the causes of phishing site takedown, and investigates attacker strategies for prolonging phishing website lifespans, including fast-flux DNS, redirection chains, and hosting provider rotation. By analyzing phishing sites from deployment to takedown, we provide a deeper understanding of phishing persistence, revealing why some phishing websites remain accessible long after detection.

# Chapter 4

# Attacker Behavior and Attack Creation

## 4.1 Client-side Resources in Phishing

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. The overview of data collection is shown in Figure 4.1

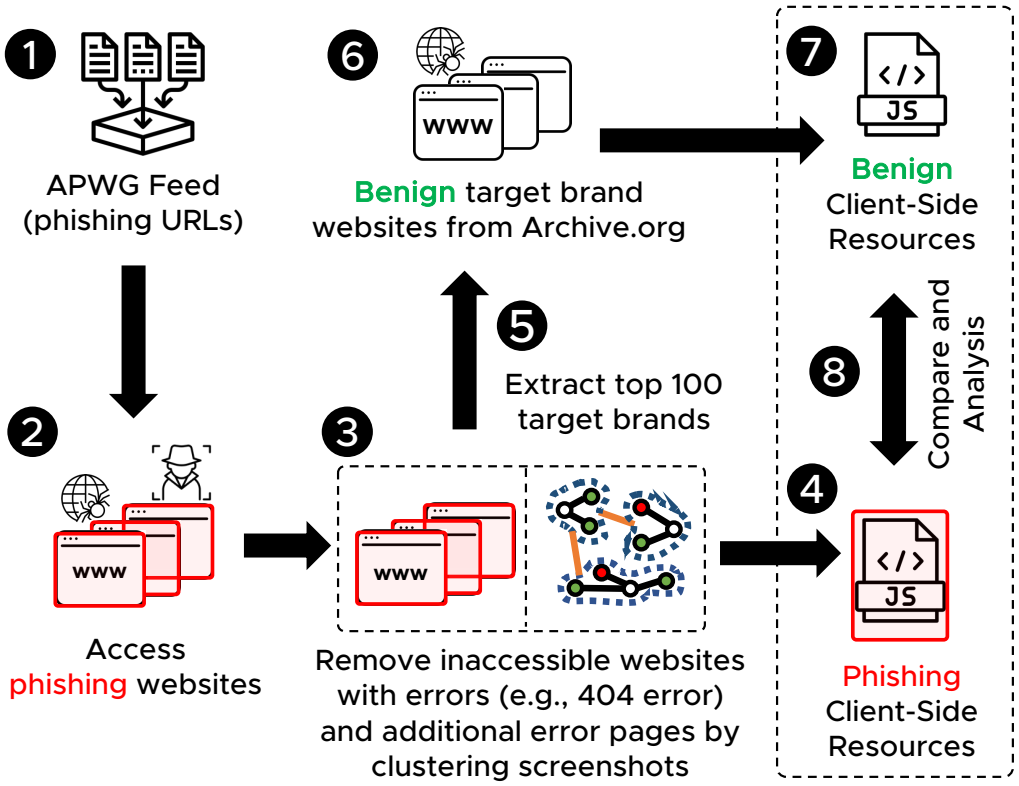



Figure 4.1: Overview of Data Collection. 1) Collect phishing URLs from APWG eCX. 2) Access each phishing URL. 3) Remove inaccessible websites with errors and cluster screenshots. 4) Collect phishing client-side resources. 5), 6), and 7) Extract the top 100 target brands and collect client-side resources of these benign websites, and 8) Compare and analyze the benign and phishing resources.

### 4.1.1 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.

#### 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`) [81] 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[147, 204, 202, 206, 203, 258, 239, 260]. 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 [57] 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 [187].

### 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 4.1, 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` [97], 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 [146]. Specifically, we have a total number of 6,125,810 image screenshots of phishing websites to cluster.* 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. 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.

*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.

Table 4.1: Overview of Our Collected Dataset.

| 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) |

**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 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 [236], 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.

### Target Brand's Resource Collection

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 [7] 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 [200, 157, 96, 178, 234, 92]. 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 4.2, 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.

Table 4.2: 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.

| 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.

### 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` [53]. This profiling tool has been considered reliable and widely used in prior work to identify client-side resources and their versions on webpages [123, 238, 189, 150, 223, 98, 127]. 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.

## 4.1.2 Client-side Resources in Phishing Websites

### Overview of Client-side Resources

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 [173], 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?*" . 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.

### JavaScript Library in Phishing

Consistent with previous studies [173, 235] 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 4.3) and prior work [173, 235].

Table 4.3: Top 29 JavaScript Usage, Inclusive Type and Dominant Version of Phishing Websites and Target Brand Websites.

| 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 [47] | 436,832 (57.7%) | 33.7% | 66.3% | 91.5% | v3.5.1 (26.9%) | jQuery [47] | 52 (52%) | 75.0% | 25.0% | 33.3% | v3.5.1 (29.4%) |
| Bootstrap [43] | 236,056 (31.2%) | 32.5% | 67.5% | 89.7% | v4.0.0 (40.1%) | Bootstrap [43] | 26 (26%) | 54.5% | 45.5% | 20% | v5.0.0 (76.8%) |
| Clipboard.js [45] | 105,206 (13.9%) | 0.9% | 99.1% | 0.3% | v1.5.15(43.1%) | core-js [261] | 16 (16%) | 0% | 100% | - | v2.6.12 (72.3%) |
| core-js [261] | 47,060 (6.2%) | 4.9% | 95.1% | 98.8% | v3.0.0 (21.4%) | React [49] | 15 (15%) | 50.0% | 50% | - | v17 (37.0%) |
| Vue.js [54] | 39,496 (5.2%) | 13.4% | 86.6% | 15.4% | v3.3.4 (30.2%) | Choices [115] | 14 (14%) | 100% | 0% | - | N/A[5] |
| Modernizr [48] | 29,317 (3.9%) | 65.3% | 34.7% | 37.0% | v2.8.3 (78.2%) | Boomerang [31] | 10 (10%) | 100% | 0% | - | N/A[5] |
| jQuery-UI [38] | 21,204 (2.8%) | 29.9% | 70.1% | 87.8% | v1.10.3 (25.5%) | jQuery-UI [38] | 10 (10%) | 75.0% | 25.0% | - | v1.12.1 (45.1%) |
| React [49] | 18,670 (2.5%) | 2.2% | 97.8% | 86.2% | v16.14.0 (51.5%) | Modernizr [48] | 10 (10%) | 90.0% | 10.0% | - | v2.6.2 (82.1%) |
| Slick [14] | 14,616 (1.9%) | 87.7% | 12.3% | 39.9% | v1.6.0 (33.3%) | Emotion [129] | 8 (8%) | 0% | 100% | 33.3% | v11.9.0 (75.6%) |
| Lodash [10] | 11,163 (1.5%) | 5.8% | 94.2% | 94.1% | v4.17.21 (38.9%) | jQuery Migrate [163] | 7 (7%) | 42.9% | 57.1% | - | v3.3.2 (73.2%) |
| jQuery Migrate [163] | 10,536 (1.4%) | 17.2% | 82.8% | 7.2% | v3.3.2 (37.7%) | Lodash [10] | 7 (7%) | 66.7% | 33.3% | - | v1.13.1 (91.3%) |
| Moment.js [40] | 9,971 (1.3%) | 19.0% | 81.0% | 90.7% | v2.24.0 (45.6%) | RequireJS [20] | 5 (5%) | 66.7% | 33.3% | - | v2.2.0 (100%) |
| RequireJS [20] | 8,814 (1.2%) | 39.1% | 60.9% | 3.7% | v2.2.0 (60.7%) | Slick [14] | 5 (5%) | 50.0% | 50.0% | - | v1.8.1 (20.0%) |
| Choices [115] | 8,601 (1.1%) | 44.8% | 55.2% | 0% | v9.0.1 (20.5%) | styled-comp. [51] | 4 (4%) | 0% | 100% | - | v5.3.0 (35.9%) |
| Angular [33] | 8,130 (1.1%) | 73.8% | 26.2% | 94.1% | v1.6.4 (45.3%) | Underscore.js [244] | 4 (4%) | 66.7% | 33.3% | - | v1.13.4 (100%) |
| web-vitals [142] | 6,446 (0.9%) | 1.8% | 98.2% | 98.9% | v2.1.0 (50.0%) | Polyfill [159] | 3 (3%) | 11.1% | 88.9% | 25.0% | v3 (100%) |
| Axios [107] | 6,442 (0.9%) | 20.9% | 79.1% | 95.5% | v0.19.0 (62.1%) | Clipboard.js [45] | 3 (3%) | 75.0% | 25.0% | - | v1.0.0 (75.0%) |
| OWL Carousel [46] | 6,276 (0.8%) | 80.4% | 19.6% | 13.1% | v1.0.0 (37.4%) | Angular [33] | 3 (3%) | 100% | 0% | - | v7.2.15 (27.6%) |
| Socket.io [50] | 4,755 (0.6%) | 7.1% | 92.9% | 99.2% | v2.1.0 (31.5%) | Vue.js [54] | 3 (3%) | 0% | 100% | 100% | v2.6.11 (84.6%) |
| Lightbox [185] | 4,719 (0.6%) | 44.2% | 55.8% | 3.7% | v1.0.0 (22.0%) | Backbone.js [160] | 2 (2%) | 100% | 0% | - | v1.2.3 (100%) |
| styled-comp. [51] | 3,405 (0.4%) | 25.0% | 75.0% | 100% | v5.3.5 (23.6%) | GSAP [145] | 2 (2%) | 100% | 0% | - | v2.0.2 (100%) |
| Select2 [30] | 2,537 (0.3%) | 78.8% | 21.2% | 38.6% | v4.0.3 (35.2%) | OWL Carousel [46] | 2 (2%) | 100% | 0% | - | N/A[5] |
| SweetAlert2 [52] | 2,357 (0.3%) | 50.8% | 49.2% | 9.2% | v7.26.11 (61.2%) | Prototype [222] | 2 (2%) | 0% | 100% | - | N/A[7] |
| Polyfill [159] | 2,226 (0.3%) | 6.6% | 93.4% | 49.5% | v3 (75.1%) | LazySizes [90] | 2 (2%) | 100% | 0% | - | N/A[5] |
| Emotion [129] | 2,025 (0.3%) | 62.8% | 37.2% | 0% | v11.9.0 (24.0%) | Lightbox [185] | 2 (2%) | 100% | 0% | - | v2.2.3 (50.0%) |
| LazySizes [90] | 1,998 (0.3%) | 45.6% | 54.4% | 48.8% | v2.9.5 (28.3%) | web-vitals [142] | 2 (2%) | 100% | 0% | - | N/A[5] |
| Hammer.js [9] | 1,771 (0.2%) | 93.1% | 6.9% | 75.0% | v2.0.4 (50.4%) | Datatables [231] | 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 [31] | 1,645 (0.2%) | 1.5% | 98.5% | 49.9% | v1.0.0 (34.5%) | Moment.js [40] | 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.

### JavaScript Library Usage

In this section, we examine the prevalent JavaScript libraries in phishing websites, with a focus on the dominant libraries, their 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 [173, 235]. This proportion is smaller than the `jQuery` usage (83.9%) reported in prior work [173], 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). The details are shown in Table 4.3

### 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 4.3 (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` [45] 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. 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 4.3 (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. 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. This implies that the phishing attackers want to avoid disclosing their information (*e.g.*, the hosting server's IP information) to Google.

`Socket.IO` [50] 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. 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 4.3), 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 4.3 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..

**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.

#### 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 [170, 195] 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 remains basic and rudimentary.

### 4.1.3 HTML Structures in Phishing Websites

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` [190] 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 [144]. We first run this tool with all collected HTML files within the top 10 clusters based on the number of distinct phishing domains for a more rigorous analysis, as shown in Table 4.4. 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

Table 4.4: **Top 10 Cluster by the Number of Phishing Domains with Similarity, Target Brand, First Seen, Mimicked-Date, and Date Difference.**

| 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.

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, one example combines resources from DHL and USPS target websites. 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', 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.

### 4.1.4 Other Client-side Resources in Phishing Websites

#### 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.

## 4.2 Client-side Resources in Benign Websites

Modern Websites commonly rely on various client-side web resources, such as JavaScript libraries, to provide rich and interactive end-user web experiences. Unfortunately, anecdotal evidence shows that improperly managed client-side resources could open up attack surfaces. However, there is still a lack of a comprehensive understanding of the updating practices among web developers and the potential impact of inaccuracies in Common Vulnerabilities and Exposures (CVE) information on the security of the web ecosystem.

**Our Research Questions.** We aim to understand the longitudinal security issues and implications of the client-side web resource ecosystem from various security perspectives. In particular, we aim to answer the following research questions.

- **RQ1)** How prevalent are insecure websites in the wild due to vulnerable JavaScript libraries?
- **RQ2)** How quickly do the insecure websites (*i.e.*, web developers) react to their vulnerable client-side resources and update them upon the release of the vulnerabilities and patches?
- **RQ3)** How does the accuracy of the CVE report impact the security of websites and developers' updating behaviors of vulnerable client-side resources as they rely on CVE information?
- **RQ4)** How does Adobe Flash exhibit vulnerabilities and what factors contribute to these vulnerabilities?

### 4.2.1 Dataset Collection

Our web crawler is written in the `Go` programming language. The crawler 1) periodically accesses Alexa 1M domains, 2) downloads the landing page (*e.g.*, `index.html`) from each domain, and 3) identifies client-side resources and their versions from the downloaded webpages. We also collect the vulnerability information of affected resources' versions from multiple third-party websites for cross-reference, such as the National Vulnerability Database (NVD) [41], CVE MITRE Corporation [35], cvedetails.com [36], and SNYK Vul. DB [55].

#### Landing Page Collection

We collect the webpages of Alexa Top 1M domains[†] on a weekly basis for four years (Mar. 2018 to Feb. 2022; 207 weeks); specifically, our Web crawler (implemented in `Go` using the `net/http` library) visits each Alexa 1M domain over HTTPS and collects the landing page of each domain every week. During the four-year data collection, we had very few times experienced network issues (*e.g.*, unavailable network connection).

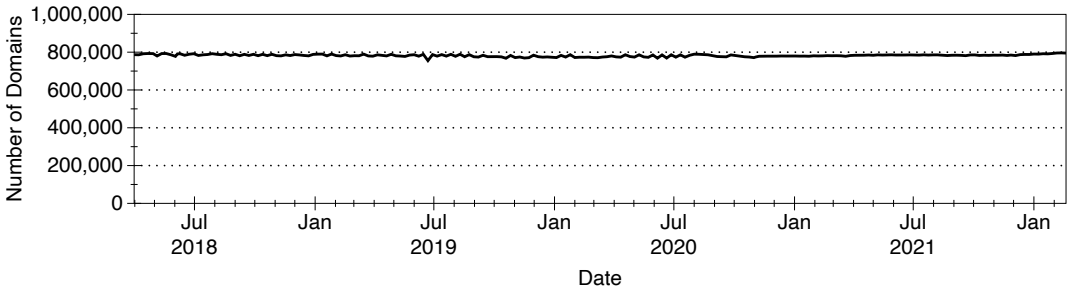
[†]We utilize the single snapshot of the Alexa Top 1M domains of Mar. 2018.

We prune out 6 snapshots (out of 207 snapshots) and remove inaccessible domains from the collected dataset.

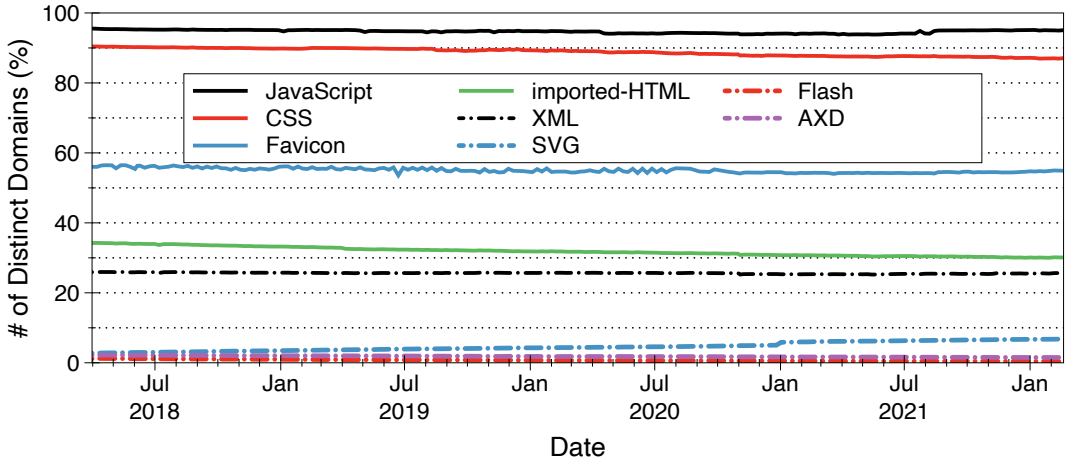
As expected, for four years (our collection period), we observe a number of inaccessible domains due to expired domains or unstable web servers. Particularly, we observe some of the lower-ranked domains are unstable in providing their services to clients. Moreover, we occasionally observe empty HTML pages and anti-crawling-bots blocking techniques (*e.g.*, '4xx' error code).

As such inaccessible domains may introduce bias into our analysis of the collected dataset, we conservatively remove them from our collected dataset. Specifically, we filter out the domains responding with error pages (*e.g.*, with '4xx' error status code) or empty pages (less than 400 bytes) for the four consecutive weeks in the last month of our data collection period. The reason why we consider 400 bytes as a threshold for the empty/error pages is that we manually check all HTML pages with less than 400 bytes and observe that all of them are either error or empty pages. Note that we manually check all of such pages in our dataset and confirm that they do not contain content related to the website's original purpose (*i.e.*, service). Instead, they present error messages (with 200 status codes) from anti-crawling-bots blocking techniques, saying "Not allowed to access."

To this end, we eventually collect 157,242,243 (157.2 M) HTML files for the 201 weeks of the four years. On average, we consistently collect the index pages from the 782,300 domains (*i.e.*, websites) every week, as shown in Figure 4.2(a). The proportion of domains that we successfully collected is similar to what has been reported in prior work [233, 226, 235].




(a) Our Collected Websites for Four Years.




(b) Top 8 Client-side Resource Usages (%).

Figure 4.2: **Our Collected Websites & Top 8 Client-side Resource Usages.** (a) On average, 782,300 websites are collected every week for four years (201 weeks). (b) On average, 94.7% of websites (740,823 out of 782,300) include JavaScript.

### Identifying Resources and Versions

To better understand the current Web ecosystem, we first need to know what client-side resources (*e.g.*, JavaScript libraries, Adobe Flash applets, etc.) are used in each website from our collected HTML files and identify their versions from the resources. The identifications of client-side resources and their versions can help answer our research questions (RQ1, RQ2, and RQ4). We utilize a website profiling tool, called `Wappalyzer` [56] that is widely used in prior work [123, 238, 189, 150, 223, 98, 127] and can identify client-side resources and their versions on webpages. Specifically, given a static HTML file, `Wappalyzer` uses regular expressions to identify the client-side web resources such as JavaScript libraries, Adobe Flash, CSS, and their versions in the given HTML file.

### Collecting Vulnerability Information

After identifying client-side resources and their versions, we attempt to identify vulnerabilities in each version of the client-side resources and collect the vulnerability information. Since there is no centralized database for vulnerabilities, we manually search each client-side resource with its versions and collect information on vulnerabilities[‡] from National Vulnerability Database (NVD) [41], CVE MITRE Corporation [35], cvedetails.com [36], and SNYK Vul. DB [55].

### Overview of Resources

From our collected dataset, we first measure various types of client-side resources that have been used in the web ecosystem. Figure 4.2(b) shows the top 8 client-side resources. JavaScript is the most used resource in the wild; on average, 94.7% of websites have at least more than one embedded client-side JavaScript code in their HTML code or URLs of external client-side JavaScript files. CSS (88.4%) is the next

[‡]We focus on only vulnerabilities in conjunction with our collection period.

most frequently used resource, followed by Favicon (55.0%), and imported-HTML[§] (31.8%). XML occupies 25.6%, and all the remainder (i.e., SVG, Adobe Flash, and AXD) account for less than 2.4%. Of 94.7% (740,823 out of 782,300) websites that use JavaScript, 97.04% (718,895 out of 740,823) websites use JavaScript libraries, meaning that the libraries are incredibly prevalent in practice. We further investigate popular JavaScript libraries in the dataset. In total, we find 79 distinct libraries in our dataset, and Table 4.5 shows the top 15 of them. Specifically, 64.0% of the websites (500,364 out of 782,300) use `jQuery`, followed by `Bootstrap` (21.5%), and `jQuery-Migrate` (20.8%). We further discuss the longitudinal landscape of JavaScript libraries. From our observations, we raise follow-up research questions regarding JavaScript libraries; **RQ1-1)** How many vulnerable JavaScript libraries are used in the wild?; **RQ2-1)** How the vulnerable JavaScript libraries are patched in practice?; **RQ3-1)** How does the CVE report impact the practices for securing websites? We aim to answer those research questions using our dataset. As shown in Figure 4.2(b), on average, 0.7% (5,678 out of 782,300) websites use Adobe Flash. We ask a follow-up question regarding Adobe Flash; specifically, **RQ4-1):** What factors contribute to the usage of Flash even after the suspension of their official support?

Table 4.5: **Top 15 JavaScript Usage, Inclusion Type, Version, and Vulnerabilities.**

| | | Inclusion Type | | | Version | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Library** | **Avg. Usage (%)** | **Avg. Int.**[1] | **Avg. Ext.**[1] | **Avg. CDN**[1,2] | **Found**[3] | **Total**[4] | **Avg. Dominant** | **Latest**[5] | **# Vul.**[6] |
| jQuery [164] | 500,364 (64.0%) | **59.2% (9.2%↓)** | 40.8% (3.9%↓) | **96.1%** (6.9%↓) | 81 | 81 | v1.12.4 (14.7%) | v3.6.0 | **8** |
| Bootstrap [112] | 168,088 (21.5%) | **71.6% (8.2%↓)** | 28.4% (11.8%↑) | 70.7% (10.3%↑) | 62 | 64 | v3.3.7 (24.3%) | v5.1.3 | **7** |
| jQuery-Migrate [166] | 163,386 (20.8%) | **88.4% (18.1%↓)** | 11.62% (44.4%↑) | 42.6% (60.3%↑) | 16 | 16 | v1.4.1 (50.4%) | v3.3.2 | **1** |
| jQuery-UI [167] | 95,058 (12.2%) | 49.7% (8.9%↓) | **50.3%** (18.9%↓) | **91.9%** (21.6%↓) | 47 | 47 | v1.12.1 (14.2%) | v1.13.1 | **6** |
| Modernizr [193] | 74,129 (9.5%) | **78.1% (20.1%↓)** | 21.9% (18.9%↓) | 68.2% (27.9%↓) | 26 | 43 | v2.6.2 (14.8%) | v3.11.8 | 0 |
| JS-Cookie [39] | 25,601 (3.3%) | **80.5% (14.2%↑)** | 19.5% (53.3%↑) | 86.5% (59.2%↑) | 23 | 23 | v2.1.4 (86.4%) | v3.0.1 | 0 |
| Underscore [244] | 19,614 (2.5%) | **83.2% (17.3%↑)** | 16.8% (47.3%↑) | 49.7% (49.2%↑) | 15 | 75 | v1.8.3 (9.5%) | v1.13.2 | **1** |
| Isotope [158] | 13,868 (1.8%) | **90.8% (6.8%↓)** | 9.2% (35.1%↑) | 24.6% (21.7%↑) | 28 | 42 | v3.0.4 (15.6%) | v3.0.6 | 0 |
| Popper [220] | 13,539 (1.7%) | 46.9% (79.7%↑) | **53.1%** (80.8%↑) | **92.0%** (80.2%↑) | 46 | 133 | v1.14.3 (24.3%) | v2.11.2 | 0 |
| Moment.js [194] | 12,827 (1.6%) | **70.4% (14.2%↑)** | 29.6% (26.8%↑) | 71.6% (27.3%↑) | 63 | 89 | v2.18.1 (8.2%) | v2.29.1 | **2** |
| RequireJS [225] | 12,541 (1.6%) | **64.8% (1.9%↑)** | 35.2% (82.6%↓) | 28.1% (21.0%↑) | 36 | 37 | v2.3.6 (32.3%) | v2.3.6 | 0 |
| SWFObject [11][7] | 10,096 (1.3%) | **74.2% (55.5%↓)** | 25.8% (48.7%↓) | 63.3% (69.7%↓) | 3 | 3 | v2.2 (46.9%) | v2.2 | 0 |
| Prototype [221] | 7,782 (1.0%) | **81.2% (54.8%↓)** | 18.8% (37.5%↓) | 57.9% (47.8%↓) | 11 | 11 | v1.7.1 (43.4%) | v1.7.3 | **2** |
| jQuery-Cookie [165][7] | 7,582 (1.0%) | **63.3% (14.3%↑)** | 36.7% (9.0%↑) | 86.5% (10.6%↑) | 7 | 7 | v1.4.1 (64.4%) | v1.4.1 | 0 |
| Polyfill.io [219] | 6,755 (0.9%) | 14.5% (69.6%↑) | **85.5%** (40.7%↑) | 37.8% (56.8%↑) | 3 | 3 | v3 (65.5%) | v3 | 0 |

1: The number in parentheses indicates how much the usage has increased from our first observation date; ↑ increase, ↓ decrease.
2: Out of Externally-Hosted JavaScript Libraries. 3: Number of versions found in our dataset. 4: Total number of JavaScript library versions.
5: Latest version in our collected dataset. 6: Number of vulnerabilities reported during our observation period. 7: No longer maintained.

[§]We infer this from the file extension (`.php`) in HTML tags (*e.g.*, `<script>` or `<link>`). Note that those PHP scripts are used to dynamically generate client-side resources such as JavaScript and CSS.

### Vulnerable JavaScript Libraries

We measure the JavaScript libraries to understand vulnerable websites due to vulnerable JavaScript libraries.

We first understand the current statistics of the JavaScript libraries' usage before we further investigate the security issues. We find an average number of 20.9 JavaScript libraries are included in a single website, and it is one of the dominant client-side resources in the wild. We first study the current landscape of JavaScript libraries such as usage trends, inclusion types, active/discontinued projects, and insecure JavaScript library versions with a focus on the Top 15 JavaScript libraries. These Top 15 libraries account for 96.5%.

**jQuery & Plugins.** `jQuery` is one of the most popular JavaScript libraries. It accounts for 64.0% (500,364 out of 782,300) of websites as shown in Table 4.5. While the usage of `jQuery` has been steadily decreasing over time from 67.2% (528,841) in Mar. 2018 to 63.1% (502,185) in Feb. 2022, as shown in Figure 4.3, it is still the most dominant library.

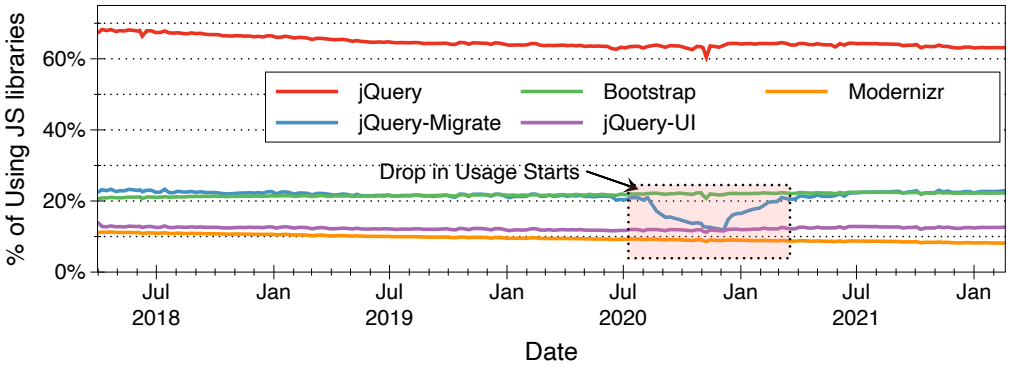
Observe that the third most popular library is a `jQuery-Migrate` [166] (used by 20.8% of websites on average), which is a `jQuery` plugin. The plugin aims to seamlessly migrate deprecated APIs of `jQuery` older than 1.9 so that the developers can use newer and secure `jQuery` versions without having compatibility issues. The plugin's popularity suggests that the JavaScript library update issues are critical in practice.

**jQuery-Migrate Usage Drop in Aug 2020.** Figure 4.3(a) shows (marked in the red box) a noticeable trend: the usage of `jQuery-Migrate` sharply dropping by approximately 10% for four months (Aug. 2020 – Dec. 2020), and then getting back (*i.e.*, went up by the 10%) in Dec. 2020. Our investigation on this dropping trend reveals that WordPress plays an important role. Specifically, WordPress versions earlier than v5.5 have been using `jQuery-Migrate` to handle compatibility issues. Then, WordPress v5.5 disables `jQuery-Migrate` by default, hoping web developers update the outdated jQuery code themselves [253]. This results in the dropping

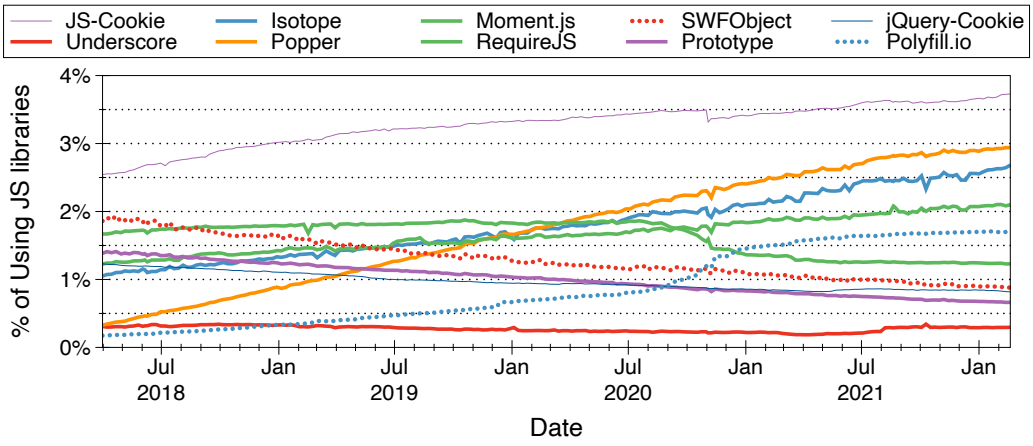
trend. Unfortunately, disabling the library resulted in numerous compatibility issues on websites using plugins and themes dependent on the old version of jQuery. As a temporary solution, WordPress introduced a new plugin '`jQuery Migrate Helper`' which essentially implements the same functionality of `jQuery-Migrate` [254]. In the next version (WordPress v5.6), WordPress officially re-includes `jQuery-Migrate` as a default library on Dec. 8th, 2020, leading to `jQuery-Migrate`'s usage increasing from Dec. 15th, 2020. As of May 2023, the latest WordPress 6 still includes `jQuery-Migrate`. One vulnerability of the library related to 'CWE-79: Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)' is reported from snyk.io [3], the GitHub issue [5] and a blog post [229], but no CVE ID is assigned to the vulnerability.

**Popular JavaScript Libraries and Trends.** Among the top 15 JavaScript libraries (in Figure 4.3), all libraries' popularities have been decreasing or steady except for four libraries, `JS-Cookie`, `Polyfill`, `Underscore`, and `Popper`. It may suggest that the security of a few popular JavaScript libraries can have a more significant impact than other libraries. `Polyfill` [219] includes polyfill code (*e.g.*, implementing the text shadow effect) for web browsers that do not support it. `JS-Cookie` [39] is a helper library for cookies. `Underscore` [244] provides various functional programming helpers (*e.g.*, map and reduce). `Popper` is a UI helper for tooltips and popovers [220].

**Inclusion Type: Internal vs. External.** Recall that there are two inclusion types: *internal* and *external*. Table 4.5 lists the percentage of the inclusion types for the JavaScript library. During our four-year observation period, the internal inclusion type (on average, 67.7%, 681,401) is more frequently used than the external type (on average, 32.3%, 325,520). However, external inclusion is also substantially used in popular JavaScript libraries: jQuery-`UI` (50.3%), `Popper` (53.1%), and `Polyfill` (85.5%). Note that, compared to internal inclusion, external inclusion has an additional attack surface because the systems delivering the libraries can be compromised. Table 4.5 shows 84.8% of the external inclusions are delivered by CDNs. In particular, over 90% of the external inclusions of `jQuery`, `jQuery-UI`, and `Popper` are delivered by CDNs.




(a) JavaScript Library Usage (Top 5) for 4 Years




(b) JavaScript Library Usage (Top 6 to 15) for 4 Years

Figure 4.3: **Percent of JavaScript Library Usage.**

The CDN usage has also been maintained steadily during our observation period. Table 4.6 presents the top 3 CDNs for libraries: ajax.googleapi.com, code.jquery.com, and cdnjs.cloudflare.com. Note that in the jQuery-`Migrate` case, `wp.com` is the most used one because the plugin is provided by WordPress [253], which is discussed more in Figure 4.2.1.

### Known Vulnerability using CVE Report

We utilize CVE (Common Vulnerabilities and Exposures) reports to identify vulnerable JavaScript libraries. CVE is a de-facto standard vulnerability reporting platform. Among the top 15 JavaScript libraries in Table 4.5, we find 27 CVE reports on seven libraries. Most vulnerabilities (20 out of 27) are related to XSS attacks, while others include one prototype pollution attack, one arbitrary code injection attack, two resource exhaustion attacks, one regular expression denial of service (ReDOS), and one missing authorization attack.

Table 4.6: **Top 3 CDNs for JavaScript Library.**

| Lib. | Hostname | % | Lib. | Hostname | % |
|---|---|---|---|---|---|
| jQuery | ajax.googleapi.com | 26.0% | isotope | secureservercdn.net | 3.3% |
| | code.jquery.com | 10.0% | | cdn.shopify.com | 2.1% |
| | cdnjs.cloudflare.com | 7.1% | | cdn.jsdelivr.net | 0.8% |
| jQuery-Migrate | c0.wp.com | 22.1% | popper | cdnjs.cloudflare.com | 77.3% |
| | cdnjs.cloudflare.com | 4.5% | | cdn.jsdelivr.net | 9.0% |
| | secureservercdn.net | 2.3% | | unpkg.com | 2.1% |
| Bootstrap | maxcdn.bootstrapcdn.com | 33.6% | polyfill | polyfill.io | 45.4% |
| | widget.trustpilot.com | 10.0% | | cdn.polyfill.io | 30.8% |
| | stackpath.bootstrapcdn.com | 9.7% | | static.parastorage.com | 4.1% |
| jQuery-UI | ajax.googleapis.com | 49.6% | moment | cdnjs.cloudflare.com | 51.8% |
| | code.jquery.com | 30.7% | | cdn.jsdelivr.net | 6.1% |
| | cdnjs.cloudflare.com | 4.2% | | momentjs.com | 1.7% |
| Modernizr | cdnjs.cloudflare.com | 32.4% | swfobject | ajax.googleapis.com | 49.1% |
| | cdn.shopify.com | 21.8% | | cdnjs.cloudflare.com | 3.0% |
| | cdn.prestosports.com | 1.0% | | s0.wp.com | 2.6% |
| JS-Cookie | cdn.jsdelivr.net | 21.1% | jquery-cookie | cdnjs.cloudflare.com | 62.6% |
| | c0.wp.com | 12.3% | | cdn.shopify.com | 8.4% |
| | cdnjs.cloudflare.com | 11.5% | | c0.wp.com | 0.9% |
| Underscore | c0.wp.com | 20.5% | prototype | ajax.googleapis.com | 27.7% |
| | cdnjs.cloudflare.com | 13.3% | | strato-editor.com | 3.7% |
| | secureservercdn.net | 1.5% | | cdnjs.cloudflare.com | 2.2% |

### Vulnerable Websites

We find an average of 41.2% of websites have libraries including at least one reported vulnerability (*i.e.*, CVE) in our dataset during our four-year observation period. This shows that even a few vulnerabilities impact many real-world websites, suggesting the significant security impact of vulnerable JavaScript libraries. As shown in Figure 4.4, for four years, an average of 0.79 vulnerabilities are carried per website (median: 0.75, max: 15.6). Updating the vulnerable versions is further discussed in Figure 4.2.1.

### jQuery & Plugins

`jQuery` has eight reported CVEs. In addition, its two plugins, `jQuery-Migrate` and `jQuery-UI`, have one and six vulnerabilities, respectively. On average, the vulnerable jQuery versions account for 37.7% (281,144 out of 782,300) of websites. CVE-2020-11023 [25] is the most impactful vulnerability that affects 56.2% of websites, including jQuery versions between v1.0.3 and v3.5.0, followed by CVE-2020-11022 [24] (56.1%) and CVE-2019-11358 [22] (54.6%). Even worse, a vulnerability (CVE-2012-6708 [16]) still accounts for 12.5% even though the vulnerability was reported more than a decade ago (in Jun. 2012).

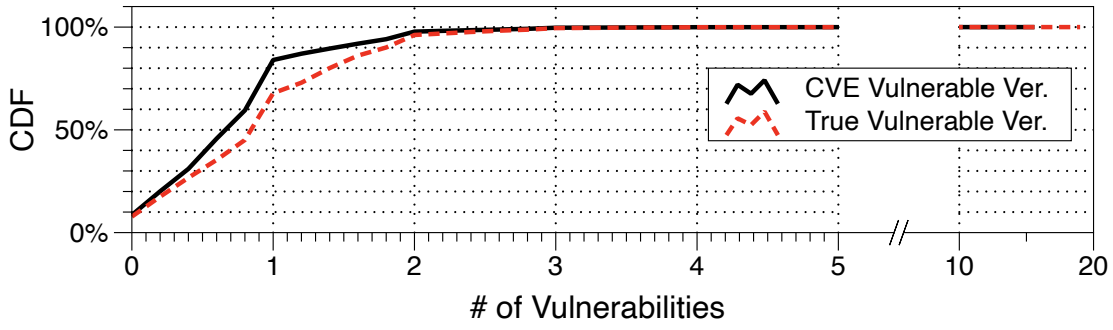



Figure 4.4: **CDF of the Avg. Number of Vulnerabilities per Website**. The average number of True Vulnerable Versions (mean: 0.97, median: 0.96) is higher than the one of CVE vulnerable versions (mean: 0.79, median: 0.75). This indicates that a significant number of True Vulnerable Versions are not disclosed due to incorrect CVE descriptions.

### Still Unpatched Library

All libraries we have analyzed in the paper, have been fixed, and the corresponding patches have been officially released, except for `Prototype` (CVE-2020-27511 [28], Regular Expression Denial of Service, or ReDOS). The vulnerability is critical as it affects all versions of the `Prototype` library. Unfortunately, it seems the vulnerability does not get proper attention from the developers. While we even find a discussion thread and a pull request for a bug fix [133] initiated in 2021, but it is not yet accepted and merged into the main repository.[¶] Such an unpatched vulnerability published in a CVE report can be exploited by adversaries. Hence, the CVEs and their corresponding patches indicate the security of JavaScript libraries in practice.

**Takeaway:** We observe a large number of websites (41.2%) in the wild contain vulnerable JavaScript libraries, that can be exploited by adversaries. The vulnerabilities are from high-profile libraries such as jQuery.

### Dominant Vulnerable Versions & Discontinued Library

Table 4.5 lists the most dominant version of each JavaScript library during our four-year observation period. Specifically, `jQuery` version 1.12.4, released in May 2016, is still dominant in the wild (accounting for 28.5% of websites). Unfortunately, this version has four reported vulnerabilities: three XSS (CVE-2020-11023 [25], CVE-2020-11022 [24], and CVE-2015-9251 [17]) and one Prototype Pollution (CVE-2019-5428 [22]).

While the newer versions, jQuery 3.0 (patched CVE-2015-9251 [17]) and 3.5 (patched all four vulnerabilities) are released in Jun 2016 and Apr. 2020, the old version (v1.12.4) is still dominant in the wild.

We further investigate various web developer communities and identify that the compatibility issues caused by the upgrade are a major concern [27, 2, 44, 21]. The compatibility issues are one of the reasons that lead to the dominant usage of such

[¶]We find that the project is not particularly active, and responses are slow. The last commit on the official repository of this library was in Apr. 2017.

outdated JavaScript libraries. For example, to resolve compatibility issues, jQuery-Migrate is developed and publicly released. This specific version (v1.4.1), released in May 2016, accounts for 50.0% of the total usage and helps resolve the compatibility issues resulting from jQuery v3.0 major changes since v3.0 is not fully compatible with jQuery before v1.12.3 or v2.2.3. Even though jQuery libraries in websites are updated to v3.0, most websites still need jQuery-Migrate v1.4.1 to use the legacy functions of jQuery before v1.12.3 or v2.2.3. This indicates that updating versions does not mean that legacy code is no longer used because such a library helps web developers use the legacy code of outdated libraries.

As shown in Table 4.5, we observe two discontinued JavaScript library projects: jQuery-Cookie [165] and SWFObject [11]. jQuery-Cookie is a jQuery plugin to help easily manage cookies. Although this library project is no longer maintained since 2015 (officially migrated and renamed to a new project, JS-Cookie [39][‖]), we observe many websites (7,582 websites) still use this discontinued library. We also measure how many websites migrated to JS-Cookie as recommended. We find, in total, only 7,800 websites (39% of the total 19,992 websites using jQuery-Cookie) are migrated to JS-Cookie. Note that 61% of websites still use the legacy library even after 7 years, meaning that updating JavaScript libraries is extremely slow in practice. Moreover, discontinued library projects typically do not address or patch the bugs if new bugs are identified. Developers are either recommended to use a newly migrated project [165] or use it with their own risk [11].

Another discontinued project is SWFObject, a JavaScript library to play Adobe Flash content on a webpage. Adobe Flash is officially no longer supported by Adobe, and major web browsers remove the Flash components. While this library project has been no longer maintained since 2013, it is still in use: an average usage of 1.3% (10,096 out of 782,300, ranked 12th) websites in our collected dataset. We find that WordPress plugins for SWFObject play a significant role in this Flash ecosystem; an average of 22.3% websites use WordPress plugins out of the total number of websites

[‖]As of May 2023, JS-Cookie is an active project; the recent commit was on Apr 24, 2023 [34].

using the `SWFObject` library. We also observe that 12.7% of the `SWFObject` is delivered by the Google CDN (`ajax.googleapis.com`). We informed Google that the project is discontinued and potentially insecure and may require further actions (*e.g.*, warning users or finding an alternative).

**Takeaway:** The dominant versions across all libraries are outdated and contain multiple vulnerabilities. We observe that backward compatibility is a major reason for preventing the update. Worse, 2.1% websites use discontinued libraries, exposing them to potential vulnerabilities.

### Accuracy of CVE Vulnerability Info.

CVE is critical information for understanding vulnerable programs and websites, potentially impacting various security practices such as updating. We aim to understand how reliable the version information in CVE reports is. As web developers may rely on CVE reports to measure how vulnerable the JavaScript libraries they use are. Incorrectly stated versions of CVE may impact subsequent decisions.

We manually investigate each CVE to validate the described vulnerability and affected versions using proof of concept (PoC) code in our controlled experiment environment. We set up the controlled experiment environment with each vulnerable library version and its required dependency. Particularly, for the `jQuery` vulnerabilities, in total, we set up 85 different environments for each different version from v1.0.0 to the latest version (v3.7.0, as of May 2023).

We find and utilize the existing seven PoC codes out of 27 CVEs (CVE-2020-7656 [23], CVE-2014-6071 [6], CVE-2018-20677 [18], CVE-2018-14040 [19], CVE-2016-10735 [13], CVE-2016-7103 [8], and `jQuery-Migrate` vulnerability [4]) For those PoCs that we initially failed to reproduce, we manually analyze the vulnerability and reimplement new PoC code. Particularly, for CVE-2020-7656 of `jQuery`, the existing PoC [23] described in the CVE report does not properly reproduce the vulnerability.

Therefore, we reimplement the PoC by removing the `jQuery` selector from the existing PoC code.

**Incorrect CVE information.** The version information can be incorrect in two ways: (1) CVE may understate the version, meaning more versions can be affected but not known to the public. (2) CVE may overstate the version, meaning that fewer versions are affected by the vulnerability than what is known to be public.

- Understated versions may cause delays in vulnerable websites' updates, as web developers do not realize that they use vulnerable libraries. For example, while CVE-2020-7656 mentions that it only affects 40 versions (lower than 1.9.1), from our experiments we find out that 79 versions (higher versions than 1.9.1, such as 1.10.1) are also affected by the vulnerability. This would make developers using version 1.10.1 believe that their websites are not vulnerable (hence less motivated to update).
- Overstated versions may cause ill-advised updates, leading to higher maintenance and development costs.

**True Vulnerable Versions (TVV) Identified.** True Vulnerable Versions (TVV) means actually-affected versions identified from our validation experiments. From our experiments, described in Table 4.7, we find that 13 out of the 27 CVEs contain incorrect versions. Specifically, five of them (⬇) have understated versions, meaning that the vulnerabilities affect more versions than reported. The remaining eight of them (⬆) have overstated versions, where the vulnerabilities affect fewer versions than reported. Particularly, Figure 4.5 illustrates the impact of incorrect versions in jQuery. For each CVE, there are two lines where the upper line represents vulnerable versions disclosed by the CVE and the lower line shows understated and overstated versions revealed by our Version Validation Experiment. In this jQuery case, 5 out of 8 CVEs have incorrect versions.

Particularly, CVE-2020-7656 specifies that the only affected version is lower than 1.9.1, while we reveal that it affects all versions until 3.6.0. Web developers who

Table 4.7: **Vulnerabilities of Top 15 JavaScript Library.** Among the top 15 libraries in Table 4.5, seven libraries are publicly reported to contain 28 vulnerabilities. The affected versions of the 12 vulnerabilities are incorrect.

| | CVE Vulnerable Ver. | | True Vulnerable Ver. | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Library** | **Ver. from CVE** | **# of Website** | **Ver.** | **# of Website** | **Patched Ver.** | **Disclosed[1]** | **Patched[2]** | **Attack Type** | **CVE ID** |
| jQuery | < 1.9.0 | 60,956 (12.2%) | < 3.6.0 | **291,579 (58.3%)** ⬇ | 1.9.0 | 05/19/2020 | 01/15/2013 | XSS | CVE-2020-7656 |
| | 1.0.3 ∼ 3.5.0 | 281,144 (56.2%) | 1.4.0 ∼ 3.5.0 | 276,956 (55.4%) ⬆ | 3.5.0 | 04/10/2020 | 04/10/2020 | XSS | CVE-2020-11023 |
| | 1.2.0 ∼ 3.5.0 | 280,968 (56.1%) | 1.12.0 ∼ 3.5.0 | 131,284 (26.2%) ⬆ | 3.5.0 | 04/29/2020 | 04/10/2020 | XSS | CVE-2020-11022 |
| | < 3.4.0 | 273,092 (54.6%) | – | – | 3.4.0 | 03/26/2019 | 04/10/2019 | Prototype Pol.[†] | CVE-2019-11358 |
| | 1.12.0 ∼ 3.0.0 | 88,757 (17.7%) | – | – | 3.0.0 | 06/26/2015 | 06/09/2016 | XSS | CVE-2015-9251 |
| | 1.4.2 ∼ 1.6.2 | 10,414 (2.1%) | 1.5.0 ∼ 2.2.4 | **214,078 (42.9%)** ⬇ | 1.6.2 | 09/01/2014 | 06/30/2011 | XSS | CVE-2014-6071 |
| | < 1.9.1 | 62,431 (12.5%) | < 1.9.0 | 60,956 (12.2%) ⬆ | 1.9.1 | 06/19/2012 | 02/04/2013 | XSS | CVE-2012-6708 |
| | < 1.6.3 | 16,857 (3.4%) | – | – | 1.6.3 | 06/05/2011 | 09/01/2011 | XSS | CVE-2011-4969 |
| Bootstrap | < 3.4.1, < 4.3.1 | 46,545 (27.7%) | – | – | 3.4.1, 4.3.1 | 02/11/2019 | 02/13/2019 | XSS | CVE-2019-8331 |
| | < 3.4.0 | 38,827 (23.1%) | 3.2.0 ∼ 3.4.0 | 36,019 (21.4%) ⬆ | 3.4.0 | 08/13/2018 | 12/13/2018 | XSS | CVE-2018-20676 |
| | < 3.4.0 | 38,827 (23.1%) | 3.2.0 ∼ 3.4.0 | 36,019 (21.4%) ⬆ | 3.4.0 | 01/09/2019 | 12/13/2018 | XSS | CVE-2018-20677 |
| | < 4.1.2 | 46,529 (27.7%) | 2.3.0 ∼ 4.1.2 | 46,287 (27.5%) ⬆ | 4.1.2 | 05/29/2018 | 07/12/2018 | XSS | CVE-2018-14042 |
| | < 4.1.2 | 46,529 (27.7%) | – | – | 4.1.2 | 05/29/2018 | 07/12/2018 | XSS | CVE-2018-14041 |
| | < 4.1.2 | 46,529 (27.7%) | 2.3.0 ∼ 4.1.2 | 46,287 (27.5%) ⬆ | 4.1.2 | 05/29/2018 | 07/12/2018 | XSS | CVE-2018-14040 |
| | < 3.4.0 | 38,827 (23.1%) | 2.1.0 ∼ 3.4.0 | 38,748 (23.1%) ⬆ | 3.4.0 | 06/27/2016 | 12/13/2018 | XSS | CVE-2016-10735 |
| jQuery-Migrate | < 1.2.1 | 956 (0.6%) | 1.0.0 ∼ 3.0.0 | **95,165 (62.8%)** ⬇ | 1.2.1 | 04/18/2013 | 09/16/2007 | | N/A* |
| jQuery-UI | < 1.10.0 | 13,948 (14.7%) | – | – | 1.10.0 | 09/02/2010 | 01/17/2013 | XSS | CVE-2010-5312 |
| | < 1.10.0 | 13,948 (14.7%) | – | – | 1.10.0 | 11/26/2012 | 01/17/2013 | XSS | CVE-2012-6662 |
| | < 1.12.0 | 42,856 (45.1%) | 1.10.0∼1.13.0 | **43,312 (45.6%)** ⬇ | 1.12.0 | 07/21/2016 | 07/08/2016 | XSS | CVE-2016-7103 |
| | < 1.13.0 | 57,261 (60.2%) | – | – | 1.13.0 | 10/27/2021 | 10/07/2021 | XSS | CVE-2021-41182 |
| | < 1.13.0 | 57,261 (60.2%) | – | – | 1.13.0 | 10/27/2021 | 10/07/2021 | XSS | CVE-2021-41183 |
| | < 1.13.0 | 57,261 (60.2%) | – | – | 1.13.0 | 10/27/2021 | 10/07/2021 | XSS | CVE-2021-41184 |
| Underscore | 1.3.2 ∼ 1.12.1 | 1,930 (9.84%) | – | – | 1.12.1 | 03/02/2021 | 03/19/2021 | Arb. Code Inj.[‡] | CVE-2021-23358 |
| Moment.js | < 2.19.3 | 4,322 (33.7%) | – | – | 2.19.3 | 09/05/2017 | 11/29/2017 | Res. Exhaust.[#] | CVE-2017-18214 |
| | < 2.11.2 | 2,115 (16.5%) | 2.8.1∼2.15.2 | **2,174 (17.0%)** ⬇ | 2.11.2 | 01/26/2016 | 2/7/2016 | Res. Exhaust.[#] | CVE-2016-4055 |
| Prototype | ≤ 1.7.3 | 7,782 (100%) | All versions | 7,782 (100%) | N/A** | 06/21/2021 | N/A** | ReDOS | CVE-2020-27511 |
| | < 1.6.0.1 | 4 (0.1%) | – | – | N/A*** | 02/03/2020 | N/A** | Missing Auth.[◦] | CVE-2020-7993 |

*: No CVE ID is assigned (vulnerability was found from snyk.io and GitHub issue.) **: No patched versions are available. ***: Affected version is no longer available.
1: Disclosed Date. 2: Patched Date. [†]: Prototype Pollution. [‡]: Arbitrary Code Injection. [#]: Resource Exhaustion. [◦]: Missing Authorization.
⬇: More versions are vulnerable than CVE's version (understated version). ⬆: Fewer versions are vulnerable than CVE's version (overstated version).

use a certain version (> 1.9.1) can be misled that their version is not affected by the vulnerability and do not need to update the library. We present 5 more cases (jQuery-`Migrate`, jQuery-`UI`, `Bootstrap`, `Moment`, and `Prototype`).

**Takeaway**: We discover that 13 CVE reports (out of 27) incorrectly state vulnerable versions of the libraries, providing misleading information. Specifically, 5 CVEs understate the impact of the vulnerabilities, potentially downplaying the vulnerabilities. 8 are understated versions.

**# of Websites Affected by Incorrect Versions.** We measure the number of websites affected by these incorrect versions in CVEs. Figure 4.6 shows the three cases of jQuery vulnerabilities; the red background refers to 'vulnerable websites not mentioned in CVEs (*i.e.*, websites using vulnerable libraries but were not spotted),' and the blue background does 'websites using vulnerable versions mentioned in CVEs.' Specifically, Figure 4.6(a) and Figure 4.6(b) show that a large number of websites

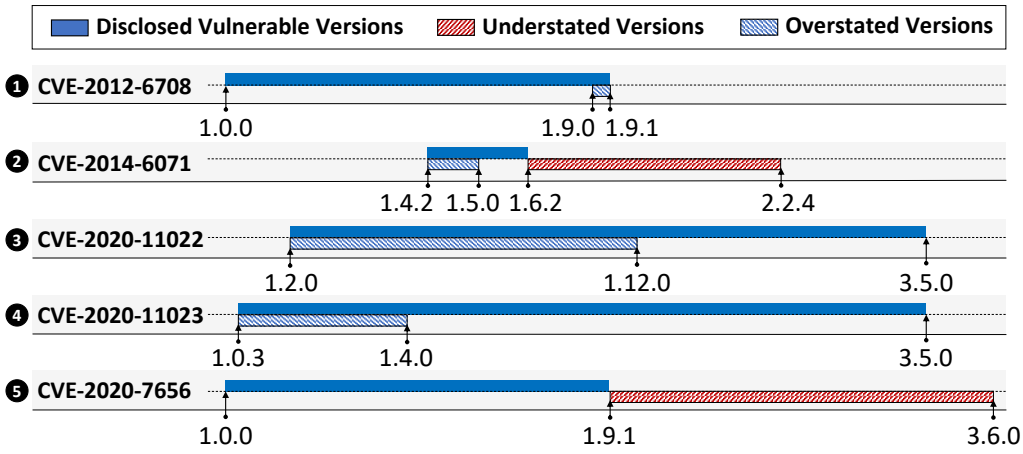



Figure 4.5: **Comparison of jQuery Disclosed Vulnerable Versions (Upper line) and Understated/Overstated Versions (Lower line)**. The Understated Versions (red stripes) represent the vulnerable versions we newly revealed. The Overstated Versions (blue stripes) indicate the versions we revealed are not vulnerable.

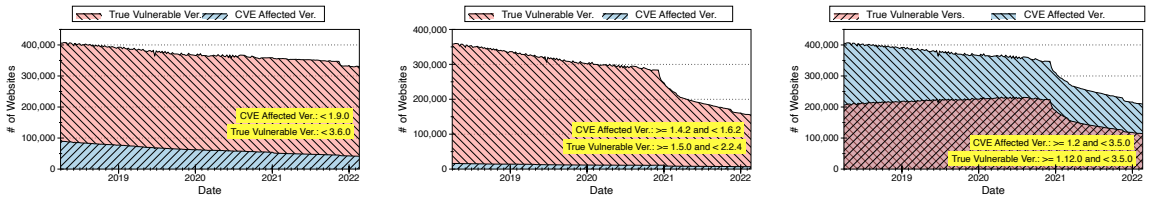
(296,818 in CVE-2020-7656 and 265,362 in CVE-2014-6071) may not be known whether they are susceptible to the vulnerabilities.

Meanwhile, Figure 4.6(c) shows that CVE versions unnecessarily include versions that are not vulnerable (*i.e.*, overstated versions), potentially misleading and causing unnecessary updates (and compatibility issues).

**Takeaway:** 337,773 websites are affected by incorrect version information in CVEs. 316,809 websites are undisclosed in the wild due to incorrect CVEs.

**Refining Vulnerable Websites.** Our previous conclusion that 41.2% are using at least one vulnerable JavaScript library in Figure 4.2.1 was based on the assumption that the CVEs' versions are correct, which we revealed that it is not true. Hence, we now measure the number of vulnerable websites again with the true vulnerable versions (TVV) we discover.

We find an average of ***43.2%*** (+2%) websites (337,773 out of 782,300) are vulnerable as carrying at least one vulnerability; the incorrect CVE information results in increasing the number of vulnerable websites by 2%. Interestingly, in 2018, the average difference between them is only 0.1%, but in 2022, the gap increases by 2.9%.




(a) CVE-2020-7656 (b) CVE-2014-6071 (c) CVE-2020-11022

Figure 4.6: **The Total Number of Websites with True Vulnerable Versions of jQuery Vulnerability.** (a) and (b) show that more vulnerable versions are revealed, while (c) shows that it reveals a fewer number of versions are vulnerable.

We find that it also affects high-profile (*i.e.*, popular) websites: `microsoft.com` (ranked 46th) and `onlinesbi.com` (ranked 111th) use jQuery v3.5.1, which we reveal that it is vulnerable while CVE does not mention (understate).

Also, `docusign.com` (ranked 1,693rd) uses jQuery v2.2.3, which is another understated case that we reveal it is vulnerable.

Furthermore, we also refine the CDF of the average number of vulnerabilities per website as shown in Figure 4.4. Compared to the results from the CVE's version, with the true vulnerable versions (TVV) we reveal, the average number of vulnerabilities per website is almost 1 (mean: 0.97 and median: 0.96), which is an alarming result.

**Takeaway:** With the corrected version information in CVEs, we observe that (337,773 websites (43.2%) are vulnerable (which is approximately 2% more than the analysis based on existing CVEs).

### Potential Security Threats of Untrustworthy External Libraries

**Potential Security Threat.** A website can include JavaScript libraries either internally (*i.e.*, locally hosted[**]) or externally (*e.g.*, remotely hosted[††]).

Potential security threats could arise when a web page loads compromised external JavaScript libraries. Compromised JavaScript libraries can obtain full privileges on the websites unless isolated in a frame; for example, the loaded JavaScript can modify

[**] *e.g.*, `<script src=‘‘./bar.js’’></script>`
[††] *e.g.*, `<script src=‘‘https://foo.com/bar.js’’></script>`

the DOM, load other external content, redirect the visitors to phishing websites, or deliver malware to the visitors. Particularly, if the external JavaScript libraries are loaded from repositories of collaborative version control, such as GitHub, GitLab, and Bitbucket, the libraries cannot be trusted because the repositories (and the libraries) can be compromised or malicious. In other words, we cannot fully establish trust in the maintainers and contributors of the open-source library projects, compared to the libraries hosted on the official CDNs. For example, a malicious contributor of a JavaScript library project could insert malicious payloads into the library, or an adversary could make a malicious pull request to the library project if these are public repositories [256].

**Libraries from Untrustful Sources.** We measure how many websites could be vulnerable due to the JavaScript libraries hosted on collaborative version control platforms, such as `GitHub`, `GitLab`, and `Bitbucket`. We find that an average of 1,670 websites load more than one external JavaScript library externally hosted on the 57 GitHub repositories. Particularly, the most popular GitHub repository is `wp-r.github.io`[‡‡] that accounts for 11.3% of the websites. It is an individual repository that hosts '`adsplacer`' and '`jquery. iframetracker`'. The first library is a WordPress plugin to help place advertisements, and the latter one is a jQuery plugin that can track users' clicks on iframes. The tracker is added on Adguard's Spyware Filter [86] to block trackers.

**Mitigation against Untrustful Sources (SRI).**

Subresource Integrity (SRI) [196] can be a viable security defense mechanism against the threat of untrustful JavaScript libraries. It ensures the unmodified JavaScript files that contain expected data are delivered, loaded, and executed. Specifically, web developers can specify an expected base64-encoded cryptographic hash (such as SHA256, SHA386, and SHA512) of a JavaScript file in the `integrity` attribute of the

[‡‡]https://github.com/wp-r/wp-r.github.io

`<script>` tag. At runtime, if a specified hash value does not match the hash value of the downloaded JavaScript file, it is neither loaded nor executed.

We measure how the SRI is used on websites in the wild. Surprisingly, as shown in Figure 4.7, 99.7% of websites have at least one externally-hosted JavaScript library without the `integrity` attribute, which cannot be trusted when external hosts are compromised. Furthermore, we closely take a look at the libraries hosted on the repositories of collaborative version control services such as GitHub. Of 1,670 websites using the external libraries on GitHub, only an average of 10.1 websites (0.6%) use the `integrity` parameter as the mitigation.

We further check how many official websites of JavaScript libraries explicitly mention the SRI (or the `integrity` attribute) in its code snippet; for example, they may provide a code snippet that includes an `integrity` attribute and hash value, which helps web developers simply copy and paste the secure code snippet. We find that out of the top 15 JavaScript libraries, only one library (`Bootstrap`) provides a code snippet that includes an `integrity` attribute and its hash value. As software developers (including web developers) are typically known to have the copy and paste behaviors [84], we argue that we might be losing chances to secure external JavaScript libraries by not having the `integrity` attributes in the example code snippets on the official websites of JavaScript libraries.

**Mitigation against Untrustful Sources (Crossorigin).** We also measure how properly developers use `cross origin` in the wild. `crossorigin` is an attribute to provide support for Cross-Origin Resource Sharing (CORS) and to control how to handle resources coming from cross-origin domains [37]. The best practice for cross-origin requests is to use the `anonymous` value because it prevents sending user credentials (*e.g.*, via cookies) to cross-origin requests (i.e., requests from a JS library fetched from a different origin). If '`use-credentials`' is specified in the `crossorigin` attribute, user credentials could be sent even for cross-origin requests [42, 252], leading to "cross-origin data leakage" [248].

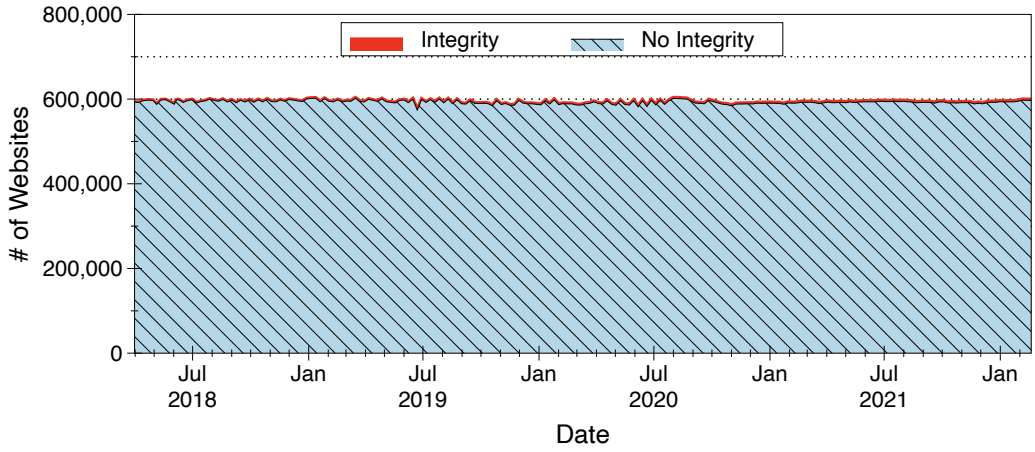



Figure 4.7: **Subresource Integrity (SRI).** 99.7% websites have at least one externally-hosted JS library without integrity.

We find 97.1% of websites use '`anonymous`' and only 1.9% use '`use–credentials`,' out of the total websites using `crossorigin` with the `integrity` attribute. This indicates that while most web developers properly follow the security best practice for this attribute, a few developers misconfigure the value for `crossorigin`. The small number of such websites may potentially leak credentials for cross-origin requests.

**Takeaway:** The mitigation (*e.g.*, `integrity`) to potential security threats of the externally-hosted libraries are barely used in the wild, which indicates such websites could remain vulnerable to potential security threats.

### Update of Vulnerable JavaScript Libraries

When a newly disclosed vulnerability affects JavaScript libraries used in a website and there is a patched version for the vulnerability, it is desirable to update the vulnerable libraries promptly. This section attempts to understand how well vulnerable JavaScript libraries are updated in practice over time.

Particularly, when a vulnerability is publicly disclosed and assigned a CVE ID, and a patch for the vulnerability is released, we measure how long (*e.g.*, how many days) it takes to update their affected vulnerable versions after the patched version's release. Note that in this section, we rely on the versions stated in CVEs even if they

might be inaccurate (*i.e.*, not the True Vulnerable Versions or TVVs). As a result, we aim to measure how the developers and administrators respond to the CVEs for security updates.

**Updating jQuery.** We focus on jquery as it is dominant in the wild – other libraries' updating patterns. Figure 4.8 shows a few versions' usage trends with the CVE disclosed dates; CVE-2020-7656 specifies that all versions lower than (< v1.9.0) are affected, and its patched version is v1.9.0 as described in Table 4.7. We observe that the usage of the patched version (v1.9.0) has not increased, but rather slightly decreased.

From this observation, we can learn that vulnerable JavaScript libraries are rarely updated, or worse, some websites newly start to use the vulnerable version even after the CVEs are publicly disclosed.

Furthermore, Figure 4.9(a) shows the updating trends of the most dominant version (v1.12.4), which is affected by CVE-2020-11022 and CVE-2020-11023.

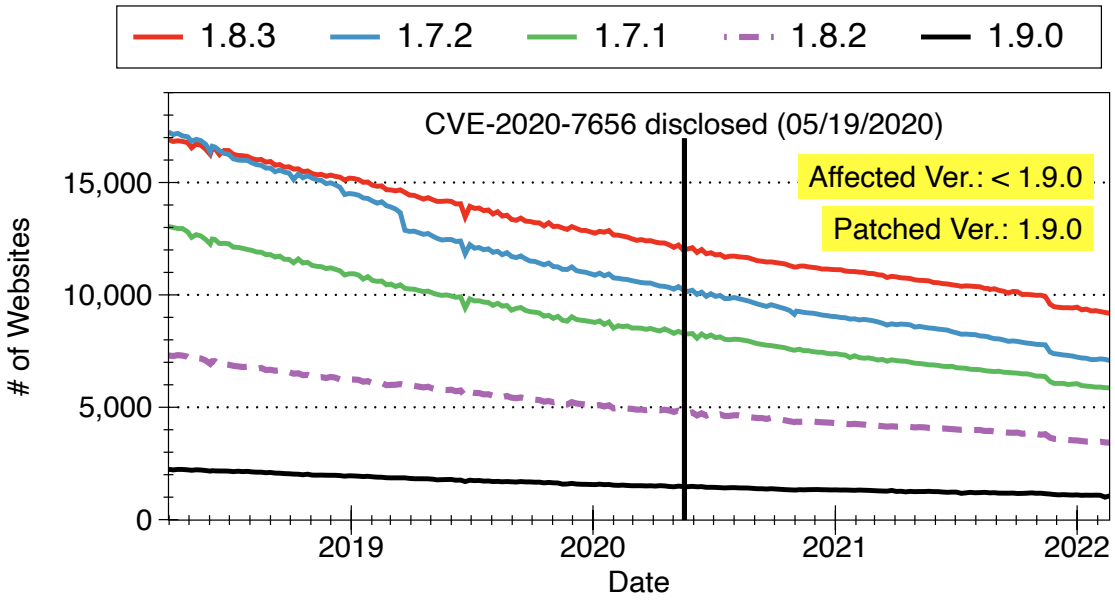
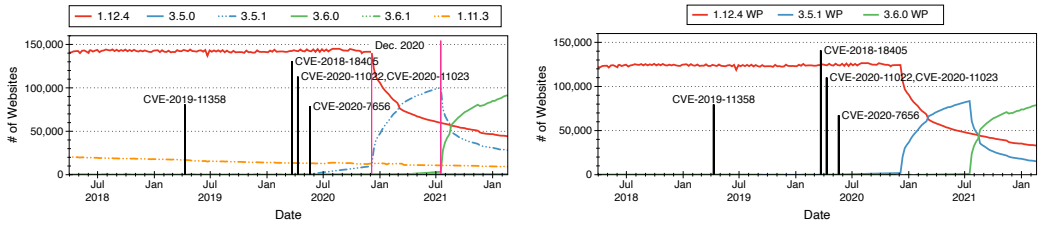



Figure 4.8: **Usage Trends of Affected Versions of jQuery CVE-2020-7656 Vulnerability** (versions are ordered by popularity).




(a) `jQuery` v1.12.4 Usage & Patched Version (< v3.5.0) Usage

(b) WordPress Versions Associated with jQuery Update

Figure 4.9: **Trends of Vulnerable jQuery v1.12.4 Usage and Patched Version (< v3.5.0) Usage & WordPress Versions Associated with the jQuery Update**.

From the 1.12.4 version, it is recommended to update to v3.5.0 or higher ($\geq$ v3.5.0). Hence, ideally, the usage of v1.12.4 should decrease, and the usage of the versions higher than v3.5.0 (including v3.5.0, v3.5.1, v3.6.0, and v3.6.1) should increase if web developers properly follow the best practice of updating vulnerable versions.

Unfortunately, we observe different usage patterns as shown in Figure 4.9(a). v3.5.0 is barely (nearly 0%) used in the wild, which means that no update operation is performed to v3.5.0; similarly, v3.6.1 has the same pattern. Observe that the v3.5.1 usage starts increasing one month after the CVEs are publicly disclosed (around July 2020). However, we do not believe that this is a potential update from v1.12.4 because v1.12.4 does not decrease when the usages of v3.5.0, v3.5.1, v3.6.0, and v3.6.1 start increasing.

Moreover, we observe a different update behavior from Dec. 2020. Specifically, in Dec. 2020, the v3.5.1 usage sharply increases while v1.12.4 accordingly decreases. We randomly choose 100 websites in our dataset and manually analyze how the versions are updated. The result shows that they are truly updated (i.e., replaced the old version with the new version). In addition, from Aug. 2021, we observe a sharp increase in version updates from v3.5.1 to v3.6.0.

**Takeaway:** We observe that it averagely takes 531.2 days (17.4 months) to update the vulnerable version. The updating trend is also not linear, implying that there should be a particular contributor to this radical update trend.

### Main Contributor to Updating

We further investigate to understand the major actors for the updates. Specifically, from our dataset, we notice a sharp increase in updating from Dec. 2020. We aim to answer the question: Who or what events drove this radical update behavior?

Interestingly, we find that WordPress has been the main contributor. Figure 4.9(b) shows the jQuery usage of WordPress over time. We can see the same patterns between the jQuery version usages (Figure 4.9(a)) and the WordPress' jQuery version usages (Figure 4.9(b)) where they have almost identical usage pattern changes for v3.5.1, v3.6.0, and v1.12.4. We further analyze how WordPress becomes the main contributor, and find the WordPress Auto-Updating feature [255] helps automatically update the old and vulnerable jQuery libraries at almost the same time, which leads the Web ecosystem to be more secure. From this observation, we can learn that automated mechanisms (*e.g.*, auto-updating) are effective in securing websites by updating JavaScript libraries.

**Takeaway:** WordPress's Auto-Updating feature contributes to the major update of jQuery. The Web security community may suggest a new auto-updating feature for the client-side resources to secure the Web ecosystem.

**Update Delays with True Vulnerable Versions.** We measure the updating delays with the True Vulnerable Versions.

We find that the understated CVE reports lead to more significant delays; on average, it takes 701.2 days (23 months, 1.9 years), compared to 510 days calculated with the versions from CVEs.

**Takeaway:** Due to the understated versions in CVEs, we find that the true delay in the update of vulnerable JS libraries is substantially more severe, +191.2 days, than our initial estimate.

### Insecure Adobe Flash

**Flash Usage & Case Study.** The support for Adobe Flash is officially terminated (Jan. 2021), meaning that Adobe will not fix security vulnerabilities anymore. As a result, all major web browsers claim that Adobe Flash components are removed from the browsers as of Jan 2021, and Flash can no longer be played on the browsers [12, 29, 26, 32]. Also, web developers are strongly recommended to replace Adobe Flash with HTML5 [89, 93]; Adobe even released a conversion tool for developers from Flash to HTML5 [88].

We empirically observe some websites using Adobe Flash applications and a new platform application promoting the use of Adobe Flash applications, even though Flash was officially no longer supported. This observation motivates us to raise the research question; *If so, how many websites still use Flash*? and *what websites and functionalities still depend on the Flash in the wild*? To answer these research questions, we measure the usage of Adobe Flash in the wild. `SWFObject` (*e.g.*, how many websites use the discontinued project), in this section, we better understand how many Flash contents (*i.e.*, `.swf` files) are embedded and played in websites.

As shown in Figure 4.10, the Flash usage has steadily decreased during our observation period (from 9,880 websites in Feb. 2018 to 4,218 and 3,195 websites in Dec. 2020 and Feb. 2022, respectively). However, an average of 3,553 websites still use Adobe Flash after its end of life.

**Case Study of Top 10K Websites.** We take a closer look at the top 10K websites that still use Adobe Flash after the end of life (Jan. 1st, 2021). Among the top 10K websites, we find thirteen websites still use Adobe Flash as of May 2023.

Six out of thirteen domains use Flash for visible, dynamic content on the webpages (*e.g.*, a banner, a dynamic image application, and a media player). The remaining seven cases include `.swf` files or links in the HTML code of the webpages, but the Flash files are neither visually displayed nor played on the web browser. In invisible

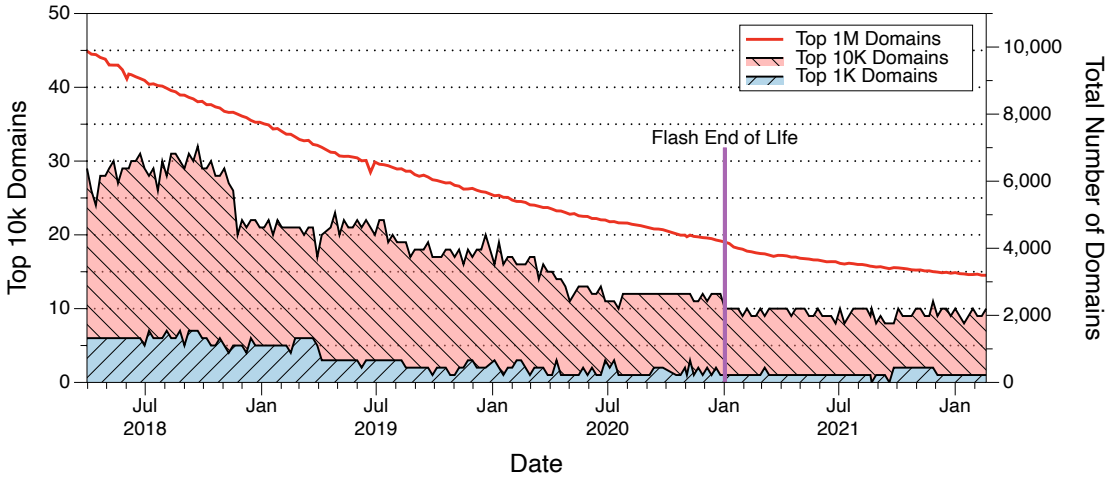



Figure 4.10: **Adobe Flash Usages.** The left y-axis indicates the number of domains for the top 1K and 10K domains, and the right y-axis is the number of domains for the top 1M domains.

cases, as `.swf` objects are positioned outside of the page or behind certain images, end-users cannot recognize Adobe Flash is being played on their web browsers.

We further analyze who manages and operates the websites and which country the websites are from. It turns out that four out of thirteen domains are managed and operated by Chinese companies; other countries are Hungary, Iran, Japan, Portugal, Spain, Russia, and Taiwan. This result motivates us to raise a follow-up research question, "*why do Chinese websites still use Adobe Flash more than other countries do even after the official support is no longer available, and all major browsers are no longer supporting Flash*?"

**Flash-Support Browsers & Ecosystem.** To answer the follow-up question, we manually examine the top 10 desktop Web browsers in the world, focusing on how they support Flash. We test them both on macOS 12.4 and Windows 10. Our results are summarized in Table 4.8. We find that all Web browsers no longer support Flash except for `360 Browser` [83]. This browser is based on Google Chrome and developed by a Chinese internet security company, Qihoo 360. The browser has two versions: `Secure` and `Extreme`. Particularly, `360 Extreme` v12.2.1662.0 for macOS (based on Chrome v78.0. 3904.108 was released in Nov. 2019) still supports Flash as of May.

Table 4.8: **Top 10 Web Browser Market share and Flash Support.** 360 Browser still supports Adobe Flash even though it is officially no longer supported and major browsers such as Chrome completely removed the Flash components.

| Browser | Market share* | Flash Support** |
|---|---|---|
| Chrome | 66.45% | N |
| Edge | 10.8% | N |
| Safari | 9.59% | N |
| Firefox | 7.16% | N |
| Opera | 3.09% | N |
| IE | 0.81% | N |
| 360 Browser | 0.66% | Y |
| Yandex Browser | 0.39% | N |
| QQ Browser | 0.20% | N |
| Edge Legacy | 0.16% | N |

*: Desktop Browser Market Share Worldwide (Apr. 2022 – Apr. 2023) [232]
**: Manually tested on May 26, 2023.

26, 2023. Moreover, when end-users access the Flash-embedded webpages, they are recommended to visit `www.flash.cn` to play the Flash contents. This website provides its own, customized version of the old Adobe Flash player components for end-users who still want to use Adobe Flash contents [257].

We can learn from these two cases (`360 Extreme` browser and `flash.cn`), this unique Flash ecosystem could play an important role in maintaining a number of Chinese websites using insecure Flash. This eventually leads users to remain exposed to the security threats of Flash.

**Takeaway:** An average of 3,553 websites still use Adobe Flash even after the end of the life of Flash. We find that a certain web browser and platform (`www.flash.cn`) still facilitate the use of Adobe Flash for end-users.

**Insecure AllowScriptAccess Parameter.**

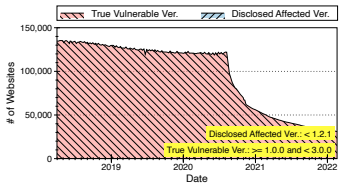
In the HTML code, the `AllowScriptAccess` parameter is to control if a `.swf` file (specifically, ActionScript) is allowed to call/access JavaScript and HTML DOM in the HTML page where the `.swf` is loaded. The parameter has three options; *always*, *sameDomain*, and *never* [87]. The *sameDomain* option allows a `.swf` file to call/access JavaScript and HTML DOM only when the `.swf` and HTML page are from the same

domain. The *never* option never permits the JavaScript calls and the HTML DOM accesses. If no value is specified to this parameter, the *sameDomain* option is by default applied. However, the *always* option permits a `.swf` file to always call/access JavaScript and HTML DOM even though the domain of the `.swf` file is different from the one of the HTML page. A potential security threat could arise when external and untrustful `.swf` files are embedded and loaded. Then, they can maliciously access, call, and manipulate JavaScript and HTML DOM. For example, in a web forum where anyone can include an external `.swf`, an adversary can upload or link a malicious `.swf` to the target forum. Therefore, web developers are strongly recommended not to use the *always* option for the parameter by Web Hypertext Application Technology Working Group (WHATWG) [249].

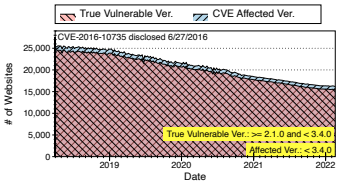
We measure if web developers properly follow the best practice (not to use `AllowScriptAccess` parameter) in the wild. We find that an average of 24.7% websites (out of the total number of websites using Flash) use insecure parameters. Specifically, the insecure usage has increased by approximately 9% (from 21% to 30%). This indicates that 24.7% of websites are vulnerable to malicious `.swf` files loaded from cross-origin sources.

**Revealed Vulnerabilities.** As we have seen in Figure 4.6, we showed the number of websites affected by incorrect CVE information. Figure 4.11(a) shows that `jQuery-Migrate` has a CVE stating that the versions before 1.2.1 are vulnerable. However, we revealed that the vulnerability expands from 1.0.0 to 3.0.0 (an understated version case). The red region in the graph shows the domains using newly revealed vulnerable versions. For Figure 4.11(d), and Figure 4.11(e) same principle applies to Figure 4.11(a). For Figure 4.11(b) and Figure 4.11(c), the CVE states that more versions are vulnerable than the versions that are truly vulnerable (an overstated version case).

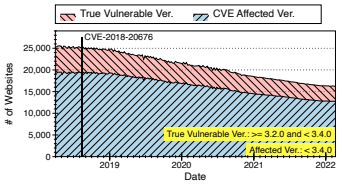
**Top 5 Affected Versions.** Figure 4.12 shows the top 5 affected versions for `Bootstrap`, `jQuery-UI`, and `Prototype`. Figure 4.12(a) has version 3.3.7 as the



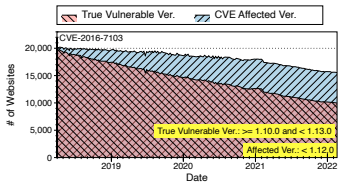

(a) jQuery-Migrate: NO CVE ID Assigned



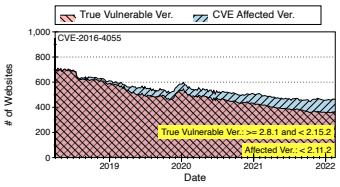

(b) Bootstrap: CVE-2016-10735



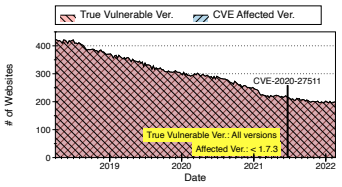

(c) Bootstrap: CVE-2018-20676




(d) jQuery-UI: CVE-2016-7103




(e) Moment: CVE-2016-4055



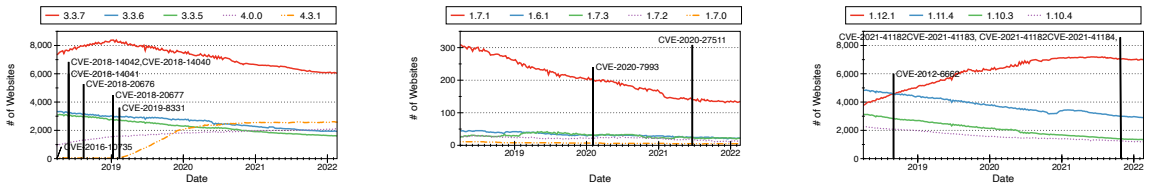

(f) Prototype: CVE-2020-27511

Figure 4.11: **CVV and TVV of jQuery-Migrate, jQuery-UI, Bootstrap, Moment, and Prototype.**

dominant version as shown in Table 4.5. However, it is affected by all of the CVEs we have discovered. From Figure 4.12(a), version 3.3.7 is decreasing. However, it is difficult to say it is caused by the disclosed vulnerability. Similarly, for Figure 4.12(b), version 1.7.1 is the most dominant version, and it decreases slightly. A similar result is shown but more pronounced because this clearly shows that the decrease is not affected by disclosed CVEs. For Figure 4.12(c), the dominant version is also in the top 5 affected versions. This figure also shows that there is no effect on updating behavior with disclosed CVEs.

**Comparison of Overstated Versions and Understated Versions.** As we have seen in Figure 4.5, a similar analysis is conducted. For `Moment`, `jQuery-Migrate`, `jQuery-UI`, and `Prototype`, all have understated versions indicated in red. As we can see with `jQuery-Migrate` and `Prototype`, more than half of the existing versions are understated versions. This indicates how inaccurate CVE information is. The `Bootstrap` only does not have understated versions, it only has revealed overstated versions. This is understated but has drawbacks.

**Top 10 Disclosed CVEs for WordPress.** Table 4.9 shows the top 10 disclosed CVEs for `WordPress` (the most recent 5 CVEs and the most critical CVEs). `WordPress`




(a) Bootstrap (b) Prototype (c) jQuery-UI

Figure 4.12: **Top 5 Affected Versions of Bootstrap, Prototype, and jQuery-UI.**

is one of the interesting factors in our observation of JavaScript library updates. `WordPress` is one of the most popular content management systems. In our dataset, 26.9% of websites use `WordPress`. `WordPress` has released a total number of 606 versions, excluding beta and RC (Release Candidate) versions. In our dataset, we found 521 versions (excluding the last two patches of each version from v3.7 to v5.9, and the 6.0 branch because they are released after our collection period). We further look into disclosed vulnerabilities for `WordPress`, and we found a total of 6,155 disclosed CVEs as of May. 2023. From 6,155 disclosed CVEs, we looked at the five most severe vulnerabilities (highest ranked in CVSS score) and five most recent CVEs (recent CVEs only have medium severity score). From what we have found, an average of 97.7% of websites is vulnerable according to the top 5 recently disclosed CVEs and 0.36% of websites are vulnerable according to the top 5 most severe CVEs. This indicates that `WordPress` tries to fix vulnerabilities as version updates and most websites are using relatively recent versions of `WordPress`. The `WordPress` has an Auto-Update function which helps administrators/developers to keep up with the most recent version of the software.

```
1  <div id="CVE-2020-7656">
2      <script>alert('Arbitrary Code Execution');
3  </script></div>
```

Listing 4.1: Modified PoC for CVE-2020-7656 (inject.html) [23]

Table 4.9: **Top 10 disclosed CVEs for WordPress** The first 5 are the most recent CVEs, and the last 5 are the most severe CVEs.

| CVE ID | Disclosed Date | Ver | Patched Ver | Patched Date | #Websites |
|---|---|---|---|---|---|
| CVE-2022-21664 | 01/06/2022 | 4.1.34 ∼ 5.8.3 | 5.8.3 | 01/06/2022 | 124,556 |
| CVE-2022-21663 | 01/06/2022 | 3.7.37 ∼ 5.8.3 | 5.8.3 | 01/06/2022 | 127,440 |
| CVE-2022-21662 | 01/06/2022 | 3.7.37 ∼ 5.8.3 | 5.8.3 | 01/06/2022 | 127,440 |
| CVE-2022-21661 | 01/06/2022 | 3.7.37 ∼ 5.8.3 | 5.8.3 | 01/06/2022 | 127,440 |
| CVE-2021-44223 | 11/25/2021 | < 5.8 | 5.8 | 07/20/2021 | 121,214 |
| CVE-2012-2400 | 04/21/2012 | < 3.3.2 | 3.3.2 | 04/20/2012 | 545 |
| CVE-2012-2399 | 04/21/2012 | < 3.5.2 | 3.5.2 | 06/21/2013* | 913 |
| CVE-2011-3125 | 08/10/2011 | < 3.1.3 | 3.1.3 | 05/25/2011 | 380 |
| CVE-2011-3122 | 08/10/2011 | < 3.1.3 | 3.1.3 | 05/25/2011 | 380 |
| CVE-2009-2853 | 08/18/2009 | < 2.8.3 | 2.8.3 | 08/03/2009 | 30 |

*: the vulnerability was disclosed more than a year before the patched version was released

## 4.3 Server-side Configurations in Phishing

Phishing attacks pose a significant threat to Internet users. Understanding the security posture of phishing infrastructure is crucial for developing effective defense strategies, as it helps identify potential weaknesses that attackers might exploit. Despite extensive research, there may still be a gap in fully understanding these security weaknesses. To address this important issue, this paper presents a longitudinal study of security configurations and vulnerabilities in phishing websites and associated kits.

### 4.3.1 Dataset Collection

As shown in Figure 4.13, our systematic measurement study comprises three phases: data collection, HTTP headers and phishing kits extraction, and categorization and analysis. We designed a web crawler that systematically accesses real-world phishing websites, gathering APWG URLs and removing errors via clustering. The crawler collects HTTP response headers, client-side resources (*e.g.*, HTML), screenshots, and phishing kits. Deployed from July 2021 to January 2024 (931 days), it accessed 16,742,415 (16.7M) phishing websites. The extraction phase focuses on HTTP headers (*e.g.*, CSP) and kit fingerprinting. The analysis phase involves host-level investigation,

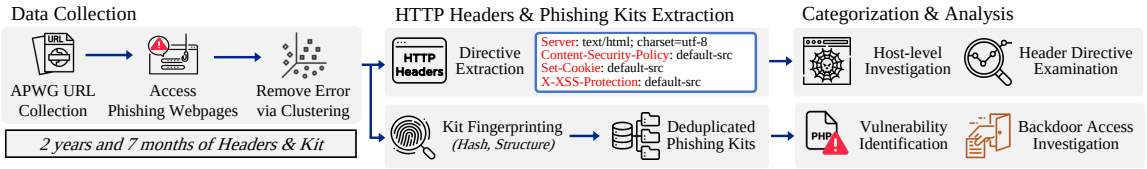



Figure 4.13: Overview of Our Systematic Measurement Study.

header examination, vulnerability identification (focusing on PHP), and backdoor access investigation.

### 4.3.2 Phishing Crawler Design

#### Phishing Website Resource Crawler Design

We design a web crawler that periodically (every 10 minutes) collects phishing website resources such as images, DOM, and HTTP response headers. Our crawler also captures screenshots of the phishing websites after fully loading and executing client-side resources. These screenshots help validate the authenticity of reported phishing URLs and detect potential access errors. We implement the crawler using Google Selenium ChromeDriver [57], which simulates real user interactions with phishing websites. This approach provides a comprehensive view of the phishing webpages by fully rendering all client-side resources and helps evade anti-bot techniques that might otherwise block our crawler [99, 179].

#### Phishing Kit Crawler Design

In addition to collecting phishing websites' client-side resources (*e.g.*, HTML) and HTTP header information, our approach also focuses on gathering phishing kits actively used in real-world attacks. However, identifying these kits poses a significant challenge due to the lack of specific information about their locations (*i.e.*, paths) and filenames on phishing web servers. To address this issue, our approach leverages the observation that phishing attackers may leave their kits publicly accessible and

downloadable at certain URL paths, even after deploying the phishing websites using these kits [118, 240].

The attackers' oversight enables our crawler to discover and download phishing kits from publicly accessible locations on compromised web servers. Specifically, the crawler visits each phishing URL and checks if directory listing (or directory indexing) is enabled. If it is, the crawler initiates a recursive process to download all files, including compressed archives (*e.g.*, zip, rar, tar, 7z) and other resources such as images, HTML, and JavaScript within the directory structure. When directory listing is not enabled, or no files are found in the initial directory, the crawler systematically navigates to the parent directory of the phishing URL in an attempt to locate accessible files.

### 4.3.3 Phishing Data Collection

#### Collecting Phishing Website Resources

The APWG eCrime Exchange `eCX` [81] is a global industry association of anti-phishing entities, including banks and financial services companies (*e.g.*, Paypal), Internet service providers (*e.g.*, ICANN, Adobe, Microsoft, and LinkedIn), law enforcement agencies, and security vendors (*e.g.*, RSA). APWG verifies the reported phishing websites from each anti-phishing entity and publishes the verified phishing URLs to its members. This repository has been widely used to better understand the phishing attack ecosystem [203, 202, 169, 259, 205]. Our crawler is fed real-world phishing URLs from the APWG eCrime Exchange (`eCX`) in real-time [81]. As illustrated in Figure 4.13, our crawler runs every 10 minutes from July 15th, 2021 to January 31st, 2024 (2 years and 7 months). During the collection period, our crawler is fed a total of 16,742,415 (16.7M) real-world phishing URLs from APWG `eCX`. Out of 16.7M phishing URLs, only 46.6% (7.81M) are accessible; the others are inaccessible due to network errors (*e.g.*, DNS), web servers being offline, etc. As APWG does not provide any information about false positives, we sample 1,000 reported phishing websites and

verify them. From our sample, we confirm that our list of phishing websites does not contain false positives.

### Refining Collected Phishing Websites Dataset

We leverage screenshots to filter internal errors, preventing bias from error pages with different HTML and headers. Using a conservative approach, we remove clusters with empty screens, client errors (*e.g.*, 400–499), and server errors (*e.g.*, 500–599). Our dataset initially contained 6.8M screenshots from 7.8M phishing websites, with 975K (12.8%) missing due to redirections or ChromeDriver issues. To automate filtering, we use `Fastdup` [97] to cluster 569,994 images and identify error messages. Two security researchers manually reviewed 23,579 clusters (90% coverage), refining the dataset to 544,173 clusters, representing 906,731 phishing websites, as shown inTable 4.10.

### Refining Collected Phishing Kit Dataset

Our phishing kit crawler collects URLs from the `eCX` platform every 10 minutes, retrieving 18,865 compressed (*e.g.*, `.zip`, `.7z`, `.rar`) or archived (*e.g.*, `.tar`) files from 1.2M phishing websites. To ensure dataset uniqueness, we employ a two-step deduplication: *First*, computing cryptographic hashes (Blake2) and *Second*, comparing directory structures and script filenames (*e.g.*, PHP, JavaScript, CSS, images).

Table 4.10: Overview of Our Collected Dataset from July 2021 to January 2024 (31 months).

| **Type** | **# of URLs** | **# of Websites*** |
|---|---|---|
| Total APWG Phishing Reports | 16,742,415 | 1,845,523 |
| Successfully Accessed URLs | 7,807,532 | 1,466,343 |
| Screenshots | 6,832,416 | 1,221,807 |
| Final Refined Dataset | 3,543,349 | 906,731 |

**# of Clusters from Refined** = 544,173;
**# of Total Kits** = 18,865; **# of Refined Kits** = 13,344;
∗ Distinct phishing websites;

If both match, the kit is classified as a duplicate and removed. *Lastly*, we validate phishing kits by analyzing `README.txt` contents for credential harvesting and exfiltration code, following prior methods [110, 240]. This process removes 5,521 duplicates (29.3%), averaging 4.4 duplicates per kit ($\sigma = 6.28$), with a maximum of 97. Our final dataset of 13,344 unique phishing kits – over 160 times larger than prior work (70 kits) [110] – undergoes security analysis using `Semgrep` [77] and `progpilot` [79].

### 4.3.4 Insecure HTTP Response Headers of Phishing Websites

We analyze HTTP response headers of phishing websites, focusing on 12 key security-related headers. Our study examines the overall header landscape, Cross-site mitigation headers, vulnerabilities linked to server information using Common Vulnerabilities and Exposures (CVE), and misconfiguration.

#### Categorization of Hosting Phishing Websites

Phishing websites are hosted via web content publishing services or self-hosting, impacting server configurations differently. Publishing services restrict attackers' control, while self-hosting allows full customization, including HTTP headers. Thus, our analysis focuses on self-hosting.

To classify websites, we use the Public Suffix List [75] to identify domains associated with content publishing services (*e.g.*, wix.com, github.io). Websites using these suffixes are categorized as content publishing services, while the rest are self-hosted. Our dataset reveals that 38.4% of phishing websites use content publishing services, whereas 61.6% are self-hosted Table 4.11.

Table 4.11: Top 30 Most Used Headers by Phishing Websites.

| Rank | Total Usage | | Content Publishing Service | | Self-hosting Server | |
|---|---|---|---|---|---|---|
| | Header | Usage (%) | Header | Usage (%) | Header | Usage (%) |
| 1 | Content-Type | 905,666 (99.9%) | Date | 346,118 (99.9%) | Content-Type | 559,549 (99.8%) |
| 2 | Date | 905,471 (99.9%) | Content-Type | 346,116 (99.9%) | Date | 559,352 (99.8%) |
| 3 | Server | 857,669 (94.6%) | Server | 315,632 (91.1%) | Server | 542,036 (96.7%) |
| 4 | Content-Encoding | 784,233 (86.5%) | Content-Encoding | 313,239 (90.4%) | Connection | 520,838 (92.9%) |
| 5 | Transfer-Encoding | 705,081 (77.8%) | Transfer-Encoding | 290,673 (83.9%) | Content-Encoding | 470,993 (84.0%) |
| 6 | Connection | 633,548 (69.9%) | Cache-Control | 281,289 (81.2%) | Transfer-Encoding | 414,408 (73.9%) |
| 7 | Cache-Control | 586,096 (64.6%) | Etag | 266,892 (77.1%) | Vary | 414,313 (73.9%) |
| 8 | Vary | 494,607 (54.5%) | Alt-Svc | 262,513 (75.8%) | Cache-Control | 304,806 (54.4%) |
| 9 | Expires | 482,550 (53.2%) | Last-Modified | 260,415 (75.2%) | Set-Cookie | 299,240 (53.4%) |
| 10 | Alt-Svc | 475,520 (52.4%) | Expires | 239,165 (69.0%) | Expires | 243,384 (43.4%) |
| 11 | Last-Modified | 349,289 (38.5%) | X-Content-Type-Options | 233,683 (67.5%) | Pragma* | 224,195 (40.0%) |
| 12 | Etag | 338,112 (37.3%) | X-XSS-Protection | 225,737 (65.2%) | Alt-Svc | 213,007 (38.0%) |
| 13 | Set-Cookie | 321,920 (35.5%) | Connection | 112,709 (32.5%) | CF-Ray | 201,163 (35.9%) |
| 14 | X-Content-Type-Options | 300,980 (33.2%) | Vary | 80,293 (23.2%) | CF-Cache-Status | 197,460 (35.2%) |
| 15 | X-XSS-Protection | 280,758 (31.0%) | Content-Length | 55,473 (16.0%) | Report-To | 191,950 (34.2%) |
| 16 | Pragma* | 242,841 (26.8%) | Strict-Transport-Security | 55,010 (15.9%) | NEL | 190,722 (34.0%) |
| 17 | CF-Ray | 228,085 (25.2%) | Accept-Ranges | 41,147 (11.9%) | Keep-Alive | 183,300 (32.7%) |
| 18 | CF-Cache-Status | 217,813 (24.0%) | X-Cache | 30,108 (8.7%) | Content-Length | 145,055 (25.9%) |
| 19 | Report-To | 212,972 (23.5%) | X-Served-By | 27,319 (7.9%) | X-Powered-By | 120,063 (21.4%) |
| 20 | Keep-Alive | 207,735 (22.9%) | X-Cache-Hits | 27,233 (7.9%) | Last-Modified | 88,874 (15.9%) |
| 21 | NEL | 207,342 (22.9%) | X-Timer | 27,228 (7.9%) | Etag | 71,220 (12.7%) |
| 22 | Content-Length | 200,529 (22.1%) | CF-Ray | 26,922 (7.8%) | X-Content-Type-Options | 67,297 (12.0%) |
| 23 | X-Powered-By | 129,992 (14.3%) | Keep-Alive | 24,435 (7.1%) | Accept-Ranges | 64,675 (11.5%) |
| 24 | Accept-Ranges | 105,822 (11.7%) | Set-Cookie | 22,680 (6.5%) | X-XSS-Protection | 55,021 (9.8%) |
| 25 | Strict-Transport-Security | 103,724 (11.4%) | Content-Security-Policy | 21,384 (6.2%) | Strict-Transport-Security | 48,714 (8.7%) |
| 26 | Expect-CT | 51,454 (5.7%) | Access-Control-Allow-Origin | 21,126 (6.1%) | X-Request-ID | 48,090 (8.6%) |
| 27 | X-Frame-Options | 51,280 (5.7%) | Report-To | 21,022 (6.1%) | X-Frame-Options | 45,520 (8.1%) |
| 28 | Content-Security-Policy | 49,130 (5.4%) | CF-Cache-Status | 20,353 (5.9%) | Expect-CT | 41,031 (7.3%) |
| 29 | Access-Control-Allow-Origin | 49,061 (5.4%) | Age | 19,872 (5.7%) | Upgrade | 37,045 (6.6%) |
| 30 | X-Cache | 48,817 (5.4%) | Pragma* | 18,645 (5.4%) | X-Host | 33,783 (6.0%) |
| **Total** | | 906,731 (100%) | | 346,366 (100%) | | 560,538 (100%) |

$*$ = Deprecated headers [67]; Insecure HTTP headers are highlighted in Red.

#### Security-related Header Types

We first identify security-related headers and categorize them into two groups: those that can actively mitigate security vulnerabilities and those that can introduce vulnerabilities if misconfigured. As shown in Table 4.12, eight headers (highlighted in orange) fall into the first category, while the remaining four headers (highlighted in cyan) belong to the second category. The 'Security Issues' column of Table 4.12 describes the types of vulnerabilities or mitigation associated with each header.

#### Collecting Benign Website Resources

To further compare the security configurations of phishing websites, we collect benign data from the top 10K domains on the Tranco 1M list [243], gathered on August 28, 2024. To further validate these findings, we examined the top 10 most targeted brand domains – `Facebook`, `Microsoft`, `Instagram`, `AT&T`, `WhatsApp`, `DHL`, `Ozon`, `USPS`, `PayPal`, and `Netflix` – using historical snapshots from Archive.org [105] across a consistent analysis period (July 2021 to January 2024). Using the same crawler employed for phishing websites, we collect resources from these benign domains, including HTTP header information.

### 4.3.5 Overview of HTTP Header Usage

#### General Usage of HTTP Headers

The top three headers used by phishing websites are `Content-Type` (99.9%), `Date` (99.9%), and `Server` (94.6%), similar to benign sites. Other common headers include `Content-Encoding` (86.5%), `Transfer-Encoding` (77.8%), and `Connection` (69.9%). Security-related headers are much less common, with `Set-Cookie` at 35.5%, `X-Content-Type-Options` at 33.2%, and `Strict-Transport-Security` at 11.4%. `Referrer-Policy` is used the least, at just 3.1%.

Table 4.12: Top 12 Security-related HTTP Headers Used in Content Publishing Service and Self-hosting.

| Security-related Header | Total Usage | | Content Publishing Service | | Self-hosting | | Security Issues |
|---|---|---|---|---|---|---|---|
| | Value | Usage (%) | Value | Usage (%) | Value | Usage (%) | |
| Content-Type | utf-8* | 788,996 (87.1%) | utf-8* | 323,495 (93.5%) | utf-8* | 465,501 (83.1%) | A resource might be read as HTML, creating potential for XSS vulnerabilities |
| | text/html | 112,912 (12.5%) | text/html | 22,101 (6.4%) | text/html | 90,811 (16.2%) | |
| | iso-8859-1* | 616 (0.1%) | iso-8859-1* | 35 (0.1%) | iso-8859-1* | 581 (0.1%) | |
| Server | Cloudflare | 227,984 (26.6%) | GSE | 220,319 (69.8%) | Cloudflare | 201,063 (37.1%) | Information leak |
| | GSE | 220,615 (25.7%) | Cloudflare | 26,921 (8.5%) | Apache | 161,709 (29.8%) | |
| | Apache | 184,004 (21.5%) | Apache | 19,624 (6.2%) | Nginx | 78,532 (14.5%) | |
| Set-Cookie | Path | 318,337 (98.9%) | Path | 22,330 (98.5%) | Path | 296,047 (98.9%) | Cookie transmission |
| | HTTPOnly | 199,334 (61.9%) | HTTPOnly | 13,358 (58.9%) | HTTPOnly | 185,976 (62.1%) | |
| | Secure | 162,386 (50.4%) | Secure | 10,764 (47.5%) | Secure | 151,622 (50.7%) | |
| X-Content-Type-Options | nosniff | 300,064 (99.7%) | nosniff | 233,664 (99.9%) | nosniff | 66,420 (98.7%) | Malicious content can be sent |
| X-XSS-Protection | 1;mode=block | 272,931 (97.2%) | 1;mode=block | 223,846 (99.2%) | 1;mode=block | 49,085 (89.2%) | XSS attacks |
| | 0 | 5,836 (2.1%) | 0 | 1,781 (0.8%) | 0 | 4,055 (7.4%) | |
| | 1 | 918 (0.4%) | 1 | 56 (0.1%) | 1 | 862 (1.6%) | |
| X-Powered-By | PHP | 111,864 (86.1%) | PHP | 7,028 (70.8%) | PHP | 104,836 (87.3%) | Information leak |
| | Plesklin | 9,898 (7.6%) | Plesklin | 2,298 (23.1%) | Plesklin | 7,600 (6.3%) | |
| | ASP.NET | 6,961 (5.4%) | express | 1,451 (14.6%) | ASP.NET | 6,265 (5.2%) | |
| Strict-Transport-Security | max-age | 102,720 (99.0%) | max-age | 55,009 (99.9%) | max-age | 47,711 (97.9%) | Can prevent HTTPS downgrade attacks |
| | includeSubDomains | 52,435 (50.6%) | includeSubDomains | 29,864 (54.1%) | includeSubDomains | 22,571 (46.3%) | |
| | preload | 36,249 (34.9%) | preload | 26,906 (48.9%) | preload | 9,343 (19.2%) | |
| Expect-CT† | max-age | 51,452 (99.9%) | max-age | 41,031 (99.9%) | max-age | 41,029 (99.9%) | Can prevent the use of misissued certificates |
| | report-uri | 51,249 (99.6%) | report-uri | 10,408 (99.8%) | report-uri | 40,841 (99.5%) | |
| | enforce | 170 (0.3%) | enforce | 50 (0.5%) | enforce | 120 (0.3%) | |
| X-Frame-Options | sameorigin | 41,164 (80.3%) | sameorigin | 4,083 (70.9%) | sameorigin | 37,081 (81.5%) | Can prevent iframe attacks |
| | deny | 6,972 (25.7%) | deny | 1,543 (26.9%) | deny | 5,430 (11.9%) | |
| | allowall | 2,302 (21.5%) | allowall | 107 (1.9%) | allowall | 2,195 (4.8%) | |
| Content-Security-Policy | upgrade‡ | 23,736 (48.3%) | upgrade‡ | 11,841 (55.4%) | upgrade‡ | 11,895 (55.6%) | Prevent cross-site-scripting attacks |
| | script-src | 18,562 (37.8%) | script-src | 7,590 (35.5%) | script-src | 10,972 (51.3%) | |
| | frame-ancestors | 12,046 (24.5%) | report-uri | 6,523 (30.5%) | default-src | 9,125 (42.7%) | |
| Access-Control-Allow-Origin | char (*) | 42,893 (87.4%) | char (*) | 21,105 (99.9%) | char (*) | 21,788 (78.0%) | Extends cross-origin resource sharing |
| | ¡origin¿ | 6,064 (12.4%) | ¡origin¿ | 14 (0.1%) | ¡origin¿ | 6,050 (21.7%) | |
| | null | 30 (0.1%) | null | 2 (0.1%) | null | 28 (0.1%) | |
| Referrer-Policy | strict-origin§ | 9,267 (54.1%) | strict-origin§ | 4,726 (84.8%) | strict-origin§ | 4,541 (39.3%) | Restrict the exposed referrer information |
| | when-downgrade‖ | 3,493 (20.4%) | same-origin | 315 (5.7%) | when-downgrade‖ | 3,424 (29.7%) | |
| | same-origin | 2,003 (11.7%) | unsafe-url | 274 (4.9%) | same-origin | 1,688 (14.6%) | |

Orange-colored headers can directly mitigate attacks; Cyan-colored headers can introduce vulnerabilities if misconfigured; *: 'text/html charset=html' is also included; †: This header is deprecated due to the lack of support by browsers; ‡: upgrade-insecure-requests; §: strict-origin-when-cross-origin; ‖: no-referrer-when-downgrade;

#### Content Publishing Service vs. Self-hosting Server

Phishing websites on content publishing services use more security-related headers than self-hosted sites, such as `X-Content-Type-Options` (67.5% vs. 12.0%) and `X-XSS-Protection` (65.2% vs. 9.8%). However, `Set-Cookie` is more common on self-hosted (6.5% vs. 53.4% ).

#### Adoption Trend of Security-related Headers

Figure 4.14 illustrates the usage trends of security-related headers in phishing websites. `Set-Cookie`, `X-Powered-By`, and `X-Frame-Options` increased significantly, while `Content-Type`, `Server`, and `Strict-Transport-Security` saw slight growth. `Expect-CT` declined after September 2022, whereas `X-Content-Type-Options`, `X-XSS-Protection`, and `Content-Security-Policy` remained stable.

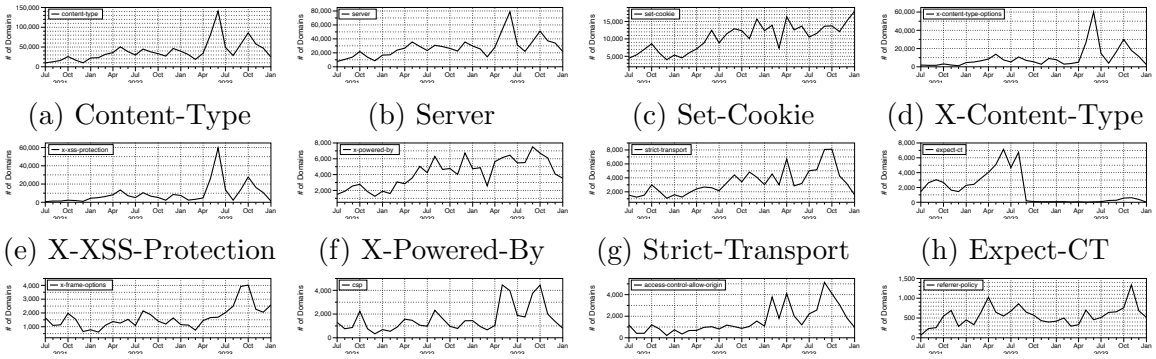
#### Comparison with Benign Domain

Prior studies highlight the increasing adoption of security headers like `CSP` and `Set-Cookie` in benign websites [226, 251, 134, 174]. Analyzing the top 10K domains from the Tranco 1M list [243], we find significantly higher adoption rates in benign websites compared to phishing sites: `Set-Cookie` (67.5% vs. 35.5%), `CSP` (75.2% vs. 5.4%), `X-XSS-Protection` (75.8% vs. 31.0%), and `Strict-Transport-Security` (15.9% vs. 11.4%).

Our longitudinal analysis of the top 10 targeted brand domains supports these findings, showing even higher adoption rates: `Set-Cookie` and `Strict-Transport-Security` (100%), `CSP` (80%), and `X-XSS-Protection` (60%).

### 4.3.6 Cross-site Mitigation Headers

Despite being a security standard since 2010, CSP adoption among phishing websites remains low, with only 5.4% implementing it. Content publishing services show slightly higher adoption (6.2%) than self-hosted servers (4.9%). The




(a) Content-Type (b) Server (c) Set-Cookie (d) X-Content-Type

(e) X-XSS-Protection (f) X-Powered-By (g) Strict-Transport (h) Expect-CT

(i) X-Frame-Options (j) CSP (k) ACAO (CORS (l) Referrer-Policy

Figure 4.14: Top 12 Security-related Headers Usage

`upgrade-insecure-requests` directive appears in just 2.6% of sites, while the critical `frame-ancestors` directive for framing control is found in only 1.3%. Additionally, 74.4% of domains use the `<meta>` tag for setting HTTP headers, but only 0.8% configure CSP this way. Misconfigurations are widespread, likely due to outdated phishing kits or automated tools lacking modern security standards. Among self-hosted phishing sites, 98.3% specify `unsafe-inline` in `script-src`, nullifying CSP protections, while 21.1% misuse `default-src`. Reporting mechanisms are also neglected: 3,439 sites use `report-uri`, and 93 use `report-to`.

Regarding TLS enforcement, 48.3% of phishing sites implement `upgrade-insecure-requests`, but only 9.7% of self-hosted sites combine it with the recommended `Strict-Transport-Security` header. Additionally, just 46.3% extend security to subdomains.

For framing control, phishing sites favor the outdated `X-Frame-Options` header over CSP's `frame-ancestors`. Misconfigurations are common: 21.5% misuse the invalid `allowall` directive, and 93.5% misconfigure `allow-from`. These issues highlight phishing sites' failure to properly implement CSP, reducing its effectiveness as a security measure.

#### Set-Cookie Header

The `Set-Cookie` header is widely used in phishing websites but often with overly permissive settings. Over 98% set the `Path` directive to '/,' making cookies accessible across all directories. While 61.9% use the `HttpOnly` attribute and 50.4% implement the `Secure` directive, a significant portion remains vulnerable. The `SameSite` directive, critical for preventing CSRF attacks, is poorly adopted – only 11.8% of self-hosted sites and 16.9% of all sites implement it. Although TLS enforcement via the `Secure` directive is more common (50.4%), 37.9% of sites still fail to implement `HttpOnly`, leaving them susceptible to XSS attacks.

**Takeaway:** Phishing websites rarely implement `CSP`, and when they do, misconfigurations weaken their effectiveness. While `Set-Cookie` is more common, key security directives are often missing. Higher TLS adoption likely boosts perceived legitimacy. These security header patterns can aid phishing detection and analysis.

### 4.3.7 Vulnerabilities in Phishing Websites

The `Server` header often reveals web server software and version details, increasing the risk of exploitation. Our data shows phishing websites are over twice as likely to expose version information compared to benign domains (14.3% vs. 6.5%), significantly expanding their attack surface. RFC 2616 [76] explicitly warns that disclosing server details poses a security risk.

#### Vulnerable Server Version Exposure

Many phishing sites run outdated and vulnerable server versions. `Apache` versions `2.4.6` and `2.4.41` contain 21 and 23 known vulnerabilities, respectively, while critical flaws like `CVE-2023-25690` [201] (severity score 9.8) affect versions `2.4.0` to `2.4.55`. `Nginx` has 75 known vulnerabilities, with versions `1.18.0` and `1.19.10` being the most

common. Even disclosing the server name alone is risky, as many `Apache` versions are known to be vulnerable.

#### Server Fingerprinting through Headers

The `X-Powered-By` header exposes server-side technologies, increasing exploitation risks. PHP is the most disclosed, with outdated versions like `5.4.16` (42 known vulnerabilities) still in use, making phishing sites potential attack targets. Additionally, `Cloudflare` and `GSE` (Google Servlet Engine) are common, with `GSE` primarily serving Blogger.com (99.9% of its usage), linking it closely to Google's infrastructure.

**Takeaway:** Phishing websites often expose server information through headers and `X-Powered-By` headers (14.3% vs 6.5% in benign sites), revealing software versions with known vulnerabilities in `Apache`, `Nginx`, and `PHP`. These exposed vulnerabilities present strategic opportunities for intelligence gathering and targeted disruption of phishing infrastructure.

### 4.3.8 Misconfiguration

#### Using Ineffective Directives

Headers enhance functionality but can pose security risks if misconfigured. For example, setting `X-XSS-Protection` to "0" disables XSS protection, yet 2.1% of websites in our dataset still use it. Similarly, 87.4% of websites use the wildcard (`*`) in `Access-Control-Allow-Origin`, allowing any origin access, with nearly all phishing sites (99.9%) on content publishing services doing so. Additionally, 11.7% of websites use the risky `unsafe-url` value in `Referrer-Policy`, exposing them to data leaks and security vulnerabilities.

#### Vulnerability of Non-standard Header

The `X-XSS-Protection` header, though intended to enable browser XSS filtering, can introduce XSS vulnerabilities under certain configurations. Despite its risks, usage has stabilized at 31.0% across phishing sites, with a particularly high prevalence (65.2%) on Content Phishing Services.

**Takeaway:** A misconfigured `X-XSS-Protection` header can introduce vulnerabilities, while common configurations in `Access-Control-Allow-Origin` expose websites to unauthorized access and data leaks.

### 4.3.9 Vulnerabilities in Phishing Kits

In this section, we look at CWE vulnerabilities and backdoors to code in phishing kits.

#### CWE in Phishing Kits

**Overview of Vulnerabilities.** 13,344 distinct phishing kits include 2,685,201 PHP scripts, each containing 205.93 PHP files on average. We run the two static analysis tools, `Semgrep` [77] and `progpilot` [79], on all the PHP scripts we collect. The result shows that phishing kits contain a wide range of vulnerabilities, which are categorized under the Common Weakness Enumeration (CWE) system. Specifically, out of 13,344 kits, 11,853 kits (88.8%) have more than one CWE reported. The result essentially shows that phishing kits might be vulnerable to various attacks. As shown in Table 4.13, the top three CWE categories identified by `Semgrep` are CWE-79 [62] (XSS, 73.71%), CWE-89 [63] (SQL Injection, 12.85%), and CWE-918 [64] (SSRF, 7.23%), accounting for 93.79% of the total CWEs detected. On the other hand, the top three CWE categories identified by `progpilot` are CWE-79 (XSS, 38.56%), CWE-1004 [59] (Insecure Cookie, 17.19%), and CWE-89 (SQL Injection, 16.38%), representing 72.13% of the total CWEs reported.

Table 4.13: Top 10 CWEs in Phishing Kits.

| ID | Type | LoE* | Severity | Semgrep # Vuln. (%) | Progpilot # Vuln. (%) |
|---|---|---|---|---|---|
| CWE-79 | XSS | M | M | 507,244 (73.71%) | 101,589 (38.56%) |
| CWE-89 | SQL Injection | H | M | 88,426 (12.85%) | 43,183 (16.38%) |
| CWE-95, 918 | SSRF | L | H | 49,755 (7.23%) | 5,398 (2.05%) |
| CWE-23 | Path Traversal | H | M | 22,254 (3.27%) | 16,932 (6.42%) |
| CWE-295, 319 | Cleartext Trans. | L | M | 9,484 (1.38%) | 14,827 (5.63%) |
| CWE-98, 489 | Leftover Debug | L | L | 3,265 (0.47%) | 12,209 (4.63%) |
| CWE-94, 601 | Code Injection | L | H | 2,244 (0.33%) | 23,221 (8.81%) |
| CWE-470, 1333 | Unsafe Redirect | M | M | 1,544 (0.22%) | 207 (0.08%) |
| CWE-78 | OS Command Inj. | L | H | 1,286 (0.19%) | 1,035 (0.39%) |
| CWE-614, 1004 | Cookies Set | L | L | 1,245 (0.18%) | 45,319 (17.19%) |

∗ **LoE** = Likelihood of Exploit; **H** = High; **M** = Medium; **L** = Low;
† Ordered by `Semgrep`'s number of Vulnerabilities

**Severity of CWEs.** To better understand the severity of the CWEs we found, we leverage semgrep's `Likelihood` metric [58], which aims to reflect the impact and ramification of the CWE and focus on the cases with high severity scores. Specifically, CWE-89 (SQL injection) and CWE-502 [61] (deserialization of untrusted data) are the two highly severe CWEs. CWE-79 (XSS) and CWE-22 [60] (path traversal) are of medium and low severity, respectively. While they are not highly severe, they can still lead to significant security breaches. Note that the weaknesses of phishing kits are likely to be reflected in the phishing websites they create.

### 4.3.10 Exploiting Vulnerabilities

This section presents a case study on XSS and SQL injection vulnerabilities in phishing kits, demonstrating how exploiting these vulnerabilities can actively disrupt phishing attacks.

#### Case Study of XSS Vulnerabilities

Listing 1 shows a code snippet from `darkx.zip`, a phishing kit used in at least 70 hosted phishing campaigns. The `verify.php` page (lines 1-5) contains an XSS vulnerability where unprocessed input from HTTP parameters is directly passed to an `echo` statement (line 3), rendering an HTML page and returning it to the user. This allows an attacker to inject JavaScript via the `email` parameter (line 4), executing malicious code in the victim's browser upon loading the page.

Once the victim submits the form, their email and password are sent to `next.php` (lines 6-19), where they are logged and exfiltrated. The script retrieves credentials from the `GET` parameters (lines 7-8) and stores them in a log file (line 11). If the `verify` parameter is 0, the script redirects users back to `verify.php` with `email` from the `GET` request and `error=true&verify=1` (lines 13-15), using a `window.location` without encoding, enabling another XSS attack. If `verify` is 1, the script starts a new session and redirects to the legitimate Microsoft login page via a meta refresh

tag (lines 16-19), likely to avoid detection. Since the code lacks input validation and output encoding, phishing websites using this kit remain vulnerable to further exploitation. Attackers can manipulate parameters in `verification` and `email` fields to inject malicious payloads, perform additional attacks, or bypass security checks within the phishing flow.

#### Case Study of SQL Injection Vulnerabilities

Listing 2 shows an SQL injection vulnerability that we manually verify. Specifically, in the `esestandard.zip` phishing kit, the `$_POST['username']` and `$_POST['password']` are directly concatenated into the SQL query without proper sanitization (lines 2-3). A maliciously crafted input containing malicious SQL commands can be injected and executed as part of the query. This allows the attacker to modify the query's logic, bypass authentication, or extract sensitive data from the database. In addition, the use of the deprecated functions (*e.g.*, `mysql_fetch_array()`) further increases the risk (lines 4-6). These codes are most often found in the code where the phisher sends information to the server and on the redirect page when the victim clicks the website's login button.

**Takeaway:** We identify multiple XSS and SQL injection vulnerabilities in phishing kits, indicating that websites built with them may also be exposed. Through manual verification, our case study demonstrates how these weaknesses can be leveraged to disrupt and neutralize phishing attacks.

### 4.3.11 Backdoor in Phishing Kits

From our collected phishing kits, we find that 13,344 kits are used across 6,825 phishing websites. We focus on backdoors as they are (1) critical for actively disrupting phishing websites and (2) the most prevalent malware type in our dataset. Specifically, 12.5% of phishing kits (1,668 out of 13,344) contain web shell backdoors, making them the

verify.php

```php
<?php
if (isset($_POST['signin'])) {
      echo "<script>window.location='next.php?
            email=".$_GET['email']."&password=".$_POST['password']
            ."&verify=0'</script>";} ?>
```

next.php

```php
<?php
$message .= "Email: ".$_GET['email']."\n";
$message .= "Password: ".$_GET['password']."\n";
// ... collect location info and user agent ...
$fp = fopen('.error.htm', 'a');
fwrite($fp, "\n".$message); fclose($fp);
// ... send email to attacker ...
if ($_GET['verify'] == 0) {
      echo "<script>window.location='verify.php?
            email=".$_GET['email']."&error=true&verify=1'</script>";
} elseif ($_GET['verify'] == 1) {
      session_start();
      echo "<meta http-equiv='refresh' content='0;
            url=https://login.live.com/'/>"; } ?>
```

Listing 1: Two Examples of XSS Vulnerability.

```php
<?php require_once('Connections/conn.php');
$user = $_POST['username'];
$pass = $_POST['password'];
$result = mysql_query("SELECT * FROM login WHERE
username='$user' AND password='$pass'") or die(mysql_error());
$row = mysql_fetch_array( $result );
// ... rest of the code ...; ?>
```

Listing 2: Example of SQL Injection Vulnerability.

most common type. Additionally, we identify 135 malware samples, including 124 trojans, 7 additional web shells, and 4 phishing pages. A manual analysis of 30 trojans reveals they are Remote Access Trojans (RATs), functionally similar to web shell backdoors. These findings highlight opportunities to leverage backdoors to actively disrupt phishing operations.

To further investigate intentional vulnerabilities within phishing kits, we analyze files containing XSS vulnerabilities that could allow unauthorized access. Using `shellray` [78] and `VirusShare` [82], we detect known malware and web shell patterns based on keywords such as `shell_exec` and `pcntl`. The results confirm that 1,668 phishing kits contain web shell backdoors, and 135 include other malware. To validate our findings, we activate and penetration-test the phishing kits in a controlled environment, adhering to ethical guidelines.

Figure 4.15(a) illustrates a backdoor we discovered in `pki-validation.zip`, which appears in 204 phishing campaigns. This backdoor maintains persistent access to phishing sites through a multi-step execution process. Deobfuscation reveals that the script first checks its execution context using `isCli()`. If not running in CLI mode, it bypasses restrictions by identifying PHP functions capable of executing system commands. It then attempts to access the '/robots' URL via `curlRobots()` to evaluate the server configuration. If successful, `runCmd()` executes a background command using `shell_exec()` or `passthru()`, establishing a persistent backdoor.

Moreover, the backdoor is stealthy. Once installed, the script deletes itself using `unlink()` to evade detection. It then calls `lockFile()`, which continuously monitors and modifies critical files. `lockFile()` scans for .htaccess, 'robots.txt,' and 'sitemap.xml' using `get_contents()` and calculates their SHA1 hashes via `hash()`. If changes are detected, the script overwrites these files with attacker-controlled content using `createFile()`, ensuring persistence. This mechanism can lead to a "theft chain of victims' information," where attackers exploit these backdoors to steal credentials already compromised by phishing operators. However, as shown in Figure 4.15(b), this backdoor control panel lacks proper traversal protections, making it easily accessible via subdirectory recursion (*e.g.*, ffuf [65]) with a wordlist. The control panel may provide attackers access to additional server information. Our findings indicate that many phishing kits are poorly secured, potentially allowing defenders or law enforcement to exploit these vulnerabilities for proactive phishing takedowns.

**Takeaway:** We find that 12.5% of phishing kits contain readily exploitable vulnerabilities, which might be abused by other adversaries who are aware of their existence. We speculate that both defenders and offenders can also leverage them.

## 4.4 Additional Security Headers

**Content-Security-Policy Configuration in HTML.** HTTP headers can also be configured using the `<meta>` tag in HTML, though this practice has been relatively uncommon in the past, as noted in previous research by Roth et al [226]. We investigate further how phishing websites configure headers via the `<meta>` tag and found that 74.4% of the domains (674,661) utilize this tag to set HTTP headers. However, only 0.8% of these domains (5,337) use the `<meta>` tag to set the `Content-Security-Policy` header.

## 4.5 Headers in Phishing Kits

We analyze the PHP script files included in the phishing kits and find HTTP header configurations in 10.73% of scripts (324,071 out of 3,019,018.

### Cache-control

`Pragma: no-cache` or `Cache-Control: no-store, no-cache, must-revalidate` are used to prevent caching of sensitive pages, which makes it easier for attackers to update and modify their phishing kits without the risk of serving outdated content. We find that 3,292 PHP files use the `Pragma: no-cache` header, while 3,001 PHP scripts employ the more comprehensive `Cache-Control` header with multiple directives.

### Redirection

Redirection is a method where headers like `Location` and `Expires` control the navigation flow within the phishing kit. Our analysis reveals that 7,778 PHP files within the phishing kits utilized relative URL paths in `Location` headers, such as `Location: ../../`, allowing attackers to redirect victims to different pages within the phishing kit. This tactic enables the creation of elaborate phishing campaigns

that guide victims through pages designed to harvest sensitive information, increasing the chances of successful attacks. We note that dynamic redirection, exemplified by variables like `location: $dst`, adds an additional layer of complexity to phishing kits. By enabling personalized navigation based on specific conditions or user interactions, attackers can create highly targeted and adaptive phishing experiences. In addition, we discover that 12,283 PHP files employed the `Expires` header to set expiration dates in the past, such as `Expires: Mon, 26 Jul 1997 05:00:00 GMT`. By explicitly telling the browser that the phishing page has already expired, attackers force a reload of the latest version, ensuring that victims interact with the most up-to-date iteration of the phishing kit.

### Content-Security-Policy in Phishing Kits

`Content-Security-Policy` headers are rarely utilized within the analyzed phishing kit, with only 10 out of the total PHP script files implementing strict CSP rules. Our analysis of 711 PHP phishing kit scripts shows that attackers manipulate CSP headers to weaken security and evade detection. Notably, they dynamically generate CSP rules, such as `frame-ancestors`, to control which sites can frame their phishing pages, as shown in listing3.

### Cloaking

`Cloaking` is a common method used in phishing kits to serve different content based on the characteristics of the incoming request.

```
1 public function removeCspHeader(ResponseEvent $event): void{
2     if ($this->debug && $event->getRequest()
3     ->attributes->get('_remove_csp_headers', false)){
4     $event->getResponse()->headers
5     ->remove('Content-Security-Policy'); } }
```

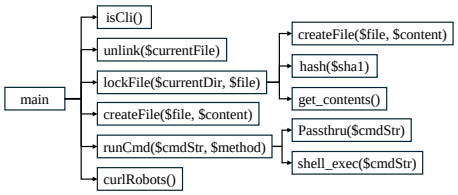
Listing 3: Example of Header Manipulation in PHP.

Our analysis reveals that 7,367 distinct PHP files within the phishing kits contained conditional HTTP responses, enabling attackers to deliver tailored content based on the requests' properties. By analyzing headers like `User-Agent` and `Referer`, phishing kit authors implement cloaking mechanisms that respond with deceptive error statuses. When a request originates from a suspicious source (*e.g.*, bots, crawlers, detectors), the phishing kit responds with deceptive error statuses like `HTTP/1.0 404 Not Found`. This cloaking behavior aligns with CWE-601, as it involves redirecting victims to malicious pages while showing benign content to security scanners. In addition, phishing kit authors may resort to obfuscation methods to conceal the cloaking logic and bypass detection. These involve encoding header directives using hex or octal representations.

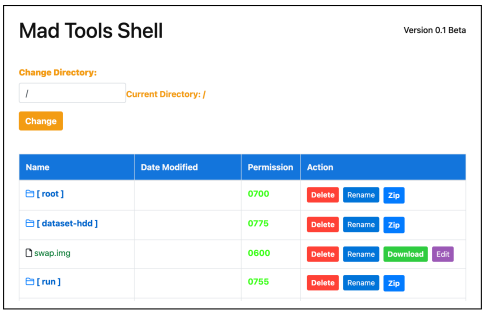
## 4.6 Phishing CPanel and Backdoor in Kits

Figure 4.15 presents two key components of a phishing operation: a reverse shell call tree and a phishing campaign control panel. These elements provide insight into the technical infrastructure and user interface employed by threat actors in sophisticated phishing attacks. The reverse shell call tree (Figure 4.15(a)) outlines a backdoor program enabling remote control of a compromised system. Its main function invokes tasks like environment checks, file manipulation, command execution, and web server interactions. Key functions include deleting, locking, creating files, and executing shell commands.

Figure 4.15(b) displays an example of a phishing campaign control panel labeled "Mad Tools Shell". This web-based interface, albeit in a beta version, provides attackers with a user-friendly way to manage their phishing infrastructure. The panel allows for directory navigation and file management on the compromised server. It lists files and directories with their permissions and modification dates and offers options to delete, rename, compress, and, in some cases, edit or download files. This interface




(a) Reverse Shell Call Tree.




(b) Example of Phishing Campaign Control Panel.

Figure 4.15: Phishing Control Panel and Backdoor Call Tree.

streamlines the process of managing stolen data and maintaining the phishing site, making it easier for attackers to operate their campaigns efficiently.

To further understand the components of these backdoors and web shells, Table 4.14 provides a comprehensive list of keywords commonly associated with backdoors and web shells found in phishing kits. The table is divided into two main categories: "Shell" and "System." The "Shell" category lists various names and types of web shells that attackers might use to maintain unauthorized access to compromised web servers. These include well-known shells like "c99," "r57," and "b374k," and region-specific shells such as "syrianshell" and "indishell."

The "System" category enumerates PHP functions and commands often used in malicious scripts to execute system-level operations. These include functions for executing shell commands (*e.g.*"shell_exec," "exec"), file manipulation ("fopen," "mkdir"), and potentially harmful operations like "eval" that can execute arbitrary PHP code. This table is a valuable reference for security professionals and researchers who want to identify potential backdoors in web applications and phishing kits.

Table 4.14: Web Shell & System Access Keywords in Kits.

| Type | Keyword Value |
|---|---|
| Shell | “angel”; “b374k”; “bv7binary”; “c99”; “c100”; “r57”; “webroot”; “kacak”; “symlink”; “h4cker”; “webadmin”; “gazashell”; “locus7shell”; “syrianshell”; “injection”; “cyberwarrior”; “erneby-pass”; “g6shell”; “pouyaserver”; “saudishell”; “simattacker”; “sosyeteshell”; “tryagshell”; “uploadshell”; “wsoshell”; “weevely”; “zehir4shell”; “lostdcshell”; “commandshell”; “mailershell”; “cwshell”; “iranshell”; “indishell”; “g6sshell”; “sqlshell”; “simshell”; “tryagshell”; “zehirshell”; “unknown”; “k2ll33d”; “b1n4ry”; |
| System | “pcntl”; “assert”; “passthru”; “shell_exec”; “exec”; “base64_decode”; “edoced_46esab”; “eval”; “system”; “proc_open”; “popen”; “curl_exec”; “php_uname”; “tcpflood”; “udpflood”; “curl_multi_exec”; “parse_ini_file”; “gzinflate”; “show_source”; “phpinfo”; “readfile”; “fclose”; “fopen”; “mkdir”; “gzip; “python_exec”; “str_rot13”; “chmod”; |

## 4.7 PHP Code Analysis of a Phishing Kit

```php
<?php
require_once('geoplugin.class.php');
$geoplugin = new geoPlugin();
$geoplugin->locate();
$date = gmdate ("Y-n-d");
$time = gmdate ("H:i:s");
$browser = $_SERVER['HTTP_USER_AGENT'];
$message .= "==============+ LOGS +==============\n";
$message .= "Email:    ".$_POST['email']."\n";
$message .= "Password: ".$_POST['pass']."\n";
$message .= "Referer:  ".$_POST['referer']."\n";
$message .= "Host:     ".$_POST['host']."\n";
$message .= "============== [ip] ==============\n";
$message .= "IP: {$geoplugin->ip}\n";
$message .= "City: {$geoplugin->city}\n";
$message .= "Region: {$geoplugin->region}\n";
$message .= "Country Name: {$geoplugin->countryName}\n";
$message .= "Country Code: {$geoplugin->countryCode}\n";
$message .= "User-Agent: ".$browser."\n";
$message .= "Date Log  : ".$date."\n";
$message .= "Time Log  : ".$time."\n";
$domain = "AUTO Logs";
$subj = "DOPE LOG !";
$from = "From: $domain<west>\n";
mail("[AttackerID]@gmail.com, [AttackerID]@yandex.com",
    $subj,$message,$from,$domain);
header("Location: http://www.aliyun.com/");
?>
```

Listing 4: Example of PHP Used in a Phishing Kit to Exfiltrate Victim Data.

Listing 4 presents a PHP script commonly found in phishing kits designed to capture and exfiltrate victim data. The script begins by utilizing the geoPlugin class to gather geographical information based on the victim's IP address. It then collects various data points, including the current date and time and the user's browser information.

The script constructs a formatted message containing sensitive victim data, including email address, password, referrer, and host details, enabling attackers to assess their phishing campaign's effectiveness. It also appends geographical information obtained from geoPlugin, such as IP address, city, region, country name, and country code, providing attackers with precise location data. The stolen information is then transmitted via PHP's mail() function to hardcoded attacker email addresses, a security flaw that makes the attack potentially traceable. The presence of hardcoded

email addresses indicates poor operational security practices. Finally, after exfiltrating the data, the script redirects victims to a legitimate website (Alibaba Cloud), likely as a deception tactic to minimize suspicion and conceal the attack.

## 4.8 Maliciously Registered Phishing Domains

## 4.9 Dataset Collection

To address our research questions, we collect a dataset comprising phishing URLs (subsection 4.9.1), DNS records using a custom-built crawler (subsection 4.9.2), and registration timestamps of phishing domains to analyze their characteristics and lifespans (subsection 4.9.3).

### 4.9.1 Phishing URL and Domain Collection

As shown in Table 4.15, we first collect 2.3M phishing URLs and their associated 765K distinct domains (1,258 TLDs) across a 39-month period spanning July 2021 to October 2024 from multiple prominent phishing blocklists including APWG (Anti-Phishing Working Group) [102], Malware-filter [119], OpenPhish [207], Phishing-Database [192], phishunt.io [217], PhishStats [213], and PhishTank [216]. These sources have been used to better understand the phishing ecosystem [203, 202, 169, 183, 206, 205]. Particularly, APWG is a global industry association of anti-phishing entities, including banks and financial services companies, Internet service providers, law enforcement agencies, and security vendors. APWG maintains an extensive database of phishing URLs gathered from multiple sources.

### 4.9.2 DNS Record Collection

To answer our research question (**RQ1:** *What are the characteristics of maliciously registered domains, and how can we find maliciously registered domains?*), we develop

Table 4.15: Overview of Our Collected Dataset from July 2021 to October 2024 (39 months). We collect a total of 2.3M phishing URLs and 765K domains.

| Type | # URLs | # Domains | # TLD |
|---|---|---|---|
| APWG [102] | 2,184,835 | 697,237 | 1,203 |
| phishunt.io [217] | 262,755 | 66,743 | 598 |
| PhishStats [213] | 221,331 | 57,299 | 541 |
| OpenPhish [207] | 115,804 | 26,127 | 480 |
| Malware-filter [119] | 76,465 | 24,300 | 470 |
| PhishTank [216] | 5,579 | 1,695 | 195 |
| Phishing.Database [192] | 393 | 236 | 51 |
| Total (Distinct) | 2,294,267 | 765,910 | 1,258 |

– APWG: collected from Jul. 15, 2021 to Oct. 31, 2024.
– Others: collected from May 31, 2024 to Oct. 31, 2024.

a comprehensive DNS data collection system to monitor and analyze how phishing attackers configure and modify their DNS settings across different geographic locations. Our system periodically collects DNS records types of our collected phishing domains (*i.e.*, `A`, `AAAA`, `NS`, `MX`, `TXT`) to provide detailed insights into their behavior.

### DNS Crawler Design

We implement a multi-threaded crawler designed for scalability and reliability, using concurrent processing to efficiently handle thousands of domain queries. The crawler maintains a connection pool for database access and implements file-locking mechanisms to prevent data corruption during parallel operations. The crawler collects detailed DNS information using the `dig` command with comprehensive parameters. This approach enables the recursive collection of DNS records, capturing all possible types. For reliability, our system implements a retry mechanism with exponential backoff, attempting each query up to 5 times before marking it as failed.

Our crawler operates at 30-minute intervals, enabling us to capture both gradual changes and modifications in DNS configurations. This high-frequency polling is crucial for detecting dynamic DNS behaviors that phishing attackers might employ to

evade detection, such as fast-flux DNS or rapid record updates. The system stores DNS responses in a structured JSON format, organized by domain, timestamp, and vantage point.

Our data collection period spanned from June 6, 2024, to October 31, 2024. We gathered URLs from blocklist feeds and extracted their domains and subdomains to perform DNS queries during this period. Using our crawler, we collected a total of 94,798 domains, including subdomains, with 11,932 being unique domains.

#### Vantage Points

Moreover, to detect location-based DNS configurations, we query DNS records from 10 geographically diverse DNS resolvers. These include global providers (Google, Cloudflare, Quad9, OpenDNS) and regional servers across six continents (Brazil, South Africa, UK, Australia, South Korea, US). This distributed approach reveals if phishing domains serve different DNS responses based on geographic location—a technique attackers might use to evade detection or target specific regions.

### 4.9.3 Domain Registration Collection

To investigate the lifecycle of maliciously registered domains (**RQ2:** "*What is the lifecycle of a maliciously registered domain?*"), we collect registration information (including timestamps and registrars) of our collected phishing domains. We first utilize WHOIS and the Registration Data Access Protocol (RDAP) [224] from registrars and registries as WHOIS and RDAP provides basic information, such as registrar names and domain registration/expiration dates.

#### GDPR Restriction

However, due to the European General Data Protection Regulation (GDPR), the registration timestamp and the registrant's information (such as their name, address, and phone number) can be unavailable to the public. To this end, we leverage the

methodology of COMAR [188] to obtain registration timestamps of the domains whose information is hidden. Furthermore, we also utilize DNS Zone files from DNS Coffee [247] and DZDB [113] for more comprehensive registration timing analysis. These services daily collect and archive TLD Zone files from ICANN [153]. The Zone file data provides first-appearance and last-seen timestamps of domains; the last-seen timestamp indicates when a domain has been deregistered and is no longer active. We also utilize passive DNS (pDNS) data through DomainTools' Farsight DNSDB [125]. This dataset includes a first-seen timestamp, indicating the earliest recorded observation of a domain in the passive DNS.

### Our Collected Registration Data

Our dataset includes domain registration timestamps collected from various sources: WHOIS (25,987 domains), RDAP (436,176 domains), CT logs (71,156 domains), DomainTools (27,929 domains), and DNS Coffee (526,867 domains, inclusive of DZDB data). In total, we have 526,954 registration timestamps for unique domains.

In sum, our approach enables us to achieve 76.3% (526,954 out of 690,502) coverage of our collected domains. While relying solely on registration timestamps from WHOIS and RDAP provides 62.4% (431,011 domains) coverage, incorporating additional data sources such as zone files, pDNS data, and CT logs significantly improves the completeness of our timestamp data.

### Def. of Maliciously-registered Domains

Phishing domains can be classified into two categories: maliciously registered domains and compromised domains. A maliciously registered domain is intentionally purchased by an attacker for malicious purposes. In contrast, a compromised domain is a legitimate domain originally used for benign purposes, but attackers exploit vulnerabilities of web servers and inject malicious content (*e.g.*, phishing pages)

into the benign servers. Detecting maliciously registered domains at an early stage is a critical step in preventing phishing attacks effectively.

### Identification of Maliciously-registered Domains

We utilize Tranco 1M domains [242] as a reference to filter out both legitimate domains and web hosting (or website builder) service domains from our collected phishing domains. For example, while '`blogspot.com`' is a legitimate blogging service, attackers may create subdomains, such as '`usps-tracking-service.blogspot.com`' for phishing purposes. After removing the platform-based phishing domains, our list remains 689,492 domains.

Furthermore, we leverage the previous approaches [188, 154, 122, 114, 148] on finding maliciously registered domains. Especially, COMAR [241] demonstrated a 95% accuracy using lexical features and registration timestamps in domains. The method from COMAR includes nine lexical features (*i.e.*, presence of a brand name in the domain name, path part of URL, and misspelled target brand name in the domain name). By merging those features and additional features we discovered, we design our method to detect maliciously registered domains by using the following four steps: (1) brand name in domains, (2) squatted domains, (3) random-looking algorithmic domains, and (4) bulk registered domains. As shown in Table 4.16, each steps are taken after removing the Tranco 1M [242] domains from the total list of domains.

Table 4.16: Maliciously-registered Domain. Each steps are taken after removing the Tranco Top 1M [242] list of domains (total of 689,492).

| **Type** | **# of URLs** | **# of Domains*** |
|---|---|---|
| (1) Brand Name in Domain | 709,694 | 247,699 (35.9%) |
| (2) Squatted Domain | 472,320 | 180,468 (26.2%) |
| (3) Random-looking Domain | 283,366 | 194,099 (28.2%) |
| (4) Bulk-registered Domain | 69,599 | 54,787 (7.9%) |
| Mal. Total† | 1,406,525 | 455,525 (66.1%) |

*: Due to the overlap, total domains are over 100%

**(1) Brand Name in Domain.** The first approach to identifying maliciously registered domains involves detecting brand names within the domain or subdomain (*e.g.*, `usps-security.example.com`, or `www.usps-security-login.com`). To establish a comprehensive baseline, we curate a list of the top 1,000 most targeted brands in our collected datasets (*i.e.*, APWG), covering 97% of the domains in our dataset. Domains or subdomains containing any of these brand names are flagged as part of this category. Our analysis reveals that 33.9% of domains in the dataset incorporate brand names in their domain or subdomain, highlighting the prevalence of this tactic among phishing attackers. Detailed results for maliciously registered domains across all categories are summarized in Table 4.16.

**(2) Squatted Domain.** The second category is one of the most common tactics used by phishing attackers: exploiting squatted domains. These domains incorporate a modified version of a brand name in the domain or subdomain, closely mimicking legitimate brand domains to deceive users. For example, a phishing website targeting `facebook.com` might use a squatted domain such as `faceb{o}ook.com` to trick victims into believing they are accessing an authentic website.

To identify potential squatted domains, we employ the dnstwist tool [128], which generates domain name variations using various squatting techniques and widely used in previous works [241, 230, 143, 168]. We apply this tool to the top 200 most targeted brand names, which account for 90% of the domains in our dataset. This process generates 765,444 possible squatted domains based on techniques such as adding extra characters to the domain name (*e.g.*, `facebook0.com` from `facebook.com`), modifying a single bit in the domain name (*e.g.*, `faaebook.com`), replacing characters with visually similar alternatives (*e.g.*, `faceb0ok.com`, where `o` is replaced with the number, `0`), and adding hyphens or extra prefixes (*e.g.*, `face-book.com` or `dfacebook.com`). Our analysis shows that 26.2% of domains in our dataset are squatted domains.

**(3) Random-looking Algorithmic Domain.** The random appearance of algorithmically generated domains makes them hard to detect [131, 227]. Attackers

exploit this trend by capitalizing on users' tendencies not to scrutinize domain names closely before clicking on links, even when the domain looks suspicious. To find random-looking algorithmic domains, we follow the approach in [227] by matching domains with English word lists. We use [228], a word list containing 108,687 words, to identify domains that include any English words. We apply this process after removing the brand in the domain and squatted domains, leaving a total of 194,099 domains.

As shown in Table 4.16, a significant portion of domains (28.2%) are random-looking algorithmic domains. While such domains may appear suspicious to a human [227], automated detection tools often struggle to classify them as malicious due to their lack of clear patterns or recognizable features.

**(4) Bulk Registration of Domain.** Attackers often register many malicious domains simultaneously through bulk registration to maximize profits with minimal effort [149]. A phishing campaign can involve registering multiple domains at the same time and deploying multiple webpages with different domains. Even if one domain is blocklisted, an attacker can rely on others to continue the attack. Our method to find bulk registered domains includes three conditions that must all be met: registered at the same time, registered through the same registrar, and domain names are similar (using Levenshtein distance [171]).

Bulk-registered domains, often created simultaneously through the same registrar, account for 7.9% of the domains in our dataset. While this percentage represents a smaller subset of the dataset, it carries significant implications.

Notably, `Alibaba Cloud`[116] frequently serves as a registrar for these domains, offering bulk registration services[94]. Furthermore, it actively promotes bulk registrations through discounted pricing [95].This combination of bulk registration functionality and discounted pricing likely lowers the barrier for registering multiple domains, making it an attractive option for attackers. This practice enables attackers to sustain their operations by registering multiple domains in bulk, ensuring that some remain active even after others are detected. Some registrars adopt proactive

measures, such as stricter verification or limits on bulk purchases, which significantly reduce the malicious use of bulk-registered domains.

#### Manual Validation

We randomly select 1,000 domains from our identified maliciously registered domains. Then, we manually validate our method of identifying maliciously registered domains by examining the contents of the phishing domains. Specifically, we utilize historical data from the Wayback Machine [106] to identify domains that either lack historical snapshots or display content designed to mimic legitimate webpages.

Our analysis reveals that 72.3% of the examined domains do not have any historical data in the Wayback Machine. Among the remaining 27.7%, 14.8% domains redirect to error pages, while the remaining 12.9% of domains host malicious content pages.

### 4.9.4 Characteristics of Maliciously registered Domains

We analyze DNS components of maliciously registered domains, including their targeted brands, TLDs, and DNS records, to gain insights into their characteristics.

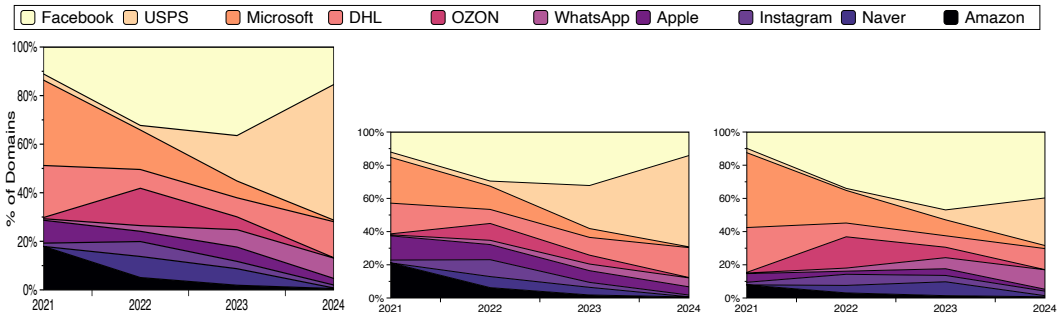
### 4.9.5 Targeted Brand

We utilize target brand information from the APWG dataset. In our analysis, we identify a diverse range of targeted brands, with `Facebook` standing out as the most targeted brand, followed by `USPS`. These two brands alone account for a significant portion with 15.5% (108,391 out of 697,237) of phishing domains, reflecting their widespread recognition and trust among users.

As shown in Figure 4.16, a clear trend emerges among popular targeted brands. Notably, `USPS` is the second most targeted brand, accounting for 6.0% of phishing domains. Interestingly, while `USPS`-targeted domains were minimal in 2021 and 2022, there has been a dramatic increase since 2023. This finding aligns with previous reports on phishing domain trends [154]. Conversely, `Microsoft` shows an overall decline in

targeting, with a more pronounced decrease observed in compromised domains, as illustrated in Figure 4.16(c). Additionally, DHL-targeted domains demonstrate an increasing trend in maliciously registered domains over the years, while showing a decline in compromised domains. We investigate the use of TLDs in phishing domains and assess whether certain TLDs are disproportionately abused. Our analysis considers the varying registration costs across TLDs, which may influence attackers' choices and strategies.

**Motivation.** TLD choice plays a significant role in domain registration for phishing attacks. Attackers may opt for cheaper TLDs to minimize costs or strategically use the same TLD as the targeted brand to enhance impersonation (*e.g.*, using .com for brands that also use .com). According to a phishing report [154], Freenom TLDs were among the most commonly exploited by phishing attackers, as they offered free registrations. However, after reports revealed widespread abuse of this functionality for malicious domain registration, Freenom ceased offering free registrations in early 2023. Moreover, a subsequent report by Interisle [154] noted a shift, with phishing domains in ccTLDs increasing after Freenom's policy change. To examine whether our findings align with this trend, we analyze TLD usage in phishing domains, focusing specifically on maliciously registered domains to uncover patterns and their implications.




(a) Top 10 Brand by Year All. (b) Top 10 Brand by Year (Malicious). (c) Top 10 Brand by Year (Comp.).

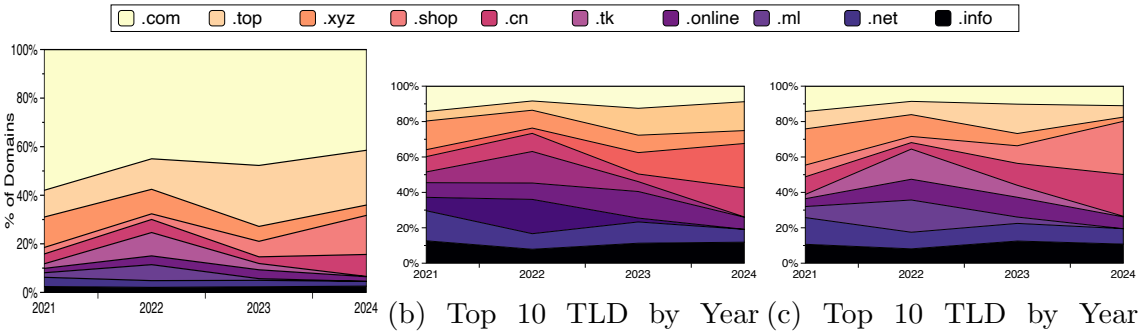
Figure 4.16: Top 10 Brand by Year. USPS increases dramatically from 2022 to 2024, specifically in maliciously registered domains. On the other hand, Microsoft decreases in all domains, DHL increases in maliciously registered domains but decreases in the compromised domains.

**Result: Trend of TLD**

The `.com` TLD dominates the landscape with 218,267 (31.3% out of 697,237) total phishing domains, likely due to its credibility and widespread familiarity, which enhance its effectiveness for deception. Low-cost new gTLDs, such as `.top` and `.shop`, become prominent in our result, with 84,686 (12.1%) and 37,698 (5.4%) domains, respectively, reflecting attackers' preference for inexpensive and lenient TLDs. The lower registration costs of new gTLDs (with prices as low as $1 in our dataset) may contribute to their increased exploitation by phishing domains.

Freenom TLDs (*e.g.*, `.tk`) are heavily exploited in earlier years, but seen a dramatic decline, dropping from 10,931 domains in 2022 to merely 52 domains in 2024, after Freenom discontinued free registrations in 2023. This finding aligns with a previous report [154]. Additionally, as shown in Figure 4.17, the growing presence of `.cn`, from 764 in 2021 to 7,060 in 2024 domains, signals a strategic adaptation by attackers to target TLDs with potentially weaker enforcement mechanisms [154].

As illustrated in Figure 4.17, there is a notable increasing trend in the use of new gTLDs, particularly `.top` and `.shop`. Interestingly, the use of `.top` in maliciously registered domains has steadily increased over the years, while its usage in compromised domains shows a decline in 2024. In contrast, `.shop` demonstrates a consistent increase in usage across both maliciously registered and compromised domains.




(a) Top 10 TLD by Year All. (b) Top 10 TLD by Year (Malicious). (c) Top 10 TLD by Year (Comp.).

Figure 4.17: Top 10 TLD by Year. While `.com` is the most used, `.shop`, `.cn` increase over the years.

#### Using Different TLD than Original Brand Domain

Phishing domains do not always use the same TLD as their original domains. For instance, phishing attackers often register `Facebook`-targeted domains using alternative TLDs such as `.top`, rather than `.com` that used by `Facebook`. Similarly, `USPS`, the second popular targeted brand in our analysis, is frequently targeted using `.top` domains instead of the brand's original `.com`. Another example is `OZON`, ranked as the 5th most targeted brand, with 49.42% of its phishing domains registered under the `.tk` instead of its original `.ru`. Both `.top` and `.tk` are significantly cheaper than `.com` for registration, with `.tk` previously offered for free by Freenom until January 2023. Interestingly, both targeted brands `USPS` and `OZON` have the quickest detected time by blocklists as shown in Figure 4.18. We will discuss detection time across different brands in subsection 4.9.9.

### 4.9.6 Top-level domain (TLD)

Another noteworthy observation is that 44.39% of `OZON`-targeted domains are registered under `.tk`, which is significantly more popular than any other brand. Additionally, `OZON`-targeted domains exhibit the smallest number of unique TLDs (34) among the top 10 brands. Furthermore, as shown in Figure 4.16, `OZON` demonstrates a decline in phishing activity over time. These findings suggest that attackers targeting `OZON` often prefer low-cost TLDs, such as those offered by Freenom, to minimize costs and maximize the scalability of their phishing campaigns.

**Takeaway:** Phishing attackers prefer low-cost TLDs like `.top` and `.tk` to target brands such as `USPS` and `OZON`, with `OZON` relying on `.tk` for 44.39% of its phishing domains. These brands also show the fastest detection times by blocklists, highlighting the importance of monitoring cost-effective TLDs to combat phishing campaigns.

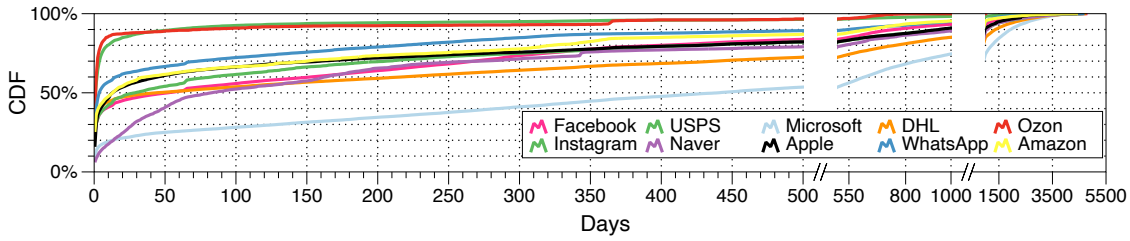



Figure 4.18: Days Between Registration and Detection by Top 10 Brand.

### Maliciously-registered Vs. Compromised

The comparison between maliciously registered and compromised domains reveals notable differences in their TLD preferences. Among a total of 218,267 `.com` domains, 142,479 (65.3%) were maliciously registered, while 75,788 (34.7%) were compromised, indicating that attackers leveraging `.com` domains often register them intentionally for malicious purposes. Conversely, new gTLDs such as `.top` and `.xyz` also show a strong preference for malicious registrations, with 54,014 (63.8%) and 23,085 (61.2%) domains, respectively, highlighting attackers' exploitation of low-cost TLDs for scalability. In contrast, Freenom TLDs like `.tk` saw relatively balanced usage between maliciously registered and compromised domains before policy changes restricted their availability. These patterns suggest that maliciously registered domains favor low-cost or lenient TLDs, while compromised domains may be distributed across a broader range of TLDs, reflecting their opportunistic use of existing infrastructures. This distinction underscores the importance of targeted monitoring and stricter enforcement in TLDs that are disproportionately used for malicious registrations.

We analyze the targeted brands between maliciously registered domains and compromised domains. `Facebook` is the most used targeted brand, with 66,700 phishing domains, of which 58.20% are maliciously registered. `USPS` and `Microsoft` follow, with 41,691 and 26,717 domains, respectively. `USPS` exhibits an exceptionally high proportion of maliciously registered domains (90.03%), indicating that attackers targeting this brand prefer creating new domains rather than compromising existing ones. Microsoft demonstrates a more balanced split, with 51.21% malicious registrations and 46.55%

compromised domains, suggesting a dual approach in leveraging both new and existing infrastructures.

**Takeaway:** New gTLDs (*e.g.*, `.top`, `.xyz`) are more prevalent in maliciously registered domains, while compromised domains favor legacy gTLDs (*e.g.*, `.net`, `.info`). Freenom TLDs like `.tk` and `.ml` have declined, while `.cn` has increased in 2024.

### 4.9.7 DNS Records

We characterize the DNS records of maliciously registered domains collected by our DNS crawler. **DNS Records.** We study the values of commonly used DNS record types (*e.g.*, `A`, `AAAA`, `CNAME`, `NS`, `MX`, and `TXT`). Phishing attackers often configure DNS records to evade detection, frequently altering them using techniques such as fast-flux DNS. Our analysis reveals that 21.4% of domains exhibit record changes, with an average frequency of 79.4 days and a median of 125.2 days.

We study the types of DNS records configured in phishing domains. `NS` records were the most common, with a total of 51,459, followed by `A` records (13,218), `SOA` records (8,960), and `TXT` records (5,573). Focusing specifically on maliciously registered domains, we specifically examined those that exhibited DNS record changes. Our analysis shows that only 4.6% of these domains (117 out of 2,550) demonstrated record changes over time. This suggests that modifying DNS records is not a commonly used tactic among maliciously registered phishing domains.

To understand the scenarios behind DNS record changes, we manually reviewed domains that exhibited such changes over time. One common case involved NS record changes, where domains shifted from one DNS provider to another (*e.g.*, from Cloudflare to Google). Such changes are often motivated by the desire to leverage specific services offered by different DNS providers. For instance, attackers may switch to providers like Cloudflare to utilize features, such as free SSL certificates, which are available for a limited duration [117].

Our analysis reveals that phishing domains show a strong preference for hosting on Amazon Web Services (AWS) infrastructure. Specifically, we extract all IP addresses associated with `A` records and utilize the `Summarize IP` feature provided by IPinfo [156] to gain insights into their hosting characteristics. Among the Autonomous System Numbers (ASNs) analyzed, AS16509 (Amazon.com, Inc.), a primary ASN for AWS, hosts 81.2% of the phishing domains, while an additional 15.4% are hosted on AS14618 (Amazon.com, Inc.), another AWS-associated ASN. Combined, these two ASNs account for 96.6% of all analyzed phishing domains, indicating a significant reliance on AWS services. This preference may be attributed to AWS's scalability, cost-effectiveness, and global reach, which make it an attractive option for attackers to host phishing domains. In comparison, other hosting providers, such as Google LLC (1.5%), JSC Selectel (0.2%), and DigitalOcean, LLC (0.1%), host far fewer phishing domains.

### Vantage Point of DNS Server

Phishing attackers can configure location-aware DNS responses. This allows attackers to deliver localized phishing content (*e.g.*, Spanish-language phishing pages for victims in South America) or to evade detection by serving benign pages when accessed from certain locations commonly used by detection systems.

Our preliminary analysis shows that some phishing domains adapt their content to different languages based on the location of the user accessing them. However, we do not find any evidence that these phishing domains alter their DNS records based on the vantage point of the queried DNS servers. Instead, further investigation reveals that these domains implement language customization through client-side code rather than DNS configuration.

### TTL in DNS Records

In DNS records, the time-to-live (TTL) specifies how long DNS settings are cached before they are automatically refreshed. Typical TTL values are 12 or 24 hours, with

recommended minimum and maximum values of 1 hour (3600 seconds) and 24 hours (86400 seconds), respectively [155]. 2.1% of the domains use TTL values less than 60 seconds, and 25.8% use values shorter than 1 hour (3600 seconds) from our dataset. Only 2.9% of the domains set TTLs longer than 12 hours, and among those, 31 domains set values between 12 and 24 hours. The median TTL value across domains is 3,994 seconds, while the average is significantly higher at 60,827 seconds. The use of short-lived TTLs can facilitate fast-flux DNS, a technique that frequently changes IP addresses to evade detection [111, 132] and often employ by attackers [121].

**Takeaway:** Our analysis finds that 21.4% of domains change their DNS records frequently. 2.1% of phishing domains configure their DNS TTL values to less than 60 seconds, a configuration commonly associated with fast-flux DNS techniques.

### 4.9.8 Lifespan of Maliciously Registered Phishing Domains

This section examines the lifecycle of phishing domains, focusing on two critical phases: (1) the time from registration to detection, (2) the time from detection to deactivation, and (3) the comparison of detection time between blocklists. These phases provide insights into how phishing attackers sustain their domains to maximize monetization and evade timely countermeasures. By analyzing detection delays and post-detection persistence across different domain types, brands, and registration strategies, we uncover characteristics in the lifespan of phishing domains (*i.e.*, maliciously registered). Our findings highlight the need for improved detection mechanisms to reduce delays and more robust enforcement measures to ensure rapid domain takedown, thereby limiting attackers' ability to exploit these domains.

### 4.9.9 Time Taken between Registration to Detection (Detection Delay)

In this section, we analyze how phishing domains are detected by blocklist after registration.

#### Motivation

Maliciously registered domains can be blocked in advance when compared to compromised domains. Phishing domains exhibit significant variation in the time it takes to be detected after registration, influenced by the type of domain and the targeted brand. As shown in Figure 4.19, these differences highlight both quicker detection for some brands and prolonged delays for others.

#### Result: Overview of Detection Delay

Across all domains, the overall median detection time is 42.4 days, with an average of 286.2 days. For the top 10 most targeted brands, the detection times have a slight improvement over these values, with an average of 286.2 days and a significantly shorter median of 11.7 days. This suggests that well-known brands tend to benefit from quicker median detection times compared to less prominent brands, likely due to more active monitoring and stronger anti-phishing measures.

#### Detection Time between Targeted Brands

`USPS` (United States Postal Service) [245], a U.S. federal agency providing postal services, and `OZON` [209], a Russian e-commerce platform founded in 1998, stand out with the fastest average detection times among targeted brands. `USPS` has a median of 1.4 days (average of 59 days), and `OZON` has a median of 1.3 days (average of 42.9 days). These quicker detections may result from more active monitoring systems or simpler phishing tactics that are easier to identify.

Both brands are targeted using non-original TLDs (subsection 4.9.6), which are often cheaper to register. Also, the detection as shown in Figure 4.18, detection time of `USPS` and `OZON` is quickest with a median of 1.4 days and 1.3 days respectively. Some registrars, such as Freenom, provide APIs for the immediate takedown of phishing domains upon detecting signs of abuse [91]. This suggests that attackers' choice of cost-effective TLDs may have inadvertently backfired, as these domains could be removed quickly.

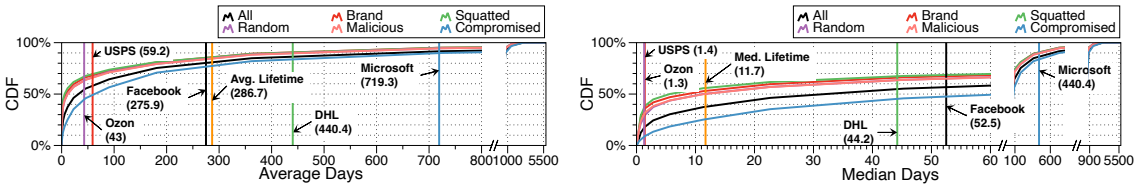
Domains targeting `Microsoft` take the longest to be detected, with an average detection time of 719 days (median of 440.4 days). `Facebook`, despite being the most impersonated brand, has a moderate detection time of 275 days (median of 52.5 days).

These findings highlight significant disparities in detection efficiency across brands and TLDs, emphasizing the impact of attackers' TLD choices on detection timelines.

### Maliciously-registered Vs. Compromised

As shown in Figure 4.19, the detection times vary across different categories of maliciously registered domains. Overall, compromised domains consistently have slower detection times compared to maliciously registered domains, though the difference is not substantial. Specifically, the median detection time for maliciously registered domains is 16.3 days, with an average of 206.4 days, while compromised domains have a median detection time of 86 days and an average of 332.1 days. This indicates that current detection methods do not perform significantly better at identifying maliciously registered domains compared to compromised domains. We will discuss potential future directions.

**Takeaway:** Detection delays for phishing domains vary, with maliciously registered domains detected faster (median 16.3 days) than compromised ones (median 86 days). Brands like `USPS` and `OZON` see rapid detection (medians of 1.4 and 1.3 days), while others, like `Microsoft`, face significant delays (median 440.4 days and 52.5 days, respectively).




(a) Average Delays (days) Between Registration and Detection.

(b) Median Delays (days) Between Registration and Detection.

Figure 4.19: Delays (days) Between Registration and Detection. Vertical bars show average (or median) days between registration time and detection time of the top 5 most targeted brands.

### 4.9.10 Time Taken between Registration and Deregistration (Takedown Delay)

Figure 4.20 highlights the significant variation in the time it takes for phishing domains to be deregistered after detection. Across all phishing domains, the average time between detection and deregistration is 11.5 days on average, reflecting a relatively short-lived post-detection activity. However, specific domain categories reveal notable discrepancies. Squatted domains persist significantly longer, with an average lifespan of 23 days post-detection. Random-looking domains and impersonating specific branded domains exhibit average post-detection duration of 14.8 days and 19.9 days, respectively.

This prolonged availability of squatted and brand-targeted domains underscores their continued risk in phishing campaigns, as these domains remain accessible to victims even after being blocklisted. The observed differences in deregistration times between maliciously registered domain categories, such as brand-targeted (19.9 days), random-looking (14.8 days), and squatted domains (23 days), may reflect variations in the policies or practices of registrars and hosting providers. These differences could also indicate that attackers exploit specific domain types for their perceived resilience or due to differences in enforcement or takedown mechanisms. These findings reveal critical gaps in enforcement mechanisms, particularly for squatted domains, which outlast other categories by a wide margin.

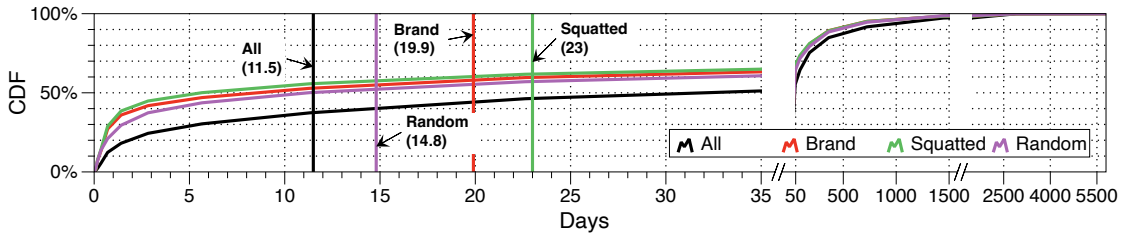



Figure 4.20: Days Between Detection and Last Seen. Vertical bars show the last seen timestamp from the zone file by each type of maliciously registered domain.

**Takeaway:** Maliciously registered domains, especially squatted domains, are key in phishing domains but are deregistered more slowly, averaging 23 days compared to 11.5 days for all phishing domains.

### 4.9.11 Comparison Between Blocklists

As shown in Figure 4.21, detection times vary significantly between APWG and other blocklists, illustrating how quickly each blocklist identifies phishing domains after they have already been detected by APWG. APWG plays a critical role in identifying phishing domains, with domains on its blocklist having an average detection time of 277.3 days and a median detection time of 42.4 days.

In contrast, other blocklists show considerable delays in detecting these same domains. For instance, `Phishunt.io` has an average detection delay of 676.1 days and a median of 930.8 days after APWG's detection, indicating significant lag. Conversely, blocklists like `PhishTank` and `OpenPhish` demonstrate faster detection times, with `PhishTank` averaging 167.7 days and a median of 4.4 days, while `OpenPhish` averages 257.9 days and a median of 4.1 days after APWG detection. `Malware-filter` and `PhishStats` also detect domains relatively quickly, with median delays of 3.7 and 2.3 days, respectively, despite higher average delays of 255.6 and 141.9 days.

`Phishing.Database` shows mixed results, with an average detection delay of 388.5 days but a stronger median delay of 64.4 days. These findings demonstrate that APWG consistently detects phishing domains earlier than all other blocklists in our

dataset. However, the variability in detection delays across blocklists highlights the need for improved synchronization and data sharing to reduce detection gaps and enhance phishing defense coverage. APWG's early detection could be further leveraged to accelerate response times across the ecosystem.

**Takeaway:** APWG consistently detects phishing domains earlier than other blocklists, but significant variability in detection delays across blocklists underscores the need for improved synchronization and data sharing to enhance timely phishing defense and reduce attacker impact.

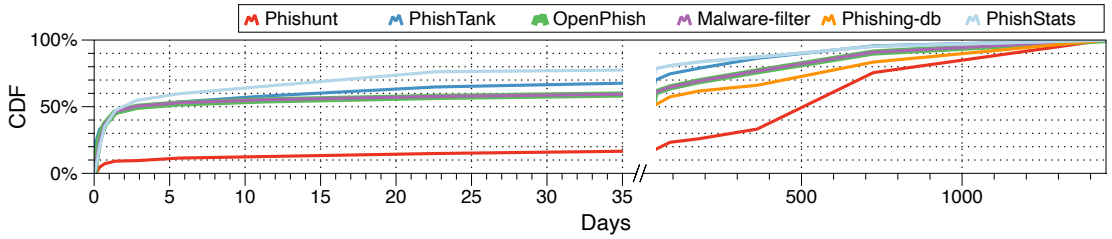



Figure 4.21: Days Between APWG Detection and Other Blocklists. Other than Phishunt, all 5 blocklists show similar median delays (2.3 to 4.4 days, except Phishunt).

# Chapter 5

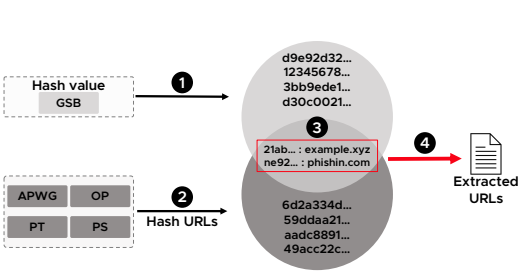
# Defensive Strategies and User Behavior

## 5.1 Effectiveness of Blocklist in Phishing Defense

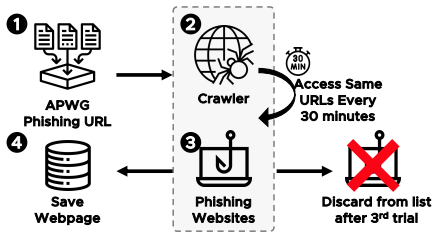
### 5.1.1 Data Collection

Recall that our primary objective is to assess the effectiveness of Google Safe Browsing (GSB) in terms of (1) maintenance behavior, (2) detection delay, and (3) detection failure. To achieve this, we collect phishing URLs from GSB and other sources.

**Phishing URLs.** To assess GSB's detection delay compared with other phishing intelligence services, we collect timestamps of phishing URLs when they are added to GSB and other phishing intelligence services (APWG, OP, PT, and PS) and compare them. In this experiment, the main challenge arises from the fact that GSB only stores hashed phishing URLs without the plaintext URLs. To address this issue, we convert all phishing URLs reported by APWG, OP, PT, and PS to hash values and then compare these converted hash values with GSB's hash values, as illustrated in Figure 5.1(a).




(a) **Workflow of URL Comparison with GSB.** ❶ Collect full hash values from GSB. ❷ Hash URLs from non-hashed phishing intelligence services (APWG, OP, PT, and PS). ❸ Find common URLs from hashed URLs. ❹ Analyze common URLs.




(b) **APWG Crawler.** ❶ Collect APWG Phishing URL. ❷ Use Crawler to access Phishing Websites. ❸ Access websites every 30 minutes. Discard if the website is inaccessible after three trials. ❹ Save Phishing Webpage resources (*e.g.*, HTML, Screenshot, and Redirected URLs).

Figure 5.1: Overview of Our Data Collection Procedure.

**Timestamps.** To analyze GSB's maintenance practice, we utilize GSB's URL hashes and their timestamps. Investigating the GSB's hashes and timestamps allows us to identify notable patterns in its maintenance behavior. Specifically, we use our APWG Crawler as shown in Figure 5.1(b) to check if the websites have any status changes (*e.g.*, inaccessibility, changes in the appearance of the website). This data from APWG Crawler is also used in analyzing GSB's detection failure in subsection 5.1.12, aiming to compare and identify potential reasons for GSB's detection failures.

### 5.1.2 Collecting Phishing URLs

We collect phishing URLs from GSB and other phishing intelligence services such as APWG [169], OpenPhish (OP) [208], PhishTank (PT) [215], and PhishStats (PS) [212], widely used in the research community and publicly available [203, 206, 202, 126, 210, 169].

#### Google Safe Browsing (GSB)

Using Google Safe Browsing Update APIs v4 [137], which facilitates the clients to download and sync the local database with GSB's blocklist database [135], we collect

a total of 6,907,047 blocklisted phishing URLs from GSB during our collection period, from May 2 to December 31, 2023 (243 days).

### Phishing Intelligence Service

We leverage four popular phishing intelligence services to collect phishing URLs to find common ground with the GSB.

- *Anti-Phishing Working Group (APWG)* [100] is a global industry association of anti-phishing entities including financial institutions, government agencies, and tech companies [103]. The APWG collects phishing URLs from diverse sources and disseminates them to better understand phishing attacks and to improve mitigation against phishing attacks.
- *OpenPhish (OP)* [208] is a phishing intelligence service that has a detection engine, designed to identify active phishing URLs and extract threat intelligence [208].
- *PhishStats (PS)* [212] is a phishing intelligence service that provides phishing URLs to help improve security [212]. PS is used by various vendors such as Antiphishing.la, CTM360, Miteru, SpiderFoot, StalkPhish, and the operator of the '.XYZ' domain.
- *PhishTank (PT)* [215] is a community resource that provides phishing URLs [215]. Yahoo Mail, McAfee, Carnegie Mellon, and Opera web browsers use this blocklist service against phishing attacks [214].

### Phishing Intelligence Service Usage

We utilize the APWG's eCrime Exchange dataset to amass a comprehensive collection of phishing URLs, marking one of the largest compilations of such data [100]. In total, we collect 1,427,822 distinct phishing URLs (129,895 distinct domains) during the same collection period as the GSB collection—from May 2, 2023, to December 31, 2023. Furthermore, to gain a more comprehensive dataset of phishing URLs, we collect a total of 260,548 URLs from OP, PT, and PS during the same collection period; 196,464 from OP, 151,137 from PT, and 10,210 from PS as shown in Table 5.1.

Table 5.1: **Distinct Number of URLs and Domains**.

| Blocklisted | GSB | APWG | OP | PT | PS | Total |
|---|---|---|---|---|---|---|
| URLs | 6,907,047 | 1,427,822 | 196,464 | 151,137 | 10,210 | 8,432,132 |
| Domains | N/A* | 129,895 | 41,552 | 36,103 | 1,244 | 167,035 |

*: As the URLs in GSB are hashed, we are unable to identify the domains of the URLs and count the number of domains.

### 5.1.3 Collecting Screenshots and APWG Crawler

To answer RQ2 and RQ3, additional datasets such as screenshots of phishing URLs and client-side resources (*i.e.*, HTML, JavaScript) of phishing websites are required. In the following subsections, we describe how we collect those additional datasets.

#### APWG Crawler

Our study investigates the factors influencing GSB's behavior, such as criteria for adding and removing URLs. We design a crawler that checks phishing websites every 30 minutes for changes in content (*i.e.*, HTML, JavaScript, screenshots) or activity status. APWG phishing URLs are monitored at 30-minute intervals, and if a site is inaccessible after three consecutive attempts, the crawl is stopped to optimize efficiency. From August 9 to December 31, 2023, our crawler collected 136,971 URLs. The overall pipeline of this process is shown in Figure 5.1(b).

**Screenshots of Phishing Website.** To answer RQ3, we analyze HTML content and screenshots from the APWG dataset to identify evasion techniques used by phishing websites, comparing URLs detected by GSB with those it misses. Using the APWG eCrime Exchange (eCX) [100] as a trusted source, we examine GSB's response to evolving phishing websites. Our crawler updates every 30 minutes, collecting web resources like HTML, JavaScript, CSS, and screenshots. This dataset, consisting of 1,427,822 URLs collected from May to December 2023, enables a detailed comparison of GSB's detection effectiveness, discussed in subsection 5.1.12.

### 5.1.4 How Properly Is GSB Maintained?

Proper maintenance of GSB, including timely addition and removal of phishing URLs, is crucial for effective phishing prevention. Our analysis reveals that GSB is the only service that removes URLs, making us measure the removal activities of the GSB.

### 5.1.5 GSB's Dynamic URL Management.

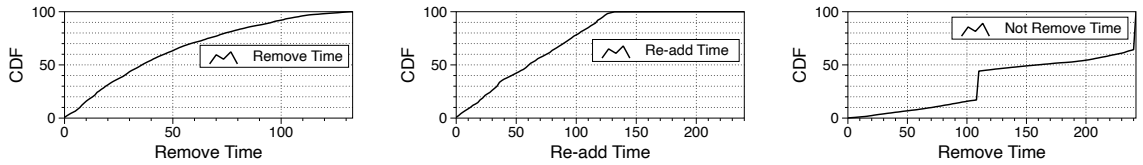
According to [141], GSB removes URLs that are false positives or are no longer a threat. During our collection, GSB added 6.9 million URLs and removed 3.1 million. The median removal time is 36 days, with 98.1% of URLs removed within 120 days, as shown in Figure 5.2(a).

**Reappeared URLs and Duration Analysis.** GSB removes URLs and then reappears after being identified as threats again. As shown in Figure 5.2(b), the median reappearance time is 62 days, with 93.3% reappearing within 120 days and 81 URLs reappearing after 200 days. In particular, 10,599 URLs reappear within a single day, indicating that this behavior occurs frequently and on a large scale. Unfortunately, such a frequent and quick reappearance behavior might not be desirable if phishing attacks were active when they were removed. From the users' perspective, keeping such URLs longer would increase the protection.

**Takeaway:** In GSB, 6.9 million URLs were added and 3.1 million were removed, with removals occurring within a median of 36 days. The median time for URLs to reappear is 62 days, yet 10,599 URLs reappear within a day, indicating that the reappearance behavior is frequent and prevalent while not desirable for the protection.

### 5.1.6 Re-addition Frequencies and Implications

We identify patterns of URL addition and removal in our dataset, referred to as the Addition & Removal (A&R) flow, shown in Table 5.2. Most phishing sites experience




(a) Remove URLs CDF (b) Re-add URLs CDF (c) Not Removed URLs CDF

Figure 5.2: **CDF of GSB's Remove, Reappear, Not Removed URLs.**Each graph shows the number of URLs removed, reappeared, and not removed by time domain. Over 90% of removal and reappearance happen within 120 days. 33.2% URLs were never removed in our collection period (8 months).

continuous state changes, with 23.2% following an $A \rightarrow A$ flow, where URLs are re-added without prior removal. Our manual inspection of 47 such URLs reveals that content changes, such as shifting target brands or web component positions, trigger re-addition. As shown in Figure 5.2(b), an average interval of 107.1 days between reappearances, seems excessively long, given that most phishing sites last only a few days. This suggests a potential gap in GSB's daily monitoring claims [141].

**Takeaway:** In GSB, 23.2% of URLs are being re-added without prior removal due to content changes. The average interval between re-additions is 107.1 days, deviating from GSB's daily monitoring policy.

### 5.1.7 Not Removed URLs Analysis

As shown in Table 5.2, the most common pattern is a single addition, accounting for 33.2% of URLs. Our examination reveals that certain URLs persist on GSB for up to 243 days as shown in Figure 5.2(c), due to abandonment or being single-use phishing sites. Notably, 78% of single-addition URLs (3.8 million) remain listed for over 80 days, indicating non-reuse, while URLs with multiple additions are typically removed within 20 days. During the study, 55.8% of URLs were not removed, and a sample analysis (with 1,000 sample URLs) revealed that 95% of these URLs were no longer accessible. This suggests that GSB may not promptly remove URLs.

Table 5.2: **Number of A&R Flow in GSB.** *A* represents *Addition* and *R* represents *Remove*. Most of the URLs (33.2%) are added, but not removed, for at most eight months. An unmanageable number of URLs are not removed but continue to be added (*e.g.*, $A \rightarrow A \rightarrow ...$ ).

| # of A&R flow | # of URLs | # of A&R flow | # of URLs |
|---|---|---|---|
| $A$ | 2,213,100 (33.2%) | $A \rightarrow \mathrm{R} \rightarrow \mathrm{A} \rightarrow \mathrm{R}$ | 263,363 (3.9%) |
| $A \rightarrow \mathrm{A}$ | 1,548,991 (23.2%) | $A \rightarrow \mathrm{A} \rightarrow \mathrm{A}$ | 46,839 (0.7%) |
| $A \rightarrow \mathrm{A} \rightarrow \mathrm{R}$ | 647,854 (9.7%) | $A \rightarrow \mathrm{A} \rightarrow \mathrm{A} \rightarrow \mathrm{A}$ | 16,270 (0.2%) |
| $A \rightarrow \mathrm{R} \rightarrow \mathrm{A}$ | 628,790 (9.4%) | $A \rightarrow \mathrm{A} \rightarrow \mathrm{A} \rightarrow \mathrm{R}$ | 10,461 (0.2%) |
| $A \rightarrow \mathrm{R}$ | 509,043 (7.6%) | $A \rightarrow \mathrm{A} \rightarrow \mathrm{A} \rightarrow \mathrm{A} \rightarrow \mathrm{A}$ | 9,747 (0.1%) |

**Takeaway:** 33.2% of URLs persist on the GSB without being removed for up to 243 days, which shows insufficient management. Notably, 95% of the sampled URLs that remain on GSB are no longer accessible.

### 5.1.8 How Effective Is GSB in Detecting Phishing?

This section compares detection delays between GSB and four phishing intelligence services (APWG, OP, PT, and PS). We analyze how quickly each service detects and adds new phishing websites to their databases, assessing the relative effectiveness of each approach in responding to phishing threats.

### 5.1.9 GSB Detection Coverage

We first aim to evaluate GSB coverage of phishing URLs by comparing it with other phishing intelligence services. To conduct this comparison, we first need to hash the URLs from the other services following GSB's method [138]. We canonicalize 1.7 million unique phishing URLs, standardizing them by lowercasing, removing default ports, and normalizing domains and protocols, resulting in 2,298,789 canonicalized URLs*. These URLs are then converted into SHA256-hash values to match GSB's

*GSB's canonicalization and standardizing step make a maximum of 30 hash prefixes.

format. We then compare the hash values with GSB's dataset to assess its detection coverage, as shown in Figure 5.1(a).

#### Result

Our analysis reveals significant differences in URL detection between GSB and other phishing intelligence services. In our dataset, GSB lists 6,907,047 URLs, while the APWG lists 1,427,822. Of these, only 181,053 URLs overlap between GSB and APWG, representing just 2.6% of GSB's URLs. When including three additional phishing intelligence services (OP, PT, and PS), we find 254,048 URLs common between them and GSB, accounting for 3.7% of GSB's listings. Notably, GSB does not include around 1.5 million URLs (89.9%) reported by the four phishing intelligence services. We further investigate the technical details of these phishing URLs in subsection 5.1.12 to better understand the GSB's effectiveness.

**Takeaway:** Existing phishing intelligence services cover 254,048 phishing URLs while GSB does not contain (or misses) over 1.5M URLs that are detected by other phishing intelligence services.

### 5.1.10 Detection Delay Comparison

In this analysis, we aim to understand how quickly each platform identifies phishing URLs compared with other platforms. In particular, we focus on the 254,048 URLs that exist in all blocklist platforms.

#### The Phishing URL Detection Window

To effectively prevent victims from visiting phishing websites, GSB needs to detect the phishing website as early as possible. Previous work [206] discovers a typical timeline of a phishing website as follows. After a phishing website is up for the attack, it takes **9 hours and 42 minutes** on average ($t_{detect}$) to get detected by one of the phishing intelligence services. After that, 90% of the victims are measured to visit a phishing

website within **8 hours 15 minutes** ($t_{90\%}$) on average (within 17 hours 57 minutes from the launch of the phishing website; $t_{detect} + t_{90\%}$). Lastly, [206] measured that the last victim visits the phishing websites **12 hours and 30 minutes** ($t_{100\%}$) after the website is detected by the first phishing intelligence service (22 hours and 12 minutes after the launching; $t_{detect} + t_{100\%}$).

To this end, we analyze the differences in the detection timelines of GSB and other phishing intelligence services to infer how effective GSB is in terms of detection speed when compared to other phishing intelligence services.

Since the earliest event we can observe from the phishing intelligence service datasets is when a phishing website is detected and added to a dataset, we consider the time of the website first appeared in the datasets ($t_{detect}$) as the starting point for each phishing website. Then we compute the detection delay of GSB or other phishing intelligence services from the starting point. From Table 5.3, we count the number of URLs detected in GSB after other phishing intelligence services have been detected. Specifically, our analysis focuses on three critical time points from the starting point which is directly taken from [206] (*i.e.*, the time of the first appearance in a dataset; $t_{detect}$): 45 minutes ($t_{75\%}$), 8 hours 15 minutes ($t_{90\%}$), and 12 hours and 30 minutes ($t_{100\%}$). As each time point indicates the deadline for 75% of victim visits, 90% of victim visits, and last victim visits (*i.e.*, 100% of victim visits) according to [206], we measure how GSB and each phishing intelligence service promptly detects before those points.

**Result.**

As shown in Table 5.3, GSB exhibits a delay in detecting 88.8% of the URLs compared to APWG. More importantly, many of them experience significant delays. Only 0.3% of the URLs were detected within the first 45 minutes ($t_{75\%}$) and 11.1% within 8 hours and 15 minutes ($t_{90\%}$). Most are detected after the 90% deadline ($t_{90\%}$), meaning that over 90% of the victims are already exploited when phishing websites' URLs are added. Worse, 47.2% of the URLs are added after the 100% deadline ($t_{100\%}$), meaning that it

is practically ineffective as almost all victims have already been exploited. Similarly, for PhishStats, 71.3% of the URLs are detected by GSB after PS. However, in this comparison, no URLs are detected within 12 hours and 30 minutes ($t_{100\%}$), indicating a critical gap, with GSB failing to prevent any phishing URLs. On the other hand, compared to OpenPhish, 28.9% of URLs are detected after OP. However, 48.1% of URLs are detected within 12 hours and 30 minutes ($t_{100\%}$). Similarly, PhishTank also showed significant detection delays, only 23.3% of URLs are detected within 12 hours 30 minutes ($t_{100\%}$).

**Takeaway:** Our findings indicate that GSB exhibited significant detection delays, identifying 88.8% of URLs after APWG, with only 11.1% detected within the 90% deadline. A significant number of phishing websites are added to GSB after the 100% deadline (47.2%, 51.9%, 76.7%, and 100% in APWG, OP, PT, and PS, respectively), suggesting GSB might be ineffective in practice as most URLs are added too late, according to [206].

Table 5.3: **Compare Detection Delay in Pairs.** The "Delay in PI" column shows the number of URLs detected later by other services compared to GSB, while the "Delay in GSB" column shows delays in GSB detection. The "75% Deadline" indicates URLs detected within 45 minutes, the time by which 75% of victims typically visit the phishing site. The "90% Deadline" refers to URLs detected within 8 hours and 15 minutes, and the "100% Deadline" shows URLs detected within 12 hours and 30 minutes, when all victims have typically visited the site.

| | | | GSB's Capture Delays | | | |
|---|---|---|---|---|---|---|
| **PI**[1] | **Delay in PI**[1] | **Delay in GSB** | **75% Deadline**[2] | **90% Deadline**[3] | **100% Deadline**[4] | **Beyond Deadline**[5] |
| APWG | 20,268 (11.2%) | 160,785 (88.8%) | 421 (0.3%) | 17,873 (11.1%) | 66,586 (41.4%) | 75,905 (47.2%) |
| OpenPhish | 33,364 (71.1%) | 13,572 (28.9%) | 1,297 (9.6%) | 3,963 (29.2%) | 1,266 (9.3%) | 7,046 (51.9%) |
| PhishTank | 28,600 (89.1%) | 3,502 (10.9%) | 115 (3.3%) | 615 (17.6%) | 85 (2.4%) | 2,687 (76.7%) |
| PhishStats | 131 (28.7%) | 326 (71.3%) | 0 (0%) | 0 (0%) | 0 (0%) | 326 (100%) |

[1] = Phishing Intelligence Services, [2] = Within 45 Minutes, [3] = Within 8 Hours 15 Minutes,[4] = Within 12 Hours 30 Minutes, [5] = After 12 Hours 30 Minutes

### 5.1.11 Detection Order Analysis

After observing that GSB and phishing intelligence services add newly detected phishing websites at a different pace, we dig deeper by looking at the overall detection time differences of common URLs in GSB and other phishing intelligence services. In particular, we focus on their relative orders and time differences among the services.

**Comparison of GSB and Phishing Intelligence Services.** To understand which service detects phishing websites first and how long it would take for other services to catch up, we aim to conduct relative detection orders among the phishing intelligence services. To conduct the comparison analysis, we first unify the phishing URLs by merging the datasets from all phishing intelligence services. Then, each service's order is computed from the time of the phishing URL's first detection. Note that in this analysis, we do not include PhishStats (PS) as most of the URLs in PS do not appear in others, making the analysis significantly biased if we include it in the analysis (potentially reducing the dataset by 94.4%).

To this end, we end up with 6,560 URLs. Note that although the overlap of URLs between GSB and other phishing intelligence services is limited, our study seeks to learn the larger operational trends and assess the detection efficiency across these entities.

#### Detection Time Order

We analyze phishing detection times for 6,560 URLs across GSB, APWG, OpenPhish, and PhishTank. Figure 5.3 illustrates the overall detection order for GSB and other services. APWG detected 5,311 URLs (81.0%) first, while GSB, OpenPhish, and PhishTank detected 1,076 (16.4%), 148 (2.3%), and 25 (0.4%) first, respectively. The second column shows GSB detecting 3,211 URLs after another service, followed by OpenPhish (2,359) and APWG (820). PhishTank typically detected URLs the latest, with 178 URLs flagged last as shown in Table 5.3. These results indicate that APWG consistently detects phishing URLs the earliest, while PhishTank, due to its

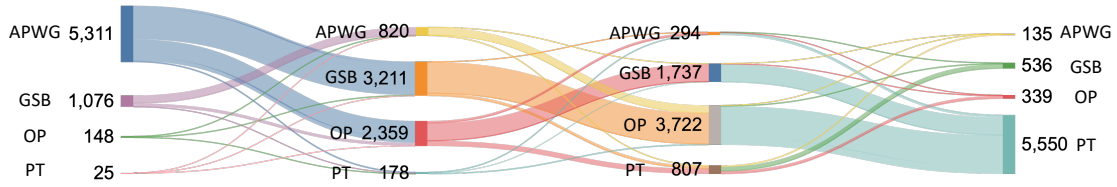



Figure 5.3: **Sankey Diagram of Common Blocklists.** The diagram illustrates the tracking of a URL (*e.g.*, "example.xyz" is first reported by APWG and then by GSB, OP, and PT, sequentially). The line thickness indicates the number of URLs that each service detects at each stage.

community-driven process, often detects them the slowest. GSB and OpenPhish show similar detection timing, suggesting comparable performance.

**Takeaway:** APWG consistently detects phishing URLs earlier than GSB, as evidenced by an analysis of URLs common across five services, positioning APWG as the most effective detector in this comparative study.

### 5.1.12 Weakness of GSB in Phishing Detection

We observe that GSB has limited overlap with other phishing intelligence services, potentially missing phishing websites identified by others. To understand technical details of the non-overlapping websites, we analyze phishing websites detected exclusively by each service, examining evasion techniques, redirection, domain squatting, CAPTCHA use, and text languages, and conducting both quantitative and qualitative assessments.

### 5.1.13 Experimental Setup

#### Dataset Preparation

A key challenge in analyzing the GSB dataset is that it provides phishing URLs in a hashed format. To address this, we use the APWG dataset to explore instances where GSB may fall short in protecting users from phishing attacks. We first compute the

overlap of phishing URLs between APWG and GSB, and identify URLs unique to APWG. As shown in Table 5.3, the detection results between GSB and APWG are comparable. To further investigate, we analyze additional APWG data (HTML files, screenshots, and redirection URLs) while minimizing false positives by excluding error pages. This pre-processing involves steps like filtering out non-200 HTTP errors and blank pages, as described in Figure 5.1(b). After this process, we identify 129,616 phishing websites that overlap between the two datasets and 471,540 that appear only in APWG.

### Categorization

Prior work [85, 203] attributes the ineffectiveness of phishing blocklists to adversaries using various evasion techniques. We build on these insights in our analysis by categorizing phishing websites based on evasion methods, including Cloaking, Redirection, and CAPTCHA.

### Cloaking

A phishing website is considered to use cloaking if its screenshot shows a blank page or if its HTML contains error messages [203]. To detect this, we analyze HTML files from our crawler for error messages. Additionally, we train a ResNet50 model [151], pre-trained on ImageNet, to classify screenshots with or without errors. Using a subset of 412 samples (206 error and 206 non-error screenshots), we fine-tune the model and apply it to the full dataset to identify phishing websites displaying errors.

### Redirection

Phishing websites may evade detection by redirecting crawlers to benign sites. To detect this, we compare the initial URLs accessed by our crawler with the final landing URLs to identify phishing websites using redirection techniques.

#### Domain squatting

Phishing websites may use domain squatting as an evasion technique. To detect this, we use the DNS fuzzing tool `dnstwist` [124]. If a domain from our dataset matches one generated by dnstwist, we classify it as domain squatting. We exclude phishing websites using web-hosting domains (*e.g.*, blogger.com, wix.com) for more accurate categorization.

#### Language

Our preliminary analysis reveals detection discrepancies between the APWG and GSB datasets, influenced by the languages used on phishing websites. To assess the impact of language representation on detection performance, we analyze the effectiveness of language detection using `fastText` [162, 161], a model from Facebook AI Research Lab. This helps highlight the potential effect of unbalanced language samples in phishing detection.

### 5.1.14 Quantitative Analysis

We first analyze quantitatively to determine if there are differences in detection performance between GSB and APWG.

#### Methodology

We compare the number of phishing websites in each category and perform a $\chi^2$ test to assess statistical significance. Since each category contains only two values (websites unique to APWG or detected by both), we use a resampling approach. We randomly select subsets of 10,000 phishing websites, repeating this process 1,000 times to create a distribution of evasion techniques. Finally, we calculate the $p$-value to evaluate statistical significance (Table 5.4).

Table 5.4: **Ratio of Phishing Websites Detected by GSB in APWG Corpus.** We use a crawled dataset by APWG and categorize them per the evasion technique. "Not in GSB" counts the phishing websites detected only by APWG, and "In GSB" shows those detected by both APWG and GSB.

| **Evasion Technique** | **Category** | **Analysis Result** | | |
|---|---|---|---|---|
| | | **Not in GSB** | **In GSB** | ***p*-value** |
| Cloaking | Using cloaking | 20,542 (4.4%) | 2,452 (1.9%) | 0.89 |
| Redirection | Redirection link | 241,746 (51.3%) | 87,180 (67.3%) | 1.00 |
| | Path changed | 57,101 (12.1%) | 4,267 (3.3%) | 1.00 |
| | Domain changed | 147,739 (31.3%) | 75,571 (58.3%) | 0.99 |
| | Subdomain changed | 33,348 (7.1%) | 2,464 (1.9%) | 0.99 |
| | Redirect to benign | 9,388 (2.0%) | 796 (0.6%) | 0.65 |
| Squatting | Use squatted domain | 294,543 (62.5%) | 80,857 (62.4%) | 1.00 |
| CAPTCHA | Using CAPTCHA | 28,191 (6.0%) | 6,459 (5.0%) | 0.42 |
| Language | Not English | 241,761 (51.3%) | 78,787 (60.8%) | 1.00 |
| | Non Latin-based[1] | 14,392 (3.1%) | 4,320 (3.3%) | 0.99 |
| **Total** | - | 471,540 | 129,616 | - |

[1] Non-Latin based languages *e.g.*, Chinese, Japanese, Korean, and Russian

**Results.**

Table 5.4 highlights key evasion techniques observed in our analysis comparing GSB and APWG detection performance. Using resampling and statistical tests, we find that GSB detects only 10.7% of 22,995 employs `cloaking`, with APWG covering the majority. For phishing sites using `redirection`, GSB identifies 87,180, while APWG detects 241,749. In cases of `redirection` to benign domains, GSB captures only 796 sites. Both services show similar performance in detecting domain squatting, with GSB at 62.4% and APWG at 62.5%. `CAPTCHA` use is rare, with 6,459 sites in GSB and 6% of APWG's dataset. Additionally, 60.8% of GSB-detected phishing sites use non-English languages, and 3.3% use non-Latin alphabets.

**Takeaway:** Our analysis shows that when phishing websites use evasion techniques or redirect to benign sites, GSB's detection rate is neither particularly strong nor weak compared to other blocklists. However, GSB captures varying numbers of phishing websites across different evasion categories.

### 5.1.15 Qualitative Analysis

In subsection 5.1.14, we observe variations in the phishing websites GSB detects and further analyze these categories.

**Methodology.**

In addition to our manual analysis, we employ `fastdup` [175], an algorithm designed for clustering images. We run `fastdup` on website screenshots from both the datasets and then manually inspect the distinctive characteristics or differences between the clusters.

### Redirection

We analyze the distribution of target brands used for redirection, as shown in Figure 5.4. In both datasets, `USPS` is the most popular brand in phishing attacks, followed by `Bing`, `Facebook`, `YouTube`, and `Naver`. A key difference is `Google Play`, which appears in APWG but not in GSB. Additionally, we examine phishing websites using both redirection and cloaking. Among GSB's detected sites, only 490 (0.1%) show domain-level changes out of 75,571 phishing websites, compared to APWG's detection of 6,169. GSB detects 831 websites with path-level changes, while APWG identifies 7,600.

### Domain squatting.

We analyze the distribution of target brands used in domain squatting, as shown in Table 5.5. We observe that GSB and APWG include a relatively small number of phishing URLs using domain squatting: 32 out of 80,857 in GSB and 161 out of 294,543 in APWG. Regarding popular brands' URLs used for domain squatting, we find that APWG exclusively includes domain-squatted phishing URLs of YouTube compared to GSB.

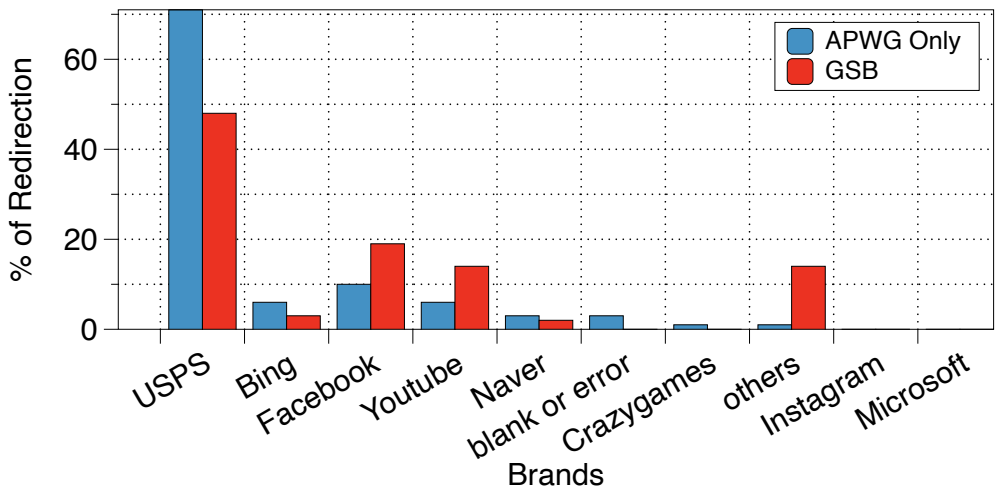



Figure 5.4: Contrasting target brands involved in the redirection that GSB detects vs. those it does not.

Table 5.5: Popular brands used for domain squatting. Domain is the number of websites that use domain squatting of each brand. JSON is the detected brand shown in the JSON file that the crawler collects.

| Brand Name | Phishing Attack | | | |
|---|---|---|---|---|
| | Not in GSB | | In GSB | |
| | Domain | JSON | Domain | JSON |
| Google | 91 | 161 | 17 | 37 |
| Facebook | 33 | 167,131 | 9 | 61,757 |
| Instagram | 17 | 4,260 | 5 | 3,580 |
| Youtube | 6 | 0 | 0 | 0 |
| Microsoft | 5 | 16,664 | 1 | 987 |
| USPS | 3 | 22,606 | 0 | 974 |
| WhatsApp | 3 | 9,758 | 0 | 563 |
| Amazon | 1 | 433 | 0 | 72 |
| Airbnb | 1 | 47 | 0 | 5 |
| AMEX* | 1 | 45 | 0 | 16 |

*AMEX: American Express

**CAPTCHA.** We analyze the types of CAPTCHA challenges used in phishing attacks, summarized in Table 5.6. GSB successfully includes websites using reCAPTCHA and hCAPTCHA but we did not find websites with other CAPTCHA types. While GSB contains no phishing websites with CAPTCHA challenges, APWG has 38 websites. Furthermore, GSB lacks websites that request both CAPTCHA and other credentials, whereas APWG contains 27 such cases.

**Language.** We further analyze phishing websites that use a non-Latin-based alphabet. Our findings indicate that 91.2% (3,940 out of 4,320) of the websites detected by GSB resemble Facebook login pages. This implies that GSB's detection mechanism may have a bias towards certain websites particularly using English as a language.

Table 5.6: CAPTCHA challenges employed in phishing.

| Description | Phishing attacks | |
|---|---|---|
| | Not in GSB | In GSB |
| reCAPTCHA | 396 (1.4%) | 86 (1.3%) |
| hCAPTCHA | 356 (1.3%) | 123 (1.9%) |
| CAPTCHA (banking) | 38 (0.1%) | 0 (0.0%) |
| reCAPTCHA with layout | 27 (0.1%) | 0 (0.0%) |

## 5.2 Lifespan of Phishing Attack

### 5.2.1 Data Collection

#### Phishing URL Source

To address our research questions, we leverage the APWG `eCX` platform [81], one of the widely used repositories in previous research [204, 202, 206, 203, 258, 260, 183, 169, 177]. APWG `eCX` aggregates phishing URLs reported from a wide range of sources, including security vendors, financial institutions, and Internet Service Providers (ISPs). It provides real-time updates on active and reliable phishing websites. Note that since APWG `eCX` provides only metadata (*e.g.*phishing URLs and target brands), we develop a custom crawler that periodically monitors phishing websites and assesses their operational status (*e.g.*take down or still alive). Our use of APWG eCX data minimizes potential bias through scale and source diversity. In our dataset, we observe consistent patterns across different domains (average lifetime of 1.76 hours for logistics and 8.68 hours for social media) and attack types, suggesting a representative sampling.

### Web Crawler Design

Figure 5.5 illustrates our data collection system. At its core, it uses a Redis-based queueing system to manage input URLs. Then, we implement a custom web crawler using Puppeteer [140] and Chromium, augmented with stealth plugins [109] to bypass potential anti-bot measures employed in sophisticated phishing websites. The system is capable of processing approximately 250 URLs per minute with 16 parallel browsing instances. It runs on a 30-minute cycle, systematically checking the operational status of phishing websites by collecting their HTTP status codes (*e.g.*HTTP 404 error). If a website returns an error code three consecutive times or is confirmed offline, we stop visiting it in the next cycle.

Our crawler collects the following comprehensive data from each visited site to facilitate in-depth analysis and tracking of phishing infrastructure and techniques: (1) network information such as IP addresses and WHOIS data, (2) full-page screenshots (with a 1280x960 viewport) and their color histogram, (3) the complete HTML contents (including all dynamically generated elements), (4) the HTTP data (including request headers, content policy, referrer information, and HTTP version), and (5) all redirected links (the entire URL chain from the original APWG link to the landing page).

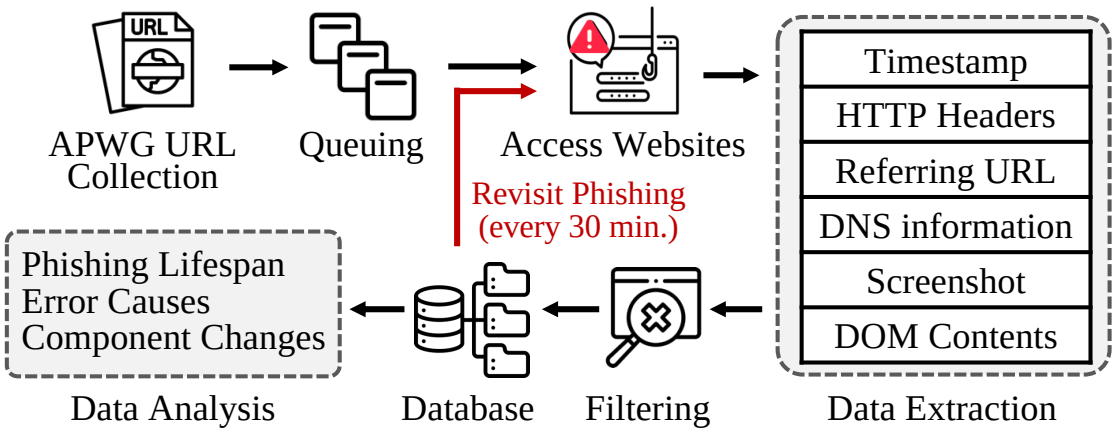



Figure 5.5: Overview of Data Collection.

### Error Handling and Resilience

To ensure data integrity and completeness, we carefully handle various errors. In case of failure, each URL is given up to three attempts, and failures are categorized using a comprehensive error classification system. Specifically, as each attempt is made every 30 minutes, if three attempts fail in a row (*i.e.*unable to connect for 90 minutes), the URL is excluded from the data for the next visits. We use the three consecutive failures (for one and half hours) to tolerate those accidental failures and do not consider them reappearing cases. The three consecutive failure threshold balances capturing temporary outages versus permanent takedowns. Our analysis shows that 42% of sites experience at least one failure. We observe that 3.2% (9,164 URLs) of sites are reappearing cases (*i.e.*becoming responsive again after three consecutive failures).

### URLs Blocklisted by Google Safe Browsing (GSB)

GSB [136] is one of the most popular blocklist-based anti-phishing systems, integrated into many modern web browsers such as Google Chrome, Firefox, and Apple Safari [104, 218, 197]. We leverage GSB Update APIs v4 [137] to collect GSB's blocked phishing URLs. GSB stores the URLs into SHA-256 hashes rather than in plain text [137]. We collect 10,249,563 URLs from GSB during the same collection period of URLs.

### Our Collected Dataset Overview

Our crawler processes 286,237 unique phishing URLs from the APWG feed system and a total of 2,742,542 (2.7M+) visits. This indicates an average of 9.58 visits per URL ($\sigma = 45.2$), reflecting our multiple crawling attempts and the dynamic nature of phishing sites. The substantial number of feed URLs and the high volume of collected traffic underscores the pervasive nature of phishing attempts during our study period. The high average of URL traffic per site suggests that many phishing websites remain active for extended periods (287.4 minutes or about 4.79 hours), with 38,911 sites (25% of the total) receiving more than 18 visits, indicating prolonged activity.

### 5.2.2 Post-detection Lifespan of Phishing (RQ1)

To answer our first research question "**How long do phishing websites remain alive even after detected, and how does this vary across different targeted brands?**", we conduct a series of analyses: (1) calculating the overall post-detection lifespan of phishing sites across different brands, (2) determining the detection times by GSB, (3) comparing the lifespans and detection times across various targeted brands, and (4) identifying patterns and factors influencing phishing site longevity.

From this analysis, we aim to obtain insights regarding the persistence of phishing sites and the effectiveness of detection mechanisms across different brand targets.

### 5.2.3 Overall Post-detection Lifespan of Phishing

#### Top Target Brands

Facebook (including Meta) is the most targeted brand in our dataset, accounting for 27.08% of phishing URLs (77,525 out of 286,237). Unlike past studies, USPS ranks second, despite never appearing in previous top 10 lists [183, 204, 258, 169], underscoring the evolving nature of phishing trends and the need for monitoring.

#### Phishing Website Lifespan

As shown in Table 5.7, the average lifespan of phishing websites in our dataset is 54.04 hours, but the median is just 5.46 hours, indicating a highly skewed distribution as shown in Figure 5.6. While 50% of sites are taken down quickly, some persist much longer, such as a Microsoft-themed phishing site lasting 38.43 days.

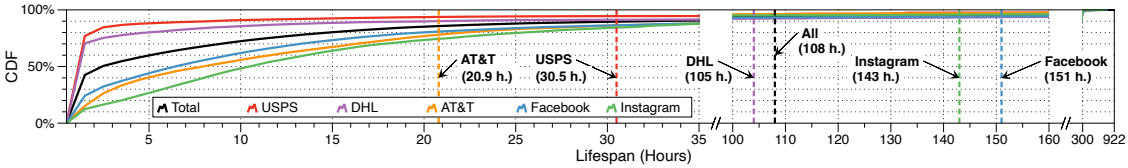



Figure 5.6: Lifespan CDF with Top 5 brands. Vertical bars indicate detection time by Google Safe Browsing.

Table 5.7: Lifespan of Top 10 Phishing Brands.

| Brands | # URL (%) | Avg. Lifespan | Med. | Max. | Std. |
|---|---|---|---|---|---|
| Facebook | 77,525 (27.08) | 56.57 h. (2.36 d.) | 8.68 h. | 33.31 d. | 4.71 d. |
| USPS | 32,089 (11.21) | 44.08 h. (1.83 d.) | 1.76 h. | 31.15 d. | 4.43 d. |
| AT&T | 9,811 (3.42) | 41.15 h. (1.71 d.) | 9.37 h. | 31.30 d. | 3.70 d. |
| WhatsApp | 7,261 (2.53) | 41.47 h. (1.73 d.) | 5.80 h. | 31.38 d. | 4.18 d. |
| Instagram | 4,746 (1.66) | 54.12 h. (2.55 d.) | 12.62 h. | 31.01 d. | 4.43 d. |
| DHL | 3,633 (1.27) | 61.95 h. (2.58 d.) | 1.89 h. | 31.26 d. | 5.82 d. |
| SwissPass | 1,912 (0.67) | 66.57 h. (2.77 d.) | 5.89 h. | 30.05 d. | 5.17 d. |
| Evri | 1,104 (0.39) | 73.85 h. (3.08 d.) | 1.71 h. | 30.22 d. | 6.33 d. |
| Rakuten | 604 (0.21) | 66.29 h. (2.76 d.) | 4.73 h. | 29.61 d. | 4.67 d. |
| Google | 582 (0.20) | 49.26 h. (2.05 d.) | 6.11 h. | 26.95 d. | 3.97 d. |
| **Total** | 286,237 (100) | 54.04 h. (2.25 d.) | 5.46 h. | 38.43 d. | 4.81 d. |

∗ **Facebook** includes Meta; **Total** includes 1,654 brands.
† **h.** indicates 'hours'; **d.** indicates 'days.'

### Lifespan by Top Target Brand

Phishing website lifespans vary by brand, averaging 41.15 hours (AT&T) to 73.85 hours (Evri). Logistics brands like DHL, USPS, and Evri have shorter median lifespans (1.71–1.89 hours) compared to Facebook (8.68 hours), suggesting a fast, aggressive attack strategy. Mann-Whitney U tests (p ¡ 0.001) highlight significant differences, particularly between USPS and Instagram, while Facebook and Instagram exhibit distinct patterns in phishing site persistence as shown in Table 5.8.

### Analysis of Outliers (Longer-lived Phishing)

As shown in section 5.5, Longer-lived phishing sites vary by brand, with Facebook leading at 23%, followed by USPS (9.52%). DHL-themed sites, though rare (1.08%), persist the longest, averaging 122.45 hours and peaking at 750.26 hours. Instagram's persistent cases have a high median lifespan (32.22 hours), while AT&T shows 2.90% of long-lived sites but the shortest average duration (78.8 hours).

Table 5.8: Mann-Whitney U Test for Longer-lived Phishing.

| Comparison | U Statistic | $p$-value | Sample Sizes | Significant |
|---|---|---|---|---|
| USPS vs DHL | 8,889,555.5 | 2.20e-78 | 14733, 1674 | Yes |
| USPS vs Facebook | 148,623,111.5 | 0.00e+00 | 14733, 37454 | Yes |
| USPS vs AT&T | 20,392,662.0 | 2.51e-271 | 14733, 4278 | Yes |
| USPS vs Instagram | 9,014,440.0 | 1.40e-274 | 14733, 2270 | Yes |
| DHL vs Facebook | 22,780,213.0 | 4.33e-80 | 1674, 37454 | Yes |
| DHL vs AT&T | 3,052,938.5 | 8.43e-19 | 1674, 4278 | Yes |
| DHL vs Instagram | 1,344,337.5 | 1.09e-55 | 1674, 2270 | Yes |
| Facebook vs AT&T | 92,660,283.5 | 2.17e-63 | 37454, 4278 | Yes |
| Facebook vs Instagram | 38,870,796.0 | 6.88e-12 | 37454, 2270 | Yes |
| AT&T vs Instagram | 3,487,045.5 | 7.86e-79 | 4278, 2270 | Yes |

$*$ All $p$-values are Bonferroni corrected; Significance level: $\alpha = 0.05$.

**Takeaway 1:** The average lifespan of phishing websites is 54.04 hours (2.25 days), with significant variations across brands (41.15 to 73.85 hours). While most sites are taken down quickly (median 5.46 hours), a concerning subset persists for extended periods. Logistic companies tend to operate within a compressed time frame than other sectors. Our analysis identifies 49 unique brands with at least one phishing site lasting 30 days or more. We find significant differences in phishing lifetime across all brand pairs and variability in blocking effectiveness across brands.

### 5.2.4 Effectiveness of Google Safe Browsing

We scrutinize phishing lifespan related to GSB [66] through the following analyses (see Table 5.9): ❶ the number of phishing attempts detected and undetected by GSB, ❷ GSB detection time differences between typical and redirected phishing URLs, ❸ point-in-time cases detected by GSB (based on APWG data and phishing site activity status), and ❹ phishing attempts that exceeded the average GSB detection time.

**GSB Detection Rate.** From our comparison, GSB lacks access to APWG data and independently detects 18.41% (52,696) of phishing URLs, leaving 81.59%

Table 5.9: Phishing Site Lifespans and GSB Detection Time.

| Category | Average | | Median | |
|---|---|---|---|---|
| ❶ ***Lifespan Analysis*** | | | | |
| GSB Detected (18.41%) | 55.89h | (2.33d) | 7.86h | (0.33d) |
| GSB Not Detected (81.59%) | 35.96h | (1.50d) | 2.67h | (0.11d) |
| ❷ ***GSB Detection Times*** | | | | |
| Typical URLs (11.44%) | 68.27h | (2.84d) | 4.89h | (0.20d) |
| Redirected URLs (6.97%) | 130.89h | (5.45d) | 5.73h | (0.24d) |
| Total URLs (18.41%) | 108.73h | (4.53d) | 5.80h | (0.24d) |
| ❸ ***Detection Scenarios*** | | | | |
| GSB detection before APWG | 375.11h | (15.63d) | 28.62h | (1.19d) |
| GSB detection after APWG | 166.69h | (6.95d) | 0.81h | (0.03d) |
| APWG & GSB detect active phishing | 223.58h | (9.32d) | 20.51h | (0.85d) |
| GSB detection after phishing is down | 404.19h | (16.84d) | 23.45h | (0.98d) |
| ❹ ***Long-tail Analysis (>4.5 days)*** | | | | |
| GSB Detected | 274.93h | (11.46d) | 236.85h | (9.87d) |
| GSB Undetected | 273.58h | (11.40d) | 260.69h | (10.86d) |

unlisted and exposing users to threats. GSB prioritizes high-impact sites, detecting those with longer lifespans (55.89 vs. 35.96 hours) and sophisticated infrastructure. While 11.44% (32,745) of detected URLs are direct phishing links, 6.97% involve redirection, highlighting GSB's limitations against this evasion technique. However, GSB outperforms seven other major blocklists (*e.g.*malware-filter [68], OpenPhish [69], Phishing Army [70], Phishing Database [172], Phishunt [73], PhishStats [71], and PhishTank [72]), as detailed in section 5.6) in detecting redirected phishing sites, balancing accuracy and minimizing false positives, as 83.93% of detections occur after sites become inactive. This may be due to Google not being a member of APWG [101] and lacks access to its data. Additionally, prior studies [211] show that blocklists do not share URL lists.

We analyze GSB's detection of phishing campaigns using intensive redirection, identifying 1,994 groups with 9,581 unique source URLs. GSB detects only 189 groups,

with 29 including the destination URLs, highlighting its limitations in handling complex redirection schemes.

### Lifespan Analysis with GSB

Our ❶ lifespan analysis reveals significant differences between phishing sites detected by GSB and the others not detected. GSB successfully detects 18.41% of the phishing sites in our dataset, which have an average lifespan of 55.89 hours (2.33 days) and a median lifespan of 7.86 hours (0.33 days). In contrast, the majority of phishing sites (81.59%) not detected by GSB have shorter lifespans, averaging 35.96 hours (1.50 days) with a median of 2.67 hours (0.11 days). This suggests that while GSB detects only a small portion of phishing sites, the detected ones tend to have longer lifespans than undetected ones.

Phishing sites using redirection evade detection longer, with a mean lifespan of 5.5 days vs. 4.5 days for non-redirected sites, though median lifespans are similar (5.7 vs. 5.8 hours). Statistical analysis confirms redirected URLs are 8.24 times less likely to be detected by GSB ($p < 0.001$, FET [130]), suggesting redirection aids phishing campaigns.

GSB-detected phishing sites vary in lifespan by brand, with DHL-targeted sites lasting the longest (61.98 hours) and AT&T-focused sites averaging 41.29 hours. USPS sites show extreme skew, with a 1.76-hour median lifespan but a 30.5-hour GSB detection time. GSB's average detection time (108.73 hours) lags behind 83.93% of site takedowns, while the remaining 16.07% persist for an average of 172.81 hours post-detection, ranging from 142.35 hours (AT&T) to 202.18 hours (DHL).

### GSB Detection Time

GSB detects phishing sites in an average of 108.73 hours (4.53 days), with a median of 5.80 hours, often after sites are already taken down. Compared to the average phishing site lifespan (54.04 hours), GSB's detection is slow, particularly for sites using evasion

techniques like redirection. Our analysis (❷ GSB detection time) shows that typical phishing URLs are detected in 68.27 hours (median: 4.89 hours), while redirected URLs take significantly longer – 130.89 hours (median: 5.73 hours), highlighting a considerable exposure period for potential victims.

### GSB Detection Time by Target Brand

As illustrated in Figure 5.6, GSB's detection speed varies by brand. AT&T-themed phishing sites are detected fastest (20.9 hours), while Facebook-themed sites take the longest (151 hours). Other brands fall in between, with USPS at 30.5 hours, DHL at 105 hours, and Instagram at 143 hours.

### Detection Scenarios

Our analysis of ❸ GSB detection scenarios in relation to APWG identification and site takedown reveals three distinct scenarios: (1) GSB detects before APWG, (2) GSB detects after APWG, and (3) GSB detects after phishing site takedown. These scenarios show significant variations in phishing site lifespans.

When GSB detects before APWG, sites persist for an average of 375.11 hours (15.63 days). In cases where GSB is detected after APWG, the average lifespan decreases to 166.69 hours (6.95 days). The average time between APWG and GSB detection is 223.58 hours (9.32 days). Most concerning is when GSB detects after-site deactivation, with an average lifespan of 404.19 hours (16.84 days). This scenario indicates a significant real-time detection gap. Early GSB detection does not always ensure shorter lifespans, possibly due to blocking delays or sophisticated evasion techniques employed by phishers.

### Long-tail Analysis of GSB Detection Time

The ❹ long-tail analysis shows an interesting pattern in the persistence of these threats, focusing on phishing sites that remain active for more than the GSB average

detection time of 4.5 days. The 16.44% of phishing sites detected by GSB have an average lifespan of 274.93 hours (11.46 days) and a median of 236.85 hours (9.87 days). In comparison, the 11.01% of phishing sites not detected by GSB have a similar average lifespan of 273.58 hours (11.40 days) but a slightly higher median of 260.69 hours (10.86 days). These results indicate that a significant number of phishing sites persist for long periods, regardless of whether GSB detects them.

**Takeaway 2:** GSB's phishing detection shows limitations, identifying only 18.41% of sites in 4.5 days on average. 83.93% are blocklisted after takedown, while others survive 7.2 more days. Phishing sites using redirection evasion typically have longer detection times. These findings suggest room for improvement in detection speed and coverage.

## 5.3 Take-down Causes of Phishing Attacks (RQ2)

To address our second research question **"What factors cause phishing websites to be taken down?"**, we investigate the dataset to identify potential factors that may cause the takedown of phishing websites. Our analysis reveals three primary categories of errors leading to phishing site takedowns: DNS resolution failure, page not found error, and timeout errors. Each category shows distinct patterns across web-hosting services and self-hosted environments and among different targeted brands.

In particular, we focus on HTTP response error codes and DNS configurations by analyzing error logs generated during accessing each phishing site. For DNS configuration, we use a combination of the public suffix list [74] and reverse DNS to distinguish between web hosting (*e.g.*wix.com) and self-hosted websites.

From the 90,356 phishing domains across the top ten major brands, we find three primary error types: (1) DNS resolution failure errors, (2) page not found errors, and (3) timeout errors. 'DNS resolution failure' error occurs when a domain name cannot be resolved to IP addresses. 'Page not found' error arises when requested resources

do not exist on servers, while 'timeout' errors result when servers take too long to respond. Each of these presents unique patterns across web hosting and self-hosted environments.

### 5.3.1 General Causes of Takedowns

As shown in Table 5.10 and Table 5.11, our analysis of phishing domains shows that DNS resolution failures are the most common takedown cause (67.23%, 60,913 domains), followed by page not found errors (27.19%, 24,573) and timeout errors (5.39%, 4,870).

Error distribution varies by hosting type and target brand. Among web-hosted sites, 'DNS resolution failure' ranges from 35.42% (Google) to 82.95% (Rakuten), while self-hosted failures range from 43.37% (DHL) to 78.46% (Google). 'Page not found' errors also vary, from 9.09% (Rakuten) to 53.09% (AT&T) in web-hosted phishing and 7.11% (Google) to 53.85% (DHL) in self-hosted sites.

#### DNS Resolution Failures

'DNS resolution failures' dominate the observed takedown causes, accounting for 67.23% of all errors. This prevalence suggests that DNS configuration is notable in phishing site lifecycles. The wide range of DNS resolution failure error rates across brands and hosting types (from 35.42% to 82.95%) indicates significant variability in DNS-related takedowns. For web-hosted sites, Rakuten-themed phishing shows the highest DNS resolution failure error rate (82.95%), while Google-themed phishing has the lowest (35.42%).

Table 5.10: Phishing Site (Root Domain) Analysis by Error Type, Hosting Method, and Brands.

| Error Type | Hosting Method | Top 10 Phishing Brand (% of errors) | | | | | | | | | | All Brands |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | FB (Meta) | USPS | AT&T | WhatsApp | DHL | Instagram | SwissPass | Evri | Google | Rakuten | |
| **DNS Resolution** | Web Hosting | 12,025 (60.05%) | 605 (73.33%) | 1,577 (44.11%) | 796 (64.35%) | 411 (60.35%) | 1,093 (62.71%) | 719 (61.40%) | 68 (69.39%) | 17 (35.42%) | 73 (82.95%) | 26,765 (66.64%) |
| | Self-Hosted | 5,984 (75.39%) | 8,071 (60.94%) | 282 (61.84%) | 1,516 (71.27%) | 1,098 (43.37%) | 205 (64.26%) | 267 (66.75%) | 613 (61.73%) | 397 (78.46%) | 285 (73.26%) | 34,148 (67.81%) |
| **Page Not Found** | Web Hosting | 7,072 (35.32%) | 168 (20.36%) | 1,898 (53.09%) | 274 (22.15%) | 240 (35.35%) | 555 (31.84%) | 395 (33.73%) | 25 (25.51%) | 25 (52.08%) | 8 (9.09%) | 11,442 (28.34%) |
| | Self-Hosted | 2,566 (32.33%) | 4,150 (31.33%) | 155 (33.99%) | 336 (15.79%) | 1,363 (53.85%) | 100 (31.35%) | 98 (24.50%) | 312 (31.42%) | 36 (7.11%) | 80 (20.57%) | 13,131 (26.05%) |
| **Timeout** | Web Hosting | 881 (4.40%) | 49 (5.94%) | 89 (2.49%) | 164 (13.26%) | 28 (4.11%) | 93 (5.33%) | 51 (4.35%) | 5 (5.10%) | 6 (12.50%) | 7 (7.95%) | 2,148 (5.36%) |
| | Self-Hosted | 334 (4.21%) | 991 (7.48%) | 17 (3.73%) | 273 (12.83%) | 64 (2.53%) | 8 (2.51%) | 33 (8.25%) | 67 (6.75%) | 73 (14.43%) | 22 (5.66%) | 2,722 (5.40%) |
| **Total** | | 28,862 (31.85%) | 14,034 (15.49%) | 4,018 (4.43%) | 3,359 (3.71%) | 3,204 (3.54%) | 2,054 (2.27%) | 1,563 (1.73%) | 1,090 (1.20%) | 554 (0.61%) | 475 (0.52%) | 90,356 (99.73%) |

Table 5.11: Summary of Error Causes and Their Frequency based on FQDN.

| Brand | DNS resolution fail | Page not found | Timeout | Access forbidden | Protocol error | DNS refused | Total |
|---|---|---|---|---|---|---|---|
| All brands | 182,239 (63.67%) | 79,626 (27.82%) | 19,468 (6.80%) | 3,767 (1.32%) | 1,070 (0.37%) | 47 (0.02%) | 286,217 (100.00%) |
| Facebook | 64,933 (83.76%) | 9,375 (12.09%) | 2,302 (2.97%) | 809 (1.04%) | 103 (0.13%) | 1 (0.00%) | 77,523 (100.00%) |
| USPS | 19,374 (60.38%) | 10,142 (31.61%) | 2,089 (6.51%) | 413 (1.29%) | 54 (0.17%) | 17 (0.05%) | 32,089 (100.00%) |
| AT&T | 6,464 (65.88%) | 2,870 (29.25%) | 333 (3.39%) | 115 (1.17%) | 28 (0.29%) | 1 (0.01%) | 9,811 (100.00%) |
| WhatsApp | 5,265 (72.52%) | 951 (13.10%) | 934 (12.86%) | 97 (1.34%) | 13 (0.18%) | 0 (0.00%) | 7,260 (100.00%) |
| Instagram | 3,360 (70.81%) | 1,094 (23.06%) | 245 (5.16%) | 34 (0.72%) | 12 (0.25%) | 0 (0.00%) | 4,745 (100.00%) |

With websites using self-hosting services, Google-themed phishing exhibits a high rate (78.46%), contrasting with DHL-themed phishing at the lower end (43.37%). These variations suggest that DNS-related takedowns differ substantially based on the targeted brand and hosting method.

### Page Not Found Errors

As the second most common takedown cause (27.19%), 'page not found' errors occur at similar rates in web-hosting (28.34%) and self-hosting (26.05%) environments. Among web-hosted sites, AT&T-themed phishing has the highest rate (53.09%), while Rakuten-themed phishing has the lowest (9.09%). For self-hosted sites, DHL-themed phishing leads (53.85%), with Google-themed phishing at the lowest (7.11%), highlighting brand-specific variations.

### Timeout Errors

'Timeout' errors, which are navigation and network timeouts, while less frequent at 5.39%, show consistency between web-hosting (5.36%) and self-hosting service (5.40%) environments. WhatsApp-themed phishing sites stand out with notably higher timeout rates in both web-hosted (13.26%) and self-hosted (12.83%) environments. In contrast, AT&T-themed sites have much lower rates (2.49% web-hosting service, 3.73% self-hosted). These differences indicate that certain brands may be associated with higher rates of timeout errors in phishing attacks.

### 5.3.2 Takedown Causes Analysis

Understanding CDN usage in phishing infrastructures is crucial, as these networks obscure the true origin of traffic, making detection and blocking significantly harder [152, 120]. For phishing websites, CDNs create an obfuscation layer that challenges traditional IP-based blocking mechanisms [246].

#### CDN Usage in Phishing Websites

Our analysis, shown in Table 5.12, reveals that 99.80% of phishing-related IPs use CDNs, rendering IP-based blocking largely ineffective. Despite a 4.13% decrease in total phishing IPs, CDNs remain heavily utilized, with Web Application Firewalls (WAFs) leading at 53.45%, followed by traditional CDNs (33.57%) and cloud services (12.78%). Among providers, Cloudflare dominates WAFs (22.17M IPs, down 1.34%), Google leads traditional CDNs (13.75M IPs, down 8.60%), and AWS is the top cloud provider (5.3M IPs, down 3.63%).

**CDN Usage by Brands.** As shown in Table 5.13, CDN usage varies by brand. WhatsApp (83.44%) and USPS (96.99%) rely heavily on WAFs, while Meta exhibits high cloud usage (40.33%). Rakuten favors WAFs (95.63%, up 24.41%), while AT&T deviates with 94.99% reliance on traditional CDNs.

Table 5.12: CDN Changes and Top Providers based on IP.

| Type | Provider | Typical | Redirected | Change (%) | |
|---|---|---|---|---|---|
| CDN | Google | 15,039,132 | 13,745,800 | -1,293,332 | (-8.60%) |
| | Cloudfront | 128,002 | 130,848 | 2,846 | (+2.22%) |
| | Fastly | 42,311 | 48,535 | 6,224 | (+14.71%) |
| Cloud | AWS | 5,498,797 | 5,299,394 | -199,403 | (-3.63%) |
| | Office365 | 276 | 194 | -82 | (-29.71%) |
| | Oracle | 39 | 25 | -14 | (-35.90%) |
| WAF | Cloudflare | 22,473,347 | 22,171,548 | -301,799 | (-1.34%) |
| | Incapsula | 24 | 25 | 1 | (4.17%) |
| **Total** | | 43,181,928 | 41,396,369 | -1,785,559 | (-4.13%) |

Table 5.13: CDN Usage Analysis for Top Phishing Brands.

| Brand | Total IPs | | Change | CDN Type Usage | | |
|---|---|---|---|---|---|---|
| | Typical | Redirected | | CDN | Cloud | WAF |
| Facebook | 13,823,887 | 12,070,911 | -12.68% | 35.87% | 13.93% | 49.74% |
| USPS | 6,317,945 | 6,357,207 | +0.62% | 2.52% | 0.27% | 96.99% |
| AT&T | 7,849,403 | 7,798,646 | -0.65% | 94.99% | 0.56% | 4.36% |
| WhatsApp | 616,768 | 770,939 | +25.00% | 11.90% | 4.20% | 83.44% |
| Instagram | 977,337 | 851,690 | -12.86% | 36.95% | 14.84% | 47.49% |
| DHL | 342,154 | 348,068 | +1.73% | 9.79% | 3.48% | 86.14% |
| SwissPass | 153,368 | 185,328 | +20.84% | 42.95% | 4.11% | 52.38% |
| Microsoft | 131,772 | 119,623 | -9.22% | 10.51% | 5.37% | 83.97% |
| Rakuten | 66,022 | 82,142 | +24.41% | 3.33% | 0.76% | 95.63% |
| **All Feeds** | 43,263,524 | 41,477,792 | -4.13% | 33.57% | 12.78% | 53.45% |

These trends suggest that phishers adapt CDN choices based on their target brands, potentially mimicking legitimate traffic patterns or exploiting vulnerabilities in brand-associated services. The widespread reliance on CDNs, particularly WAFs, presents a major challenge for phishing mitigation strategies.

## 5.4 Phishing Volatility in the Lifespan (RQ3)

We answer the third research question, **"What technical changing attempt occurs during a phishing website's lifetime, likely serving both detection evasion and site improvement purposes?"**, by analyzing screenshots, DNS Records, HTTP headers, and content of collected phishing site resources to track the volatility of phishing websites.

### 5.4.1 Visual Component Changes (Screenshots)

We analyze how phishing sites' visual representation changes over their lifespan by clustering screenshots based on similarity. Each screenshot is resized, normalized, and processed to create a feature vector representing its visual content. Screenshots with a similarity score above 0.95 are clustered together. Hence, if more than two clusters

are found within each phishing campaign, it may indicate that the phishing website's appearance has changed during the campaign.

### Result

Figure 5.7 provides the statistics on the lifespan of phishing sites that experience visual changes during their operation. The data encompasses 71,665 phishing sites, categorized based on the frequency of changes they experience. The analysis reveals that phishing sites with more frequent visual changes tend to have longer lifespans, as reflected in the median and average values. For instance, phishing sites with fewer changes (1 time) have a median lifespan of 1.93 hours and an average lifespan of 9.95 hours. In contrast, sites that experience more changes (50-99 times) exhibit a significantly longer median lifespan of 38.18 hours and an average of 76.49 hours. The longest-living phishing sites, those with over 100 changes, have a median lifespan of 404.52 hours (about 17 days) and an average of 358.35 hours, highlighting how frequent changes help phishing sites evade detection for extended periods.

The quartile statistics (Q1 and Q3) further emphasize this trend. Sites with fewer changes have lower quartile ranges, such as "1 time," with a Q1 of 0.48 hours and a Q3 of 5.80 hours, indicating that most of these sites are short-lived. However, for phishing sites with over 100 changes, the Q1 jumps to 240.30 hours, and the Q3 reaches 480.75 hours, demonstrating that many of these sites persist for long durations before being detected or taken down. These findings underscore the effectiveness of visual changes in prolonging the operational lifespan of phishing sites (see section 5.10).

**Takeaway 4:** Phishing sites frequently changing their visual appearance tend to have significantly longer lifespans. Sites with over 100 visual changes persist for a median of 17 days, compared to 1.93 hours for sites with only one change. This demonstrates that visual modifications are effective for evading detection and prolonging phishing campaigns.

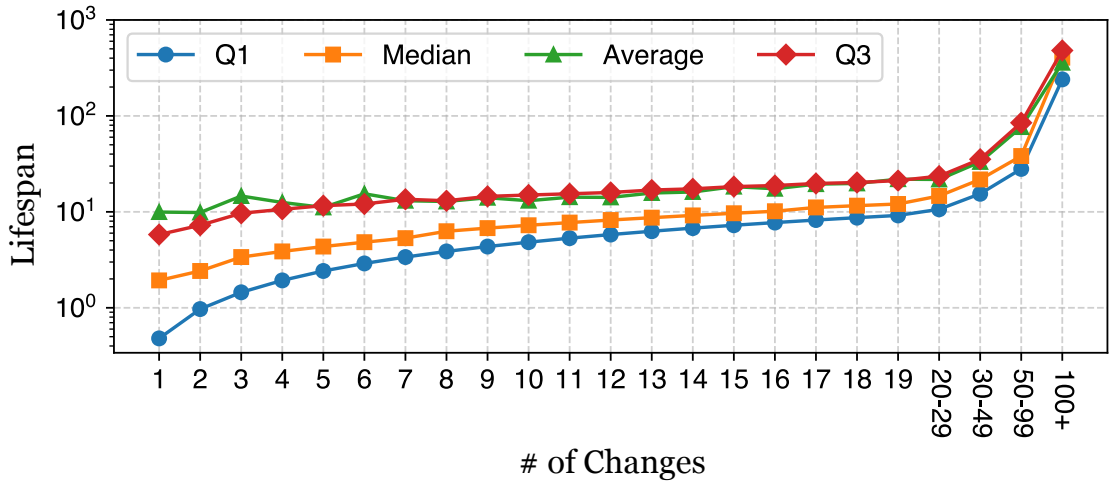



Figure 5.7: Phishing Visual Component Frequency of Changes. The X-axis indicates the # of changes, and the Y-axis indicates the lifespan. It reveals that phishing sites with more visual changes tend to have longer lifespans.

### 5.4.2 DOM Changes from Discovery to Takedown

We analyze changes in website resources between discovery and takedown for 286,237 phishing sites, with 10.5% (30,069) showing final modifications. Our analysis reveals statistically significant changes ($p < 0.001$, t-test) across various metrics. These changes suggest phishing sites evolve their structure and behavior over time, likely to evade detection, improve effectiveness, or adapt to target environments.

#### Structural Changes

Phishing sites simplify their structure between discovery and takedown. The total number of DOM elements decreases by 11.58%, maximum DOM depth by 6.27%, and third-party scripts by 12.37%. This reduction may help sites load faster or appear less suspicious to detection systems. We detect these changes using specific methods tailored to each technique; for example, Canvas fingerprinting is identified through JavaScript API call patterns and shows a 37.09% increase, particularly in phishing sites that survive beyond the median lifespan.

#### Anti-Detection Techniques

Canvas fingerprinting increases by 37.09%, suggesting a shift toward more advanced visitor tracking, while browser plugin detection and screen resolution checks decrease by 23.08% and 26.92%, respectively. We measure JavaScript obfuscation using AST analysis and entropy scoring, observing a 9.30% decrease, while CSS obfuscation increases by 3.73%. These shifts indicate that phishers may be moving obfuscation from JavaScript—frequently scrutinized by security tools—to CSS, which might receive less attention.

#### Dynamic Content & Obfuscation

The use of AJAX decreases by 27.53%, marking the largest decline in this category, while event listeners increase by 6.11%. Additionally, potential Base64 usage drops by 6.69%, while potential Unicode escapes rise by 6.03%. These shifts suggest a move from server-side dynamic content (AJAX) to client-side interactivity (event listeners), making static analysis more difficult. Obfuscation trends also diverge, with JavaScript obfuscation decreasing by 9.30% and CSS obfuscation increasing by 3.73%, as detailed in Table 5.14.

### 5.4.3 DNS Configuration Changes in Phishings

Our analysis of DNS configurations in phishing websites reveals significant variations and changes between the discovery and takedown phases. We examine DNS information associated with phishing websites' IP addresses, performing both forward and reverse DNS lookups to understand how these configurations evolve.

Table 5.14: Comprehensive Changes in Phishing Resources.

| Metric | Discovery | Takedown | Change |
|---|---|---|---|
| ***Site Complexity*** | | | |
| Avg. Total Elements | 211.06 | 186.62 | -24.44 (-11.58%) |
| Avg. Max Depth | 11.17 | 10.47 | -0.70 (-6.27%) |
| Avg. Third-Party Scripts | 2.91 | 2.55 | -0.36 (-12.37%) |
| ***Anti-detection Techniques (%)*** | | | |
| User Agent Checks | 12.39 | 12.43 | +0.04 (+0.32%) |
| Canvas Fingerprinting | 4.53 | 6.21 | +1.68 (+37.09%) |
| WebGL Fingerprinting | 0.13 | 0.11 | -0.02 (-15.38%) |
| Browser Plugin Detection | 0.13 | 0.10 | -0.03 (-23.08%) |
| Screen Resolution Checks | 1.04 | 0.76 | -0.28 (-26.92%) |
| ***Dynamic Content and Obfuscation (%)*** | | | |
| Uses AJAX | 10.79 | 7.82 | -2.97 (-27.53%) |
| Dynamic DOM Manipulation | 26.42 | 25.43 | -0.99 (-3.75%) |
| Uses Event Listeners | 27.19 | 28.85 | +1.66 (+6.11%) |
| Potential Eval Usage | 4.11 | 4.09 | -0.02 (-0.49%) |
| Potential Base64 Usage | 62.31 | 58.14 | -4.17 (-6.69%) |
| Potential Hex Encoding | 12.67 | 12.38 | -0.29 (-2.29%) |
| Potential Unicode Escapes | 14.09 | 14.94 | +0.85 (+6.03%) |
| ***Obfuscation Techniques (%)*** | | | |
| JS Obfuscation | 5.16 | 4.68 | -0.48 (-9.30%) |
| CSS Obfuscation | 12.59 | 13.06 | +0.47 (+3.73%) |

$*$ All changes are statistically significant ($p < 0.001$).

† Percentages in parentheses show relative change.

DNS usage patterns vary considerably across different brands targeted by phishing attacks. For instance, Facebook-themed phishing predominantly relies on `1e100.net` (35.43% at discovery, increasing to 36.18% at takedown) and `github.com` (29.10% at discovery, decreasing to 27.81% at takedown). In contrast, USPS-themed phishing mainly utilizes `cloudfront.net`, with usage increasing slightly from 48.76% at discovery to 49.46% at takedown. We observe fluidity in DNS services across phishing websites. 7.24% of the sites exhibit changes in their DNS services during their lifetime, with 2.61% adding services, 2.73% removing services, and 1.90% switching services. This fluidity suggests active management of DNS configurations by phishers.

We uncover significant modifications in DNS record configurations, with a clear trend towards shorter time-to-live (TTL) durations. Our analysis reveals that 75.84% of phishing websites alter their DNS settings between discovery and takedown. These values span from 0 to 604,800 seconds, with an average of 1,559.87 seconds and a median of 295 seconds. Notably, the majority of these configurations are set to brief intervals: 81.44% at discovery and 81.89% at takedown are below 1800 seconds. Even more striking, 71.90% fall under 600 seconds and 51.10% are less than 300 seconds. This tendency towards shorter durations intensifies over time, with the mean value decreasing from 1,705.63 seconds at discovery to 1,383.54 seconds at takedown. In extreme cases, some are set to 0 seconds, effectively disabling caching. These short-lived DNS configurations can enable fast-flux networks, a technique observed in sophisticated phishing operations [111, 132]. Such changes, particularly using brief and decreasing durations, align with known strategies for evading detection and complicating efforts to track and block malicious infrastructure [121].

### 5.4.4 Phishing Server Transition

We analyze server information from HTTP headers and track changes in server configurations on phishing websites, focusing on the widely used Apache and Nginx servers [250]. Our findings show that 13 websites using Nginx change versions over

time, with 12 upgrading to newer versions and one downgrading. Notably, 2 of the upgraded websites switch to lower minor versions. Similarly, four websites using Apache changed versions, with one downgrading to a lower patch version. Phishing websites often update their server configurations, typically upgrading to newer versions, but occasional downgrades to lower versions suggest potential security weaknesses that could be exploited for detection and mitigation.

**Takeaway 5:** Our analysis reveals that 30,069 phishing sites undergo final changes before takedown, demonstrating significant volatility in their characteristics over time. Notably, while most features decreased, canvas fingerprinting showed a substantial increase of 37.09%, suggesting an evolution in anti-detection techniques. The contrasting trends in obfuscation methods, with JS obfuscation decreasing by 9.30% while CSS obfuscation increasing by 3.73%, indicate a shift in how phishing sites attempt to conceal their behavior.

## 5.5 Comparison of Loger-lived Phishing Brands.

Our analysis of longer-lived phishing sites, defined as those lasting beyond each brand's median lifespan, reveals significant variations across targeted brands (Table 5.15). While less common, DHL-themed sites show remarkable persistence, with the highest mean duration of 122.45 hours (approximately 5.10 days) and a maximum lifespan of 750.26 hours (about 31.26 days). Facebook-themed sites, being more prevalent, exhibit a high mean lifespan of 109.77 hours (about 4.57 days) and the highest recorded maximum of 799.52 hours (approximately 33.31 days). Despite lower overall phishing incidence, Instagram shows a notably high median lifespan of 32.22 hours for its persistent cases. USPS-themed sites present an interesting case. They have the lowest median lifespan (6.95 hours) among longer-lived sites but still maintain a substantial mean duration of 86.72 hours, indicating a skewed distribution

Table 5.15: Comparison of Longer-lived Top 5 Phishing Brands.

| Metric (hours) | USPS | DHL | Facebook | AT&T | Instagram |
|---|---|---|---|---|---|
| Mean | 86.72 | 122.45 | 109.77 | 78.83 | 102.67 |
| Median | 6.95 | 18.59 | 29.92 | 24.74 | 32.22 |
| 90th Percentile | 319.62 | 351.28 | 304.11 | 250.06 | 295.65 |
| 95th Percentile | 353.36 | 535.42 | 350.61 | 324.42 | 341.59 |
| Max | 747.57 | 750.26 | 799.52 | 751.09 | 744.15 |

with persistent outliers. AT&T exhibits the shortest mean lifespan (78.83 hours) among the longer-lasting group.

## 5.6 Result of Anti-phishing Blocklists

As discussed in subsection 5.2.4, GSB detected 18.41% of the phishing URLs in our dataset. However, our analysis of seven other popular anti-phishing blocklists reveals significantly lower detection rates, as shown in Table 5.16. These results underscore the challenges of anti-phishing blocklists in keeping pace with the rapidly evolving phishing landscape. The stark contrast between GSB's performance and other blocklists suggests that GSB may employ more sophisticated detection methods or access a broader range of data sources. The generally lower detection rates for redirect URLs across all blocklists indicate that phishers use redirection techniques to evade detection effectively. This aligns with our findings in section 5.4 regarding the prevalence of redirection in phishing attacks.

## 5.7 Usage of ASNs and TLDs in Phishing Sites

Our analysis of the distribution of Autonomous System Numbers (ASNs) and Top-Level Domains (TLDs) in phishing sites reveals interesting patterns in the infrastructure used by attackers. As shown in Table 5.17, Cloudflare emerges as the dominant ASN, hosting 53.68% of the phishing sites in our dataset. This is followed by a significant

Table 5.16: Comparison of 7 Blocklist Phishing URLs.

| Blocklist | Typical URL | | Redirected URL | |
|---|---|---|---|---|
| | Matches | Percentage | Matches | Percentage |
| malware-filter [68] | 3,600 | 1.26% | 2,400 | 0.84% |
| OpenPhish [69] | 9,525 | 3.33% | 7,163 | 2.50% |
| Phishunt [73] | 761 | 0.27% | 576 | 0.20% |
| Phishing Army [70] | 756 | 0.26% | 424 | 0.15% |
| Phishing Database [172] | 10 | 0.00% | 9 | 0.00% |
| PhishStats [71] | 1 | 0.00% | 0 | 0.00% |
| PhishTank [72] | 0 | 0.00% | 0 | 0.00% |

number of sites (7.68%) with unknown ASNs and then by major cloud providers such as Google (4.55%), Alibaba (4.18%), and Amazon (3.85%). These findings suggest that phishers often leverage popular cloud and CDN services to host their malicious content, potentially benefiting from these platforms' reliability and ability to obfuscate the true origin of the attacks. In terms of TLDs, while the traditional '.com' domain remains the most prevalent (28.11%), we observe a notable use of newer or less common TLDs. The '.shop' TLD, for instance, accounts for 24.87% of the phishing sites, indicating its popularity among attackers. Other frequently used TLDs include '.top' (7.54%), '.dev' (5.83%), and '.io' (3.41%). This diversification in TLD usage may reflect attempts by phishers to evade detection or to create more convincing fake domains that align with their targeted brands or services.

## 5.8 Statistical Results for Longer-lived Phishing

Mann-Whitney U tests [186] compare the lifespan distributions of longer-lived phishing sites across different brands, as shown in Table 5.8. The results reveal statistically significant differences ($p < 0.05$) between all brand pairs, even after applying the conservative Bonferroni correction for multiple comparisons. This indicates that the persistence patterns for phishing attacks vary substantially depending on the targeted brand. The most pronounced differences are observed between USPS and

Table 5.17: Distribution of ASNs and TLDs.

| ASN | Name | Number (%) | TLD | Number (%) |
|---|---|---|---|---|
| 13335 | Cloudflare | 153664 (53.68%) | com | 80455 (28.11%) |
| Unknown | Unknown | 21995 (7.68%) | shop | 71200 (24.87%) |
| 15169 | Google | 13025 (4.55%) | top | 21568 (7.54%) |
| 45102 | Alibaba | 11961 (4.18%) | dev | 16684 (5.83%) |
| 16509 | Amazon | 11009 (3.85%) | io | 9754 (3.41%) |
| 54113 | Fastly | 10520 (3.68%) | org | 7157 (2.50%) |
| 132203 | Tencent | 8008 (2.80%) | app | 7104 (2.48%) |
| 133199 | SonderCloud | 7618 (2.66%) | cfd | 6415 (2.24%) |
| 27323 | ServerStadium | 4109 (1.44%) | cn | 5261 (1.84%) |
| 14061 | DigitalOcean | 3626 (1.27%) | me | 4765 (1.67%) |

Instagram (U = 9,014,440, $p$ = 1.40e-274) and USPS and AT&T (U = 20,392,662, $p$ = 2.51e-271), suggesting markedly different attack persistence strategies for these brands. Interestingly, while Facebook has the largest sample size of a longer-lived span, its comparison with Instagram still shows a highly significant difference ($p$ = 6.88e-12), with the highest p-value among all comparisons. This suggests that while there are statistically significant differences in phishing campaign characteristics between these social media platforms, they may be more similar than to other brands in the study. However, it's important to note that all comparisons showed extremely low p-values ($p$ ¡ 0.05), indicating significant differences across all brand pairs in how long-lived phishing attacks persist.

## 5.9 Error Causes

Our analysis of error causes in accessing phishing sites reveals distinct patterns across brands, as shown in Table 5.11. The most common issue is DNS resolution failure (63.67%), suggesting that many phishing sites become inaccessible due to rapid takedowns or short-lived, disposable domains. The second most frequent error,

"Page not found" (27.82%), likely results from content removal or site restructuring by attackers, possibly as an evasion tactic or due to changes in their infrastructure.

Interestingly, the distribution of errors varies across different targeted brands. For instance, Facebook-related phishing sites show a notably higher rate of DNS resolution failures (83.76%) than the overall average. In contrast, USPS-targeted sites have a higher incidence of "Page not found" errors (31.61%). Timeout errors, while less frequent overall (6.80%), are particularly prominent in WhatsApp-related phishing attempts (12.86%). These variations might reflect differences in anti-phishing strategies employed by various brands or unique characteristics of the phishing campaigns targeting them. The relatively low occurrence of access forbidden errors (1.32%) and protocol errors (0.37%) across all brands suggests that when phishing sites are online, they generally remain accessible.

## 5.10 Visual Component Changes

We analyze visual changes of phishing websites, which include alterations to the website layout. The analysis covers 71,665 URLs, categorized by the frequency of visual changes during their lifespan. A notable trend emerges as the frequency of changes increases. Websites with more frequent changes tend to have longer average lifespans. For instance, URLs with only one change have an average lifespan of 9.95 hours, while those with 15 changes survive an average of 18.31 hours. This trend continues, with URLs experiencing 50-99 changes lasting an average of 76.49 hours and those with 100+ changes persisting for an impressive 358.35 hours on average. This positive correlation between change frequency and lifespan suggests that frequent visual updates might be a strategy employed by phishers to evade detection and prolong their operations. URLs with 100+ changes show a standard deviation of 199.69 hours, highlighting the wide range of outcomes even within this most persistent group. Interestingly, while only 3.49% of URLs fall into this 100+ changes category,

they demonstrate remarkably extended lifespans, with a median of 404.52 hours (about 16.9 days). This suggests that a small proportion of highly adaptive phishing sites contribute disproportionately to the overall threat landscape by remaining active for extended periods as shown in Table 5.18

Table 5.18: Changed Phishing Lifespan Statistics (in hours).

| **Freq.*** | **URLs (%)** | **Q1** | **Med.** | **Avg.** | **Q3** | **Max** | **Std.** |
|---|---|---|---|---|---|---|---|
| 1 | 6,014 (8.39%) | 0.48 | 1.93 | 9.95 | 5.80 | 720 | 36.08 |
| 2 | 4,356 (6.08%) | 0.97 | 2.42 | 9.84 | 7.25 | 624 | 32.54 |
| 3 | 3,817 (5.33%) | 1.45 | 3.38 | 14.62 | 9.67 | 720 | 39.03 |
| 4 | 3,223 (4.50%) | 1.93 | 3.87 | 12.53 | 10.63 | 576 | 34.86 |
| 5 | 2,918 (4.07%) | 2.42 | 4.35 | 11.12 | 11.60 | 456 | 31.49 |
| 6 | 2,842 (3.97%) | 2.90 | 4.83 | 15.43 | 12.08 | 672 | 43.47 |
| 7 | 4,756 (6.64%) | 3.38 | 5.32 | 13.06 | 13.53 | 504 | 35.04 |
| 8 | 2,367 (3.30%) | 3.87 | 6.28 | 12.83 | 13.05 | 384 | 31.46 |
| 9 | 2,162 (3.02%) | 4.35 | 6.77 | 14.03 | 14.50 | 432 | 34.28 |
| 10 | 2,028 (2.83%) | 4.83 | 7.25 | 13.07 | 14.98 | 333 | 26.89 |
| 11 | 1,995 (2.78%) | 5.32 | 7.73 | 14.22 | 15.43 | 381 | 28.84 |
| 12 | 1,859 (2.59%) | 5.80 | 8.22 | 14.15 | 15.92 | 360 | 27.45 |
| 13 | 1,762 (2.46%) | 6.28 | 8.70 | 15.72 | 16.88 | 408 | 29.13 |
| 14 | 1,733 (2.42%) | 6.77 | 9.18 | 16.11 | 17.37 | 384 | 29.30 |
| 15 | 1,713 (2.39%) | 7.25 | 9.67 | 18.31 | 18.33 | 455 | 34.56 |
| 16 | 2,540 (3.54%) | 7.73 | 10.15 | 17.41 | 18.80 | 408 | 30.13 |
| 17 | 1,545 (2.16%) | 8.22 | 11.12 | 19.44 | 19.75 | 480 | 37.78 |
| 18 | 1,352 (1.89%) | 8.70 | 11.60 | 19.79 | 20.23 | 456 | 36.26 |
| 19 | 1,256 (1.75%) | 9.18 | 12.08 | 21.90 | 21.18 | 504 | 42.08 |
| 20-29 | 9,640 (13.45%) | 10.63 | 14.65 | 21.76 | 23.57 | 624 | 33.60 |
| 30-49 | 7,637 (10.66%) | 15.43 | 21.77 | 33.15 | 35.35 | 720 | 45.55 |
| 50-99 | 4,652 (6.49%) | 28.03 | 38.18 | 76.49 | 84.82 | 893 | 91.61 |
| 100+ | 2,498 (3.49%) | 240.30 | 404.52 | 358.35 | 480.75 | 1440 | 199.69 |
| **Total** | 71,665 (100%) | 3.38 | 14.65 | 30.62 | 28.03 | 1440 | 78.43 |

∗ **Freq.** stands for the frequency which phishing sites have changed their website layout or target brands.

# Chapter 6

# Future Research

**Future Research on Proactive Malicious Domain Detection.** This future research aims to develop a unified framework for the early detection of malicious domains that integrates features across the entire domain lifecycle. The primary motivation stems from the critical gap in current cybersecurity defenses, where malicious domains are typically identified only after attacks have occurred, giving attackers a significant time advantage. By detecting malicious intent at or near registration time, this work seeks to minimize the window during which attackers can operate, thus reducing the effectiveness of phishing campaigns, malware distribution networks, and command-and-control infrastructure. The study will leverage comprehensive multi-source datasets including APWG phishing blocklists, WHOIS/RDAP registration data, Certificate Transparency logs, passive DNS data with first-seen timestamps, and zone files with registration patterns. These datasets provide a rich foundation for understanding domain behavior across different stages of the malicious domain lifecycle. Additionally, this future research will incorporate website content data collected through targeted crawling to enable content-based analysis that distinguishes between maliciously registered domains and compromised legitimate websites. The technical approach will implement a three-stage classification pipeline

that combines methodologies from existing research: PREDATOR's registration-time analysis, Avalanche's proactive detection of algorithmically generated domains, and COMAR's distinction between malicious and compromised domains. This integrated approach will use gradient boosted tree ensembles with confidence scoring mechanisms to optimize the balance between automated classification and human review. By extracting over 30 features across lexical, temporal, structural, and behavioral dimensions, the system will capture the multifaceted characteristics that differentiate malicious registrations from legitimate ones. Expected results include a unified detection system capable of classifying domains with over 95% accuracy, significantly reducing detection times compared to current blocklisting approaches. This future research will also produce empirical insights into evolving attacker tactics and their adaptation to detection mechanisms, providing valuable intelligence for the cybersecurity community. The ultimate contribution will be a practical framework that security vendors, registrars, and network operators can integrate into their existing infrastructure, fundamentally shifting the industry from reactive to proactive defense postures against domain-based threats.

## 6.1 Overview

The proposed unified framework integrates methodologies from PREDATOR, COMAR, and Avalanche to create a comprehensive domain reputation system that operates across the entire domain lifecycle. The system architecture consists of data collection modules that continuously gather information from multiple sources, feature extraction engines that process this data into meaningful signals, a multi-stage classification pipeline that evaluates domains at different points in their lifecycle, and a confidence scoring mechanism that determines appropriate defensive actions. This approach addresses the limitations of existing systems by providing early detection while still distinguishing between maliciously registered and compromised domains. The framework operates as a continuous pipeline, where domains are evaluated at

registration time and then monitored throughout their operational lifecycle, with confidence scores updated as new information becomes available.

## 6.2 Data Collection Framework

### 6.2.1 Multi-Source Integration

The data collection framework aggregates information from diverse sources to provide comprehensive visibility into domain activity. APWG phishing blocklists, WHOIS/RDAP registration data, Certificate Transparency logs, passive DNS data, and zone files form the core dataset. Each data source is processed through dedicated collectors that normalize the information into a unified schema optimized for feature extraction.

### 6.2.2 Missing Data Compensation

Following COMAR's approach, the framework implements sophisticated methods to handle missing data, particularly for essential fields like domain registration dates. When WHOIS information is incomplete, the system estimates creation dates using the earliest observed timestamps from CT logs, passive DNS observations, or Internet Archive captures. This ensures robust feature computation even when the source data is incomplete.

## 6.3 Feature Engineering System

### 6.3.1 Registration-Time Features

Drawing from PREDATOR's methodology, the system extracts comprehensive registration-time features that capture domain profiles, lexical characteristics, and batch correlation patterns. These features include registrar selection patterns,

nameserver configurations, temporal registration behaviors, trigram analysis of domain names, and registration burst detection. These signals provide early indicators of potentially malicious intent before domains become operational.

### 6.3.2 Content and Infrastructure Features

Following COMAR's approach, the system analyzes website content and infrastructure configurations when they become available. Features include content length, technological fingerprints, internal and external hyperlink patterns, and vulnerable technology detection. For domains hosting content, these features help distinguish between legitimately registered domains that have been compromised and those registered with malicious intent.

### 6.3.3 Behavioral and Network Features

Inspired by the Avalanche study, the framework incorporates behavioral features that monitor how domains operate over time. These include DNS record stability patterns, certificate renewal behaviors, and hosting infrastructure changes. By tracking these behaviors, the system can identify domains that exhibit characteristics consistent with command and control infrastructure or phishing campaigns.

## 6.4 Multi-Stage Classification Pipeline

### 6.4.1 Registration Analysis Module

The first stage of the classification pipeline evaluates domains at registration time using gradient boosted tree models trained on historical data. This module implements PREDATOR's approach of identifying suspicious registrations before domains become operational, providing an early warning system for potentially malicious domains.

The registration analysis module assigns initial reputation scores based solely on registration characteristics.

**Registrar Reputation Features**

- Registrar historical abuse ratio (percentage of domains registered through this registrar that later appeared on blocklists)
- Registrar pricing tier (lower-cost registrars are often favored by attackers [182])
- Registrar geographic location and jurisdiction
- Registration time patterns (hour of day, day of week distributions)

**Domain Name Lexical Features**

- Character frequency distribution (n-gram analysis)
- Presence of brand names or misspelled variants (using edit distance metrics)
- Length of domain name and ratio of digits/special characters
- Entropy measures of randomness in a domain name
- Semantic analysis of domain name components

**Registration Batch Features**

- Temporal clustering of registrations (number of domains registered within the same timeframe)
- Structural similarity between domain names in the same batch
- Shared nameserver infrastructure within registration batch
- Abnormal registration velocity compared to baseline

**Historical Pattern Features**

- Edit distance to previously blocklisted domains
- Similarity to known DGA patterns from different malware families
- Recurrence of registration patterns across different time periods
- Life-cycle categorization (brand-new, drop-catch, or retread domains)

**Infrastructure Setup Features**

- Self-resolving nameserver configuration
- Authoritative nameservers and their reputation scores
- ASN diversity and reputation of hosting infrastructure
- DNSSEC implementation status

### 6.4.2 Domain Classification Module

Once domains become operational, the domain classification module applies COMAR's methodology to distinguish between compromised and maliciously registered domains. This distinction is crucial for determining appropriate mitigation strategies—compromised domains require notification to legitimate owners, while malicious registrations warrant more aggressive interventions. This module updates domain reputation scores based on operational characteristics.

**Content-Based Features**

- Webpage content length and complexity metrics
- Number of internal working hyperlinks (with unique content)
- Ratio of external to internal links

- Presence of default index pages or common CMS templates
- Content relationship to domain name (semantic matching between domain name and visible text)
- Technologies used in website construction (JavaScript libraries, frameworks, CMS platforms)
- Presence of vulnerable technologies or outdated software versions
- HTML structure complexity and sophistication
- Content consistency across different pages of the domain

**TLS Certificate Features**

- Certificate authority type (free vs. paid)
- Certificate validity period
- Subject Alternative Name (SAN) configurations
- Certificate transparency log appearance timing
- Historical certificate issuance patterns
- Certificate revocation status

**Hosting Infrastructure Features**

- Hosting provider reputation and abuse history
- IP address geolocation consistency with registrant information
- Autonomous system reputation and diversity
- Infrastructure stability (frequency of IP address changes)

- Reverse proxy usage patterns
- Server-side technology fingerprints

**Domain Usage Patterns**

- Wayback Machine capture history and frequency
- Search engine indexing depth and history
- Website popularity metrics from multiple ranking sources
- DNS record completeness (presence of MX, TXT, DMARC records)
- Email configuration quality (SPF, DKIM implementation)

**Temporal Features**

- Domain age relative to first suspicious activity
- Time between registration and content deployment
- Historical changes in content and infrastructure
- Consistency of activity patterns over time

### 6.4.3 Algorithm Detection Component

The algorithm detection component specializes in identifying domains generated by algorithms (DGAs), leveraging techniques from the Avalanche study. This component analyzes linguistic patterns, registration behaviors, and network configurations to identify coordinated domain campaigns. By recognizing algorithmic patterns, the system can proactively block entire families of malicious domains based on a small number of observed examples.

## 6.5 Confidence Scoring and Decision System

### 6.5.1 Unified Reputation Mechanism

The confidence scoring system combines evidence from all three classification stages to produce a unified reputation score. This mechanism implements a weighted ensemble approach that accounts for the reliability of different feature types across the domain lifecycle. The system maintains confidence intervals around reputation scores to reflect uncertainty in the classification.

### 6.5.2 Threshold Optimization

To balance false positives against detection rates, the framework implements dynamic thresholds that adapt to changing threat landscapes. These thresholds determine when domains should be automatically blocked, flagged for human review, or allowed to operate. The threshold optimization component continuously evaluates system performance and adjusts decision boundaries to maintain target accuracy levels.

### 6.5.3 Analyst Interface and Feedback Loop

For domains that fall into uncertainty regions, the system presents relevant evidence to security analysts through an intuitive interface. Analyst decisions are captured and fed back into the system to improve future classifications. This human-in-the-loop approach ensures that edge cases receive appropriate attention while continually improving automated decision-making.

## 6.6 Deployment and Integration

### 6.6.1 Real-Time Processing Infrastructure

The framework is implemented as a scalable, real-time processing infrastructure built on modern streaming technologies. Domain registration events trigger immediate analysis, with results available within seconds of a new domain being registered. Continuous monitoring processes update reputation scores as new information becomes available, ensuring that the system maintains current assessments of all domains.

## 6.7 Evaluation and Performance Monitoring

### 6.7.1 Metrics and Benchmarking

The framework includes comprehensive performance monitoring that tracks accuracy, precision, recall, false positive rates, and detection lead times compared to traditional blocklists. A primary goal is achieving detection rates above 95%, significantly increase performance of existing reputation systems.

### 6.7.2 Adversarial Testing

To ensure resistance to evasion, the system undergoes continuous adversarial testing that simulates sophisticated attack techniques. This testing identifies potential weaknesses in the classification system and drives feature engineering to address emerging evasion strategies. The multi-faceted nature of the feature set ensures that attackers must overcome multiple independent detection mechanisms to evade classification.

### 6.7.3 Continuous Improvement Cycle

A continuous improvement cycle integrates operational feedback, emerging threat intelligence, and performance metrics to refine the classification models. Regular retraining incorporates newly observed attack patterns, ensuring that the system adapts to evolving threats. This adaptive approach maintains detection effectiveness in the face of changing attacker behaviors.

## 6.8 Future Extensions and Research Directions

The unified framework establishes a foundation for ongoing research into proactive domain reputation systems. Future extensions include deeper integration with endpoint security solutions, expanded analysis of internationalized domain names, and incorporation of browser telemetry to identify domains involved in drive-by downloads. As attack techniques evolve, the modular architecture allows for integration of new detection methodologies without disrupting existing capabilities. By implementing this comprehensive framework, organizations can significantly reduce the time window during which attackers can operate, thereby minimizing the effectiveness of phishing campaigns, malware distribution networks, and command-and-control infrastructure.

# Chapter 7

# Conclusions

Phishing remains a persistent and evolving cybersecurity threat, exploiting user trust through deceptive website construction, domain abuse, and evasion techniques. This dissertation provides a comprehensive analysis of phishing websites, examining their client-side structures, server configurations, and domain registration strategies. By measuring phishing attackers' behaviors and evaluating the effectiveness of existing defense mechanisms, we identify key gaps in blocklist-based protections and highlight the challenges posed by detection delays. Our findings underscore the need for more proactive and adaptive phishing detection models that integrate domain registration analysis, client-side behavioral patterns, and DNS-based threat intelligence. By bridging the gap between attacker strategies and defense limitations, this research aims to enhance phishing mitigation techniques, ultimately reducing the impact of phishing attacks and strengthening online security.

# Bibliography


[1] Fancybox - fancy jquery lightbox alternative. http://fancybox.net/, 11 2010. (Accessed on 09/27/2023). 21

[2] jquery 1.2 released — official jquery blog. https://blog.jquery.com/2007/09/10/jquery-1-2-released/#jQuery_1.1_Compatibility_Plugin, 09 2011. (Accessed on 05/26/2023). 41

[3] Cross-site scripting (xss) in jquery-migrate — snyk. https://security.snyk.io/vuln/npm:jquery-migrate:20130419, 04 2013. (Accessed on 05/26/2023). 37

[4] Js bin - collaborative javascript debugging. https://jsbin.com/UQEgAsO/3/edit?html,output, 11 2013. (Accessed on 05/26/2023). 43

[5] Xss · issue #36 · jquery/jquery-migrate. https://github.com/jquery/jquery-migrate/issues/36, 04 2013. (Accessed on 05/26/2023). 37

[6] Full disclosure: Xss reflected jquery 1.4.2 - create object option in runtime client-side. https://seclists.org/fulldisclosure/2014/Sep/10, 09 2014. (Accessed on 05/26/2023). 43

[7] Internet archive: Digital library of free & borrowable books, movies, music & wayback machine. https://archive.org/, 12 2014. (Accessed on 09/26/2023). 18

[8] Xss vulnerability on closetext option of dialog jquery ui · issue #281 · jquery/api.jqueryui.com. https://github.com/jquery/api.jqueryui.com/issues/281, 10 2015. (Accessed on 05/26/2023). 43

[9] Hammer.js hammer.js. https://hammerjs.github.io/, 04 2016. (Accessed on 09/27/2023). 21

[10] Lodash. https://lodash.com/, 01 2016. (Accessed on 09/27/2023). 21

[11] swfobject/swfobject: An open source javascript framework for detecting the adobe flash player plugin and embedding flash (swf) files. https://github.com/swfobject/swfobject, 06 2016. (Accessed on 05/26/2023). 35, 42

[12] Flash player is no longer available - google chrome help. https://support.google.com/chrome/answer/6258784?hl=en, 07 2017. (Accessed on 05/26/2023). 54

[13] Js bin - collaborative javascript debugging. https://jsbin.com/qalekeroke/edit?html,output, 08 2017. (Accessed on 05/26/2023). 43

[14] slick - the last carousel you'll ever need. https://kenwheeler.github.io/slick/, 09 2017. (Accessed on 09/27/2023). 21

[15] Chrome releases: Stable channel update for desktop. https://chromereleases.googleblog.com/2018/09/stable-channel-update-for-desktop_17.html, 09 2018. (Accessed on 05/26/2023). 7

[16] Cve-2012-6708 : jquery before 1.9.0 is vulnerable to cross-site scripting (xss) attacks. the jquery(strinput) function does not differen. https://www.cvedetails.com/cve/CVE-2012-6708/?q=CVE-2012-6708, 01 2018. (Accessed on 05/26/2023). 40

[17] Cve-2015-9251 : jquery before 3.0.0 is vulnerable to cross-site scripting (xss) attacks when a cross-domain ajax request is performed wi. https://www.

cvedetails.com/cve/CVE-2015-9251/, 01 2018. (Accessed on 05/26/2023). 41

[18] Js bin - collaborative javascript debugging. https://jsbin.com/palokaxina/edit?html,output, 08 2018. (Accessed on 05/26/2023). 43

[19] Js bin - collaborative javascript debugging. https://jsbin.com/xeminoniku/edit?html,output, 05 2018. (Accessed on 05/26/2023). 43

[20] Requirejs. https://requirejs.org/, 08 2018. (Accessed on 09/27/2023). 21

[21] Compatibility issue with jquery 3.4.x — web-datarocks. https://www.webdatarocks.com/question/compatibility-issue-with-jquery-3-4-x-2/, 10 2019. (Accessed on 05/26/2023). 41

[22] Cve-2019-11358 : jquery before 3.4.0, as used in drupal, backdrop cms, and other products, mishandles jquery.extend(true, {}, ...) becaus. https://www.cvedetails.com/cve/CVE-2019-11358/, 04 2019. (Accessed on 05/26/2023). 40, 41

[23] Cross-site scripting (xss) in jquery — cve-2020-7656 — snyk. https://security.snyk.io/vuln/SNYK-JS-JQUERY-569619, 05 2020. (Accessed on 05/26/2023). 43, 59

[24] Cve-2020-11022 : In jquery versions greater than or equal to 1.2 and before 3.5.0, passing html from untrusted sources - even after sanit. https://www.cvedetails.com/cve/CVE-2020-11022/, 04 2020. (Accessed on 05/26/2023). 40, 41

[25] Cve-2020-11023 : In jquery versions greater than or equal to 1.0.3 and before 3.5.0, passing html containing ¡option¿ elements from. https://www.cvedetails.com/cve/CVE-2020-11023/, 04 2020. (Accessed on 05/26/2023). 40, 41

[26] Safari 14 and flash player - apple community. https://discussions.apple.com/thread/251900220, 10 2020. (Accessed on 05/26/2023). 7, 54

[27] Compatibility issues with latest jquery 3.5.1. https://datatables.net/forums/discussion/67375/compatibility-issues-with-latest-jquery-3-5-1, 03 2021. (Accessed on 05/26/2023). 41

[28] Cve-2020-27511 : An issue was discovered in the striptags and unescapehtml components in prototype 1.7.3 where an attacker can cause a re. https://www.cvedetails.com/cve/CVE-2020-27511/?q=CVE-2020-27511, 06 2021. (Accessed on 05/26/2023). 41

[29] End of support for adobe flash — firefox help. https://support.mozilla.org/en-US/kb/end-support-adobe-flash, 01 2021. (Accessed on 05/26/2023). 54

[30] Getting started: Select2 - the jquery replacement for select boxes. https://select2.org/, 01 2021. (Accessed on 09/27/2023). 21

[31] Github akamai boomerang: End user oriented web performance testing and beaconing. https://github.com/akamai/boomerang, 08 2021. (Accessed on 09/27/2023). 21

[32] Update on adobe flash player end of support - microsoft edge blog. https://blogs.windows.com/msedgedev/2020/09/04/update-adobe-flash-end-support/, 04 2021. (Accessed on 05/26/2023). 54

[33] Angularjs superheroic javascript mvw framework. https://angularjs.org/, 01 2022. (Accessed on 09/27/2023). 21

[34] Commits · js-cookie/js-cookie. https://github.com/js-cookie/js-cookie/commits/main, 10 2022. (Accessed on 05/26/2023). 42

[35] Cve - cve. https://cve.mitre.org/index.html, 07 2022. (Accessed on 05/26/2023). 31, 34

[36] Cve security vulnerability database. security vulnerabilities, exploits, references and more. https://www.cvedetails.com/index.php, 07 2022. (Accessed on 05/26/2023). 31, 34

[37] Html attribute: crossorigin - html: Hypertext markup language — mdn. https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/crossorigin, 10 2022. (Accessed on 05/26/2023). 49

[38] jquery ui. https://jqueryui.com/, 07 2022. (Accessed on 09/27/2023). 21

[39] js-cookie/js-cookie: A simple, lightweight javascript api for handling browser cookies. https://github.com/js-cookie/js-cookie, 10 2022. (Accessed on 05/26/2023). 35, 37, 42

[40] Moment.js — home. https://momentjs.com/, 07 2022. (Accessed on 09/27/2023). 21

[41] Nvd - vulnerabilities. https://nvd.nist.gov/vuln, 09 2022. (Accessed on 05/26/2023). 31, 34

[42] Request.credentials - web apis — mdn. https://developer.mozilla.org/en-US/docs/Web/API/Request/credentials, 09 2022. (Accessed on 05/26/2023). 49

[43] Bootstrap · the most popular html, css, and js library in the world. https://getbootstrap.com/, 09 2023. (Accessed on 09/27/2023). 21

[44] Browser support — jquery. https://jquery.com/browser-support/, 01 2023. (Accessed on 05/26/2023). 41

[45] clipboard.js — copy to clipboard without flash. https://clipboardjs.com/, 09 2023. (Accessed on 09/27/2023). 21, 23

[46] Home — owl carousel — 2.3.4. https://owlcarousel2.github.io/OwlCarousel2/, 09 2023. (Accessed on 09/27/2023). 21

[47] jquery. https://jquery.com/, 09 2023. (Accessed on 09/27/2023). 21

[48] Modernizr: the feature detection library for html5/css3. https://modernizr.com/, 09 2023. (Accessed on 09/27/2023). 21

[49] React – a javascript library for building user interfaces. https://legacy.reactjs.org/, 09 2023. (Accessed on 09/27/2023). 21

[50] Socket.io. https://socket.io/, 09 2023. (Accessed on 09/27/2023). 21, 23

[51] styled-components. https://styled-components.com/, 10 2023. (Accessed on 09/27/2023). 21

[52] Sweetalert2 - a beautiful, responsive, customizable and accessible (wai-aria) replacement for javascript's popup boxes. https://sweetalert2.github.io/, 10 2023. (Accessed on 09/27/2023). 21

[53] Technologies - wappalyzer. https://www.wappalyzer.com/technologies/, 08 2023. (Accessed on 09/18/2023). 19

[54] Vue.js - the progressive javascript framework — vue.js. https://vuejs.org/, 09 2023. (Accessed on 09/27/2023). 21

[55] Vulnerability db — snyk. https://security.snyk.io/vuln, 01 2023. (Accessed on 05/26/2023). 31, 34

[56] wappalyzer/wappalyzer: Identify technology on websites. https://github.com/wappalyzer/wappalyzer, 1 2023. (Accessed on 05/26/2023). 34

[57] ChromeDriver. https://chromedriver.chromium.org, 2024. (Accessed on 09/13/2024). 15, 61

[58] Contributing rules — Semgrep. https://semgrep.dev/docs/contributing/contributing-to-semgrep-rules-repository, 09 2024. (Accessed on 05/15/2024). 74

[59] CWE - CWE-1004: Sensitive Cookie Without 'HttpOnly' Flag. https://cwe.mitre.org/data/definitions/1004.html, 09 2024. (Accessed on 09/14/2024). 72

[60] CWE - CWE-22: Improper Limitation of a Pathname to a Restricted Directory. https://cwe.mitre.org/data/definitions/22.html, 09 2024. (Accessed on 09/14/2024). 74

[61] CWE - CWE-502: Deserialization of Untrusted Data. https://cwe.mitre.org/data/definitions/502.html, 09 2024. (Accessed on 09/14/2024). 74

[62] CWE - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'). https://cwe.mitre.org/data/definitions/79.html, 09 2024. (Accessed on 09/14/2024). 72

[63] CWE - CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'). https://cwe.mitre.org/data/definitions/89.html, 09 2024. (Accessed on 09/14/2024). 72

[64] CWE - CWE-918: Server-Side Request Forgery (SSRF). https://cwe.mitre.org/data/definitions/918.html, 09 2024. (Accessed on 09/14/2024). 72

[65] Fast web fuzzer, 09 2024. (Accessed on 09/19/2024). 77

[66] Google Safe Browsing. https://safebrowsing.google.com/, 06 2024. (Accessed on 10/30/2023). 2, 127

[67] HTTP headers - HTTP — MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers, 09 2024. (Accessed on 09/07/2024). 65

[68] malware-filter. https://gitlab.com/malware-filter/phishing-filter, 06 2024. (Accessed on 10/14/2024). 128, 143

[69] Openphish. https://openphish.com/, 06 2024. (Accessed on 10/14/2024). 7, 9, 128, 143

[70] Phishing army. https://phishing.army, 06 2024. (Accessed on 10/14/2024). 128, 143

[71] Phishstats. https://phishstats.info/, 06 2024. (Accessed on 10/14/2024). 7, 128, 143

[72] Phishtank. https://phishtank.org/, 06 2024. (Accessed on 10/14/2024). 7, 9, 128, 143

[73] Phishunt. https://phishunt.io/, 06 2024. (Accessed on 10/14/2024). 128, 143

[74] Public suffix list, 06 2024. (Accessed on 10/11/2024). 131

[75] publicsuffix/list: The Public Suffix List. https://github.com/publicsuffix/list?tab=readme-ov-file, 09 2024. (Accessed on 09/18/2024). 64

[76] RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. https://datatracker.ietf.org/doc/html/rfc2616#section-15.1.1, 09 2024. (Accessed on 09/08/2024). 70

[77] Semgrep, 09 2024. (Accessed on 09/06/2024). 64, 72

[78] shellray. https://shellray.com/, 09 2024. (Accessed on 09/13/2024). 76

[79] Stegosploit, 09 2024. (Accessed on 09/06/2024). 64, 72

[80] The Apache HTTP Server Project. https://httpd.apache.org/, 09 2024. (Accessed on 09/07/2024). 2

[81] The APWG eCrime Exchange (eCX). https://apwg.org/ecx/, 2024. (Accessed on 09/13/2024). 15, 62, 122

[82] VirusShare.com. https://virusshare.com/, 09 2024. (Accessed on 09/13/2024). 76

[83] 360. 360 browser. https://browser.360.cn/ee/mac/index.html, 01 2023. (Accessed on 05/26/2023). 55

[84] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. You get where you're looking for: The impact of information sources on code security. In 2016 IEEE Symposium on Security and Privacy (SP), pages 289–305. IEEE, 2016. 49

[85] Bhupendra Acharya and Phani Vadrevu. {PhishPrint}: Evading phishing detection crawlers by prior profiling. In 30th USENIX Security Symposium (USENIX Security 21), pages 3775–3792, 2021. 116

[86] Adguard. Adguardfilters/specific.txt at master · adguardteam/adguardfilters. https://github.com/AdguardTeam/AdguardFilters/blob/master/SpywareFilter/sections/specific.txt, 01 2023. (Accessed on 05/26/2023). 48

[87] Adobe. Control access to scripts — host web page. https://helpx.adobe.com/flash/kb/control-access-scripts-host-web.html, 04 2017. (Accessed on 05/26/2023). 56

[88] Adobe. Create html5 canvas documents in animate. https://helpx.adobe.com/animate/using/creating-publishing-html5-canvas-document.html, 12 2021. (Accessed on 05/26/2023). 54

[89] Adobe. Best practices to convert/publish existing flash-based projects to html5 in captivate. https://helpx.adobe.com/captivate/kb/

best-practices-convert-flash-html5-captivate.html, 01 2022. (Accessed on 05/26/2023). 54

[90] aFarkas. 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, 03 2021. (Accessed on 09/27/2023). 21

[91] Antonia Affinito, Raffaele Sommese, Gautam Akiwate, Stefan Savage, kc claffy, Geoffrey M Voelker, Alessio Botta, and Mattijs Jonker. Domain name lifetimes: Baseline and threats. In TMA, 2022. 100

[92] Vibhor Agarwal and Nishanth Sastry. "way back then": A data-driven view of 25+ years of web evolution. In Proceedings of the ACM Web Conference 2022, pages 3471–3479, 2022. 18

[93] National Security Agency. Csa - continued use of adobe flash invites compromise.pdf. https://media.defense.gov/2019/Sep/25/2002186834/-1/-1/0/CSA%20-%20CONTINUED%20USE%20OF%20ADOBE%20FLASH%20INVITES%20COMPROMISE.PDF, 09 2019. (Accessed on 05/26/2023). 54

[94] Alibaba. Domain name bulk registration-basic edition. https://check.aliyun.com/domain/bulk-search/basic.htm, 11 2024. (Accessed on 11/16/2024). 90

[95] Alibaba. New user discount! only $0.5 - alibaba cloud. https://www.alibabacloud.com/ko/domain/1dollar?_p_lc=1&spm=a3c0i.145322.1761468070.2.522a4635pLdxUS, 11 2024. (Accessed on 11/16/2024). 90

[96] Mshabab Alrizah, Sencun Zhu, Xinyu Xing, and Gang Wang. Errors, misunderstandings, and attacks: Analyzing the crowdsourcing process of ad-blocking systems. In Proceedings of the Internet Measurement Conference,

IMC '19, page 230–244, New York, NY, USA, 2019. Association for Computing Machinery. 18

[97] Amir Alush, Dickson Neoh, and Danny Bickson et al. Fastdup. GitHub.Note: https://github.com/visuallayer/fastdup, 2023. 16, 63

[98] Danny E. Alvarez, Daniel B. Correa, and Fernando I. Arango. 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), pages 1–5, 2016. 19, 34

[99] Babak Amin Azad, Oleksii Starov, Pierre Laperdrix, and Nick Nikiforakis. Web runner 2049: Evaluating third-party anti-bot services. In Proc. of the Detection of Intrusions and Malware, and Vulnerability Assessment, 2020. 61

[100] APWG. Apwg — the apwg ecrime exchange (ecx). https://apwg.org/ecx/, 06 2023. (Accessed on 09/14/2024). 2, 7, 9, 106, 107

[101] APWG. Apwg — members directory, 06 2024. [Online; accessed 2025-02-04]. 128

[102] APWG. Apwg — the apwg ecrime exchange (ecx). https://apwg.org/ecx/, 11 2024. (Accessed on 11/17/2024). 84, 85

[103] APWG. Apwg — unifying the global response to cybercrime. https://apwg.org/, 08 2024. (Accessed on 09/07/2024). 106

[104] Archive. Inside safari 3.2's anti-phishing features. https://web.archive.org/web/20210521014149/https://www.macworld.com/article/193605/safari-safe-browsing.html, 06 2024. (Accessed on 09/13/2024). 124

[105] Archive. Wayback machine, 09 2024. [Online; accessed 2024-08-30]. 66

[106] Archive. Wayback machine. https://web.archive.org/, 11 2024. (Accessed on 11/20/2024). 91

[107] axios. Getting started — axios docs. https://axios-http.com/docs/intro, 10 2023. (Accessed on 09/27/2023). 21

[108] Simon Bell and Peter Komisarczuk. An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank. In Proceedings of the Australasian Computer Science Week Multiconference, pages 1–11, 2020. 11, 12

[109] Berstend, 06 2024. (Accessed on 03/13/2024). 123

[110] Hugo Bijmans, Tim Booij, Anneke Schwedersky, Aria Nedgabat, and Rolf van Wegberg. Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection. In Proc. of the USENIX security symposium, 2021. 64

[111] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In Proc. of the Network and Distributed System Security (NDSS), 2011. 98, 140

[112] Bootstrap. Bootstrap · the most popular html, css, and js library in the world. https://getbootstrap.com/, 01 2023. (Accessed on 05/26/2023). 35

[113] CAIDA. home - dzdb. https://dzdb.caida.org/, 11 2024. (Accessed on 11/15/2024). 87

[114] Davide Canali, Davide Balzarotti, and Aurélien Francillon. The role of web hosting providers in detecting compromised websites. In Proc. of the international conference on World Wide Web (WWW), 2013. 88

[115] Choices-js. Github choices-js/choices: A vanilla js customisable select box/text input plugin. https://github.com/Choices-js/Choices, 11 2022. (Accessed on 09/27/2023). 21

[116] Alibaba Cloud. Alibaba cloud. https://www.alibabacloud.com/, 11 2024. (Accessed on 11/19/2024). 90

[117] Cloudflare. Cloudflare free ssl/tls. https://www.cloudflare.com/application-services/products/ssl/, 11 2024. (Accessed on 11/20/2024). 96

[118] Marco Cova, Christopher Kruegel, and Giovanni Vigna. There is no free phish: An analysis of" free" and live phishing kits. Proc. of the WOOT, 2008. 62

[119] curbengh. curbengh/malware-filter: Mirror of https://gitlab.com/malware-filter/malware-filter. https://github.com/curbengh/malware-filter, 11 2024. (Accessed on 11/05/2024). 84, 85

[120] Cybersecurity and Infrastructure Security Agency (CISA). Phishing infrastructure trends: Use of cdns and public clouds. https://www.cisa.gov/, 2021. 134

[121] David Dagon, Niels Provos, Christopher P Lee, and Wenke Lee. Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority. In Proc. of the Annual Network and Distributed System Security Symposium (NDSS), 2008. 98, 140

[122] Ravindu De Silva, Mohamed Nabeel, Charith Elvitigala, Issa Khalil, Ting Yu, and Chamath Keppitiyagama. Compromised or {Attacker-Owned}: A large scale classification and study of hosting domains of malicious {URLs}. In 30th USENIX security symposium (USENIX security 21), pages 3721–3738, 2021. 88

[123] Nurullah Demir, Tobias Urban, Kevin Wittek, and Norbert Pohlmann. Our (in)secure web: Understanding update behavior of websites and its impact on security. In Passive and Active Measurement, pages 76–92, Cham, 2021. Springer International Publishing. 19, 34

[124] dnstwist. elceef/dnstwist: Domain name permutation engine for detecting homograph phishing attacks, typo squatting, and brand impersonation. https://github.com/elceef/dnstwist, 05 2024. (Accessed on 09/09/2024). 117

[125] DomainTools. Introducing dnsdb 2.0 — passive dns — domaintools. https://www.domaintools.com/products/farsight-dnsdb/, 11 2024. (Accessed on 11/15/2024). 87

[126] Vincent Drury, Luisa Lux, and Ulrike Meyer. Dating phish: An analysis of the life cycles of phishing attacks and campaigns. In Proc. of the International Conference on Availability, Reliability and Security (ARES), 2022. 12, 105

[127] Carlos Duarte, Inês Matos, João Vicente, Ana Salvado, Carlos M. Duarte, and Luís Carriço. Development technologies impact in web accessibility. In Proceedings of the 13th International Web for All Conference, W4A '16, New York, NY, USA, 2016. Association for Computing Machinery. 19, 34

[128] elceef. elceef/dnstwist: Domain name permutation engine for detecting homograph phishing attacks, typo squatting, and brand impersonation. https://github.com/elceef/dnstwist, 09 2024. (Accessed on 11/16/2024). 89

[129] emotion js. Github emotion-js/emotion: Css-in-js library designed for high performance style composition. https://github.com/emotion-js/emotion, 6 2023. (Accessed on 09/27/2023). 21

[130] Ronald A Fisher. On the interpretation of $\chi$ 2 from contingency tables, and the calculation of P. Journal of the royal statistical society, 85(1):87–94, 1922. 129

[131] Fox-IT. Using anomaly detection to find malicious domains – fox-it international blog. https://blog.fox-it.com/2019/06/11/using-anomaly-detection-to-find-malicious-domains/, 06 2019. (Accessed on 11/16/2024). 89

[132] Tillson Galloway, Kleanthis Karakolios, Zane Ma, Roberto Perdisci, Angelos Keromytis, and Manos Antonakakis. Practical Attacks Against DNS Reputation Systems. In Proc. of the IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2024. 98, 140

[133] GitHub. Update regex for striptags method to prevent regex dos by jwestbrook · pull request #349 · prototypejs/prototype. https://github.com/prototypejs/prototype/pull/349, 01 2021. (Accessed on 05/26/2023). 41

[134] Roberto Gonzalez, Lili Jiang, Mohamed Ahmed, Miriam Marciel, Ruben Cuevas, Hassan Metwalley, and Saverio Niccolini. The cookie recipe: Untangling the use of cookies in the wild. In 2017 Network Traffic Measurement and Analysis Conference (TMA), pages 1–9. IEEE, 2017. 68

[135] google. Overview safe browsing apis (v4) google for developers. https://developers.google.com/safe-browsing/v4, 06 2023. (Accessed on 09/04/2024). 105

[136] Google. Safe browsing – google safe browsing. https://safebrowsing.google.com/, 06 2023. (Accessed on 06/08/2023). 124

[137] google. Urls and hashing safe browsing apis (v4) google for developers. https://developers.google.com/safe-browsing/v4/urls-hashing, 06 2023. (Accessed on 09/12/2024). 105, 124

[138] google. Urls and hashing safe browsing apis (v4) google for developers. https://developers.google.com/safe-browsing/v4/urls-hashing#suffixprefix-expressions, 06 2023. (Accessed on 09/14/2024). 110

[139] Google. Google transparency report. https://transparencyreport.google.com/safe-browsing/overview?hl=en&warnings_displayed=dataset:users;start:1288508400000;end:1672560000000&lu=warnings_displayed, 01 2024. (Accessed on 09/29/2024). 7, 9

[140] Google. Puppeteer, 06 2024. (Accessed on 08/13/2024). 123

[141] google. Safe browsing faqs - transparency report help center. https://support.google.com/transparencyreport/answer/

7380435?hl=en#zippy=%2Chow-accurate-is-this-information%2Chow-frequently-do-you-scan-your-web-index, 02 2024. (Accessed on 09/14/2024). 108, 109

[142] GoogleChrome. Github googlechrome/web-vitals: Essential metrics for a healthy site. https://github.com/GoogleChrome/web-vitals, 10 2023. (Accessed on 09/27/2023). 21

[143] Anders Gorboe. Detection of fake domain names in e-mails. Master's thesis, OsloMet-storbyuniversitetet, 2022. 89

[144] Thamme Gowda and Chris A Mattmann. Clustering web pages based on structure and style similarity (application paper). In 2016 IEEE 17th International conference on information reuse and integration (IRI), pages 175–180. IEEE, 2016. 27

[145] GreenSock. Gsap - greensock. https://greensock.com/gsap/, 10 2023. (Accessed on 10/10/2023). 21

[146] Fabian Gröger, Simone Lionetti, Philippe Gottfrois, Alvaro Gonzalez-Jimenez, Ludovic Amruthalingam, Labelling Consortium, Matthew Groh, Alexander A. Navarini, and Marc Pouly. Selfclean: A self-supervised data cleaning strategy, 2023. 16

[147] Srishti Gupta and Ponnurangam Kumaraguru. Emerging phishing trends and effectiveness of the anti-phishing landing page. In 2014 APWG Symposium on Electronic Crime Research (eCrime), pages 36–47. IEEE, 2014. 15

[148] 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, 2016. 88

[149] Shuang Hao, Matthew Thomas, Vern Paxson, Nick Feamster, Christian Kreibich, Chris Grier, and Scott Hollenbeck. Understanding the domain registration behavior of spammers. In Proc. of the conference on Internet Measurement Conference (IMC), 2013. 90

[150] Hao He, Lulu Chen, and Wenpu Guo. 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), pages 150–155. Atlantis Press, 2017/03. 19, 34

[151] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015. 116

[152] Cheng Huang, Nabil Samaan, and Danny De Vleeschauwer. CDN-enabled secure delivery of web content. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 14(3):1–20, 2018. 134

[153] ICANN. About zone file access - icann. https://www.icann.org/resources/pages/zfa-2013-06-28-en, 06 2013. (Accessed on 11/15/2024). 87

[154] Interisle. Phishing landscape 2024: An annual study of the scope and distribution of phishing — interisle consulting group. https://interisle.net/insights/phishing-landscape-2024-an-annual-study-of-the-scope-and-distribution-of-phi
07 2024. (Accessed on 11/17/2024). 88, 91, 92, 93

[155] ionos. Dns ttl best practices: Understanding and configuring dns ttl - ionos. https://www.ionos.com/digitalguide/server/configuration/understanding-and-configuring-dns-ttl/, 11 2024. (Accessed on 11/19/2024). 98

[156] IPinfo. Ip summarization & data visualization - ipinfo.io. https://ipinfo.io/tools/summarize-ips, 11 2024. (Accessed on 11/22/2024). 97

[157] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. The ad wars: Retrospective measurement and analysis of anti-adblock filter lists. In Proceedings of the 2017 Internet Measurement Conference, IMC '17, page 171–183, New York, NY, USA, 2017. Association for Computing Machinery. 18

[158] Isotope. Isotope - filter & sort magical layouts. https://isotope.metafizzy.co/, 01 2023. (Accessed on 05/26/2023). 35

[159] JakeChampion. Polyfill.io. https://polyfill.io/v3/, 06 2023. (Accessed on 09/27/2023). 21

[160] jashkenas. Backbone.js. https://backbonejs.org/, 07 2023. (Accessed on 10/10/2023). 21

[161] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. arXiv preprint arXiv:1612.03651, 2016. 117

[162] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 427–431. Association for Computational Linguistics, April 2017. 117

[163] jquery. 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, 02 2023. (Accessed on 09/27/2023). 21

[164] jQuery. jquery. https://jquery.com/, 01 2023. (Accessed on 05/26/2023). 35

[165] jquery cookie. carhartl/jquery-cookie: No longer maintained, superseded by js cookie:. https://github.com/carhartl/jquery-cookie, 03 2015. (Accessed on 05/26/2023). 35, 42

[166] jquery migrate. jquery/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, 01 2023. (Accessed on 05/26/2023). 35, 36

[167] jQuery UI. jquery ui. https://jqueryui.com/, 01 2023. (Accessed on 05/26/2023). 35

[168] Keshav Kaushik, Sahajpreet Singh, Saksham Garg, Sarthak Singhal, and Shambhavi Pandey. Exploring the mechanisms of phishing. Computer Fraud & Security, 2021(11):14–19, 2021. 89

[169] Doowon Kim, Haehyun Cho, Yonghwi Kwon, Adam Doupé, Sooel Son, Gail-Joon Ahn, and Tudor Dumitras. 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, pages 407–420, 2021. 62, 84, 105, 122, 125

[170] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. Catching transparent phish: Analyzing and detecting mitm phishing toolkits. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 36–50, 2021. 26

[171] Maciej Korczynski, Maarten Wullink, Samaneh Tajalizadehkhoob, Giovane CM Moura, Arman Noroozian, Drew Bagley, and Cristian Hesselman. Cybercrime after the sunrise: A statistical analysis of dns abuse in new gtlds. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pages 609–623, 2018. 90

[172] Mitchell Krog. Phishing Database. https://github.com/mitchellkrogza/Phishing.Database, 06 2024. (Accessed on 10/14/2024). 128, 143

[173] Tobias Lauinger, Abdelberi Chaabane, Sajjad Arshad, William Robertson, Christo Wilson, and Engin Kirda. Thou shalt not depend on me: Analysing the use of outdated javascript libraries on the web. arXiv preprint arXiv:1811.00918, 2018. 20, 22

[174] Arturs Lavrenovs and F Jesús Rubio Melón. Http security headers analysis of top one million websites. In Proc. of the International Conference on Cyber Conflict, 2018. 68

[175] Visual Layer. fastdup v1 api. https://visual-layer.readme.io/docs/v1-api, 05 2024. (Accessed on 09/09/2024). 119

[176] Kiho Lee, Kyungchan Lim, Hyoungshick Kim, Yonghwi Kwon, and Doowon Kim. 7 days later: Analyzing phishing-site lifespan after detected. In THE WEB CONFERENCE 2025, page 1, 2025. 9

[177] Woonghee Lee, Junbeom Hur, and Doowon Kim. Beneath the Phishing Scripts: A Script-Level Analysis of Phishing Kits and Their Impact on Real-World Phishing Websites. In Proc. of the ACM Asia Conference on Computer and Communications Security (AsiaCCS), 2024. 122

[178] Ada Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. 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), 2016. 18

[179] Xigao Li, Babak Amin Azad, Amir Rahmati, and Nick Nikiforakis. Good bot, bad bot: Characterizing automated browsing activity. In Proc. of the IEEE symposium on security and privacy, 2021. 61

[180] Kyungchan Lim, Yonghwi Kwon, and Doowon Kim. A longitudinal study of vulnerable client-side resources and web developers' updating behaviors. In

Proceedings of the 2023 ACM on Internet Measurement Conference, pages 162–180, 2023. 8

[181] Kyungchan Lim, Kiho Lee, Fujiao Ji, Yonghwi Kwon, Hyoungshick Kim, and Doowon Kim. What's in phishers: A longitudinal study of security configurations in phishing websites and kits. In THE WEB CONFERENCE 2025, page 1, 2025. 7, 8

[182] Kyungchan Lim, Kiho Lee, Raffaele Sommese, Mattis Jonker, Ricky Mok, kc claffy, and Doowon Kim. Registration, detection, and deregistration: Analyzing dns abuse for phishing attacks, 2025. 8, 152

[183] Kyungchan Lim, Jaehwan Park, and Doowon Kim. Phishing vs. legit: Comparative analysis of client-side resources of phishing and target brand websites. In Proceedings of the ACM on Web Conference 2024, pages 1756–1767, 2024. 3, 6, 7, 10, 84, 122, 125

[184] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In 30th USENIX Security Symposium (USENIX Security 21), pages 3793–3810, 2021. 3

[185] lokesh. Github lokesh lightbox2: The original lightbox script (v2). https://github.com/lokesh/lightbox2, 02 2023. (Accessed on 09/27/2023). 21

[186] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. The annals of mathematical statistics, pages 50–60, 1947. 143

[187] Sourena Maroofi, Maciej Korczyński, and Andrzej Duda. Are you human? resilience of phishing detection to evasion techniques based on human verification.

In Proceedings of the ACM Internet Measurement Conference, pages 78–86, 2020. 15

[188] Sourena Maroofi, Maciej Korczyński, Cristian Hesselman, Benoit Ampeau, and Andrzej Duda. Comar: classification of compromised versus maliciously registered domains. In 2020 IEEE European Symposium on Security and Privacy (EuroS&P), pages 607–623. IEEE, 2020. 87, 88

[189] Fabian Marquardt and Lennart Buhl. Déjà vu? client-side fingerprinting and version detection of web application software. In 2021 IEEE 46th Conference on Local Computer Networks (LCN), pages 81–89, 2021. 19, 34

[190] matiskay. matiskay/html-similarity: Compare html similarity using structural and style metrics. https://github.com/matiskay/html-similarity, 10 2023. (Accessed on 10/11/2023). 27

[191] D Kevin McGrath and Minaxi Gupta. Behind Phishing: An Examination of Phisher Modi Operandi. LEET, 8:4, 2008. 12

[192] mitchellkrogza. mitchellkrogza/phishing.database: Phishing domains, urls websites and threats database. we use the pyfunceble testing tool to validate the status of all known phishing domains and provide stats to reveal how many unique domains used for phishing are still active. https://github.com/mitchellkrogza/Phishing.Database, 11 2024. (Accessed on 11/05/2024). 84, 85

[193] Modernizr. Modernizr: the feature detection library for html5/css3. https://modernizr.com/, 01 2023. (Accessed on 05/26/2023). 35

[194] Moment. Moment.js — home. https://momentjs.com/, 01 2023. (Accessed on 05/26/2023). 35

[195] Elizabeth Montalbano. Evilproxy commodifies reverse-proxy tactic for phishing, bypassing 2fa. https://www.darkreading.com/vulnerabilities-threats/

evilproxy-commodifies-reverse-proxy-tactic-phishing-bypassing-2fa, 09 2022. (Accessed on 10/11/2023). 26

[196] Mozilla. Subresource integrity - web security — mdn. https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity, 11 2022. (Accessed on 05/26/2023). 48

[197] Mozilla. How does built-in phishing and malware protection work? — firefox help. https://support.mozilla.org/en-US/kb/how-does-phishing-and-malware-protection-work, 06 2023. (Accessed on 09/13/2024). 124

[198] Mozilla. Get firefox for desktop — mozilla (us), 02 2025. [Online; accessed 2025-02-20]. 7

[199] Nginx. Nginx, 02 2025. [Online; accessed 2025-02-20]. 2

[200] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You are what you include: large-scale evaluation of remote javascript inclusions. In Proceedings of the 2012 ACM conference on Computer and communications security, pages 736–747, 2012. 18

[201] NVD. Nvd - cve-2023-25690. https://nvd.nist.gov/vuln/detail/CVE-2023-25690, 09 2024. (Accessed on 09/30/2024). 70

[202] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. Phishfarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists. In 2019 IEEE Symposium on Security and Privacy (SP), pages 1344–1361. IEEE, 2019. 15, 62, 84, 105, 122

[203] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. {PhishTime}: Continuous longitudinal

measurement of the effectiveness of anti-phishing blacklists. In 29th USENIX Security Symposium (USENIX Security 20), pages 379–396, 2020. 9, 15, 62, 84, 105, 116, 122

[204] 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. 15, 122, 125

[205] 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 Proc. of the APWG Symposium on Electronic Crime Research, 2018. 62, 84

[206] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. 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), 2020. 2, 3, 12, 15, 84, 105, 111, 112, 113, 122

[207] openphish. Openphish - openphish database. https://openphish.com/phishing_database.html, 06 2023. (Accessed on 09/14/2024). 84, 85

[208] openphish. Openphish - phishing intelligence. https://openphish.com/, 06 2023. (Accessed on 09/14/2024). 2, 105, 106

[209] OZON. OZON Marketplace. https://www.ozon.ru/, 11 2024. (Accessed on 11/19/2024). 99

[210] Hyunjun Park, Kyungchan Lim, Doowon Kim, Donghyun Yu, and Hyungjoon Koo. Demystifying the regional phishing landscape in south korea. IEEE Access, 11:130131–130143, 2023. 105

[211] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In Proc. of the Internet Measurement Conference (IMC), 2019. 128

[212] phishstats. Phishstats. https://phishstats.info/, 06 2023. (Accessed on 09/14/2024). 105, 106

[213] phishstats. Phishstats. https://phishstats.info/, 11 2024. (Accessed on 11/05/2024). 84, 85

[214] phishtank. Phishtank ¿ friends of phishtank. https://phishtank.org/friends.php, 06 2023. (Accessed on 09/14/2024). 106

[215] phishtank. Phishtank — join the fight against phishing. https://phishtank.org/, 06 2023. (Accessed on 09/14/2024). 105, 106

[216] phishtank. Phishtank — join the fight against phishing. https://phishtank.org/, 11 2024. (Accessed on 11/05/2024). 2, 84, 85

[217] phishunt. phishunt - free phishings and scams feed. https://phishunt.io/, 11 2024. (Accessed on 11/05/2024). 84, 85

[218] Phoronix. Development kicks off with safe browsing, better flatpak handling. https://www.phoronix.com/news/Epiphany-3.27.1-Released, 06 2023. (Accessed on 09/13/2024). 124

[219] Polyfill. Polyfill.io. https://polyfill.io/v3/, 01 2023. (Accessed on 05/26/2023). 35, 37

[220] Popper. Tooltip & popover positioning engine. https://popper.js.org/, 01 2023. (Accessed on 05/26/2023). 35, 37

[221] Prototype. Prototype javascript framework: a foundation for ambitious web applications. http://prototypejs.org/, 09 2015. (Accessed on 05/26/2023). 35

[222] prototype. Prototype javascript framework: a foundation for ambitious web applications. http://prototypejs.org/, 10 2023. (Accessed on 10/10/2023). 21

[223] Nur Aini Rakhmawati, Sayekti Harits, Deny Hermansyah, and Muhammad Ariful Furqon. A survey of web technologies used in indonesia local governments. SISFO Vol 7 No 3, 7, 2018. 19, 34

[224] RDAP. Rdap.org. https://about.rdap.org/, 11 2024. (Accessed on 11/15/2024). 86

[225] RequireJS. Requirejs. https://requirejs.org/, 12 2018. (Accessed on 05/26/2023). 35

[226] Sebastian Roth, Timothy Barron, Stefano Calzavara, Nick Nikiforakis, and Ben Stock. Complex security policy? a longitudinal analysis of deployed content security policies. In Proceedings of the 27th Network and Distributed System Security Symposium (NDSS), 2020. 11, 32, 68, 78

[227] Joe St Sauver. Automating detection of "random-looking" algorithmic domain names - domaintools — start here. know now. https://www.domaintools.com/resources/blog/automating-detection-of-random-looking-algorithmic-domain-names/, 05 2019. (Accessed on 11/12/2024). 89, 90

[228] SCOWL. Scowl custom list/dictionary creator. http://app.aspell.net/create, 11 2024. (Accessed on 11/20/2024). 90

[229] IMQ Minded Security. Imq minded security blog: "jquery mi-grate"' is a sink, too?! https://blog.mindedsecurity.com/2013/04/jquery-migrate-is-sink-too.html, 4 2013. (Accessed on 09/05/2023). 37

[230] Ruchi Sharma, Bhag Dei Thakur, Neelam Kaushik, and Purnima Chauhan. Securing the web: A study on look-alike domain detection using open-source

intelligence tools. Journal of Information Security and Cybercrimes Research, 7(1):05–28, 2024. 89

[231] SpryMedia. Datatables — table plug-in for jquery. https://datatables.net/, 10 2023. (Accessed on 10/10/2023). 21

[232] Statcounter. Browser market share worldwide. https://gs.statcounter.com/browser-market-share/desktop/worldwide, 01 2023. (Accessed on 05/26/2023). 56

[233] Marius Steffens, Marius Musch, Martin Johns, and Ben Stock. Who's hosting the block party? studying third-party blockage of csp and sri. In Network and Distributed Systems Security (NDSS) Symposium 2021, 2021. 32

[234] Ben Stock, Martin Johns, Marius Steffens, and Michael Backes. How the web tangled itself: Uncovering the history of {Client-Side} web ({In) Security}. In 26th USENIX Security Symposium (USENIX Security 17), pages 971–987, 2017. 18

[235] Ben Stock, Martin Johns, Marius Steffens, and Michael Backes. How the web tangled itself: Uncovering the history of Client-Side web (In)Security. In 26th USENIX Security Symposium (USENIX Security 17), pages 971–987, Vancouver, BC, August 2017. USENIX Association. 20, 22, 32

[236] Karthika Subramani, William Melicher, Oleksii Starov, Phani Vadrevu, and Roberto Perdisci. Phishinpatterns: measuring elicited user interactions at scale on phishing websites. In Proceedings of the 22nd ACM Internet Measurement Conference, pages 589–604, 2022. 17

[237] Xiaoqing Sun, Mingkai Tong, Jiahai Yang, Liu Xinran, and Liu Heng. {HinDom}: A robust malicious domain detection system based on heterogeneous information network with transductive classification. In 22nd International Symposium on

Research in Attacks, Intrusions and Defenses (RAID 2019), pages 399–412, 2019. 3

[238] Yuta Takata, Hiroshi Kumagai, and Masaki Kamizono. The uncontrolled web: Measuring security governance on the web. IEICE Transactions on Information and Systems, 104(11):1828–1838, 2021. 19, 34

[239] Bhaskar Tejaswi, Nayanamana Samarasinghe, Sajjad Pourali, Mohammad Mannan, and Amr Youssef. Leaky kits: The increased risk of data exposure from phishing kits. In 2022 APWG Symposium on Electronic Crime Research (eCrime), pages 1–13. IEEE, 2022. 15

[240] Bhaskar Tejaswi, Nayanamana Samarasinghe, Sajjad Pourali, Mohammad Mannan, and Amr Youssef. Leaky kits: The increased risk of data exposure from phishing kits. In Proc. of the APWG Symposium on Electronic Crime Research, 2022. 62, 64

[241] tldlist. Compare prices of all top-level domains — tld-list. https://tld-list.com/, 11 2024. (Accessed on 11/15/2024). 88, 89

[242] Tranco. A research-oriented top sites ranking hardened against manipulation - tranco. https://tranco-list.eu/, 11 2024. (Accessed on 11/20/2024). 88

[243] Tranco. Tranco. https://tranco-list.eu/, 09 2024. (Accessed on 09/19/2024). 66, 68

[244] Underscore. Underscore.js. https://underscorejs.org/, 09 2022. (Accessed on 05/26/2023). 21, 35, 37

[245] USPS. Welcome — usps. https://www.usps.com/, 11 2024. (Accessed on 11/19/2024). 99

[246] Parul Verma and Kevin P Dyer. Detecting phishing attacks in the modern web using content delivery networks. Journal of Cybersecurity Research, 4(2):41–55, 2018. 134

[247] Vorsk. home - dns coffee. https://dns.coffee/, 11 2024. (Accessed on 11/15/2024). 87

[248] W3. Subresource integrity. https://www.w3.org/TR/SRI/#cross-origin-data-leakage, 06 2016. (Accessed on 05/26/2023). 49

[249] W3.org. Html standard. https://html.spec.whatwg.org/multipage/iframe-embed-object.html#the-object-element, 01 2023. (Accessed on 05/26/2023). 57

[250] w3techs. Usage statistics and market share of web servers, october 2024. https://w3techs.com/technologies/overview/web_server, 10 2024. (Accessed on 10/14/2024). 140

[251] Lukas Weichselbaum, Michele Spagnuolo, Sebastian Lekies, and Artur Janc. Csp is dead, long live csp! on the insecurity of whitelists and the future of content security policy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 1376–1387, 2016. 68

[252] Whatwg. Html standard. https://html.spec.whatwg.org/multipage/urls-and-fetching.html#cors-settings-attributes, 01 2023. (Accessed on 05/26/2023). 49

[253] WordPress. Enable jquery migrate helper – wordpress plugin. https://wordpress.org/plugins/enable-jquery-migrate-helper/, 04 2022. (Accessed on 05/26/2023). 36, 39

[254] WordPress. Enable jquery migrate helper – wordpress plugin. https://wordpress.org/plugins/enable-jquery-migrate-helper/#description, 04 2022. (Accessed on 05/26/2023). 37

[255] WordPress. Configuring automatic background updates. https://wordpress.org/support/article/configuring-automatic-background-updates/, 01 2023. (Accessed on 05/26/2023). 53

[256] Qiushi Wu and Kangjie Lu. On the feasibility of stealthily introducing vulnerabilities in open-source software via hypocrite commits. In Proc. Oakland, 2021. 48

[257] ZDNET. Flash version distributed in china after eol is installing adware — zdnet. https://www.zdnet.com/article/flash-version-distributed-in-china-after-eol-is-installing-adware/, 02 2021. (Accessed on 05/26/2023). 56

[258] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In 2021 IEEE Symposium on Security and Privacy (SP), pages 1109–1124. IEEE, 2021. 15, 122, 125

[259] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In Proc. of the IEEE Symposium on Security and Privacy, 2021. 62

[260] Penghui Zhang, Zhibo Sun, Sukwha Kyung, Hans Walter Behrens, Zion Leonahenahe Basque, Haehyun Cho, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, et al. 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, pages 3165–3179, 2022. 15, 122

[261] zloirock. Github zloirock core-js: Standard library. https://github.com/zloirock/core-js, 09 2023. (Accessed on 09/27/2023). 21

# Vita

Kyungchan Lim is a Ph.D. candidate in the Min H. Kao Department of Electrical Engineering and Computer Science at The University of Tennessee, Knoxville (UTK). His research interests focus on Web Security, Phishing Attacks, DNS Abuse, and Measurement Studies. His work has been published in top-tier measurement and web venues, including The Web Conference (WWW), ACM IMC, and WISA. He is the recipient of the Best Student Paper Award at WISA 2022 and multiple travel grant awards from IEEE S&P, ACSAC, and The Web Conference. His research contributes to understanding phishing threats through large-scale data-driven analyses, examining phishing websites' security configurations, DNS exploitation, and the effectiveness of anti-phishing defenses.