What's in Phishers: A Longitudinal Study of Security Configurations in Phishing Websites and Kits

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

Kiho Lee
University of Tennessee
Knoxville, TN, USA
klee120@utk.edu

Fujiao Ji
University of Tennessee
Knoxville, TN, USA
fji1@utk.edu

Yonghwi Kwon
University of Maryland
College Park, MD, USA
yongkwon@umd.edu

Hyoungshick Kim
Sungkyunkwan University
Suwon, South Korea
hyoung@skku.edu

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

Abstract

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. We focus on two main areas: (1) analyzing the security configurations of phishing websites and servers, particularly HTTP headers and application-level security, and (2) examining the prevalence and types of vulnerabilities in phishing kits. We analyze data from 906,731 distinct phishing websites collected over 2.5 years, covering HTML headers, client-side resources, and phishing kits. Our findings suggest that phishing websites often employ weak security configurations, with 88.8% of the 13,344 collected phishing kits containing at least one potential vulnerability, and 12.5% containing backdoor vulnerabilities. These vulnerabilities present an opportunity for defenders to shift from passive defense to active disruption of phishing operations. Our research proposes a new approach to leverage weaknesses in phishing infrastructure, allowing defenders to take proactive actions to disable phishing sites earlier and reduce their effectiveness.

CCS Concepts

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

Keywords

Web Security; Phishing; Phishing Kits

ACM Reference Format:
Kyungchan Lim, Kiho Lee, Fujiao Ji, Yonghwi Kwon, Hyoungshick Kim, and Doowon Kim. 2025. What's in Phishers: A Longitudinal Study of Security Configurations in Phishing Websites and Kits. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696410.3714710

WWW '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1274-6/25/04
https://doi.org/10.1145/3696410.3714710

1 Introduction

Phishing, a type of social engineering attack, has emerged as a major security risk to billions of Internet users [57]. In a typical phishing campaign, attackers craft deceptive phishing websites that masquerade as legitimate websites, such as financial institutions or social media platforms (*e.g.*, PayPal and Facebook), to allure victims and steal sensitive information (*e.g.*, login credentials).

To launch phishing attacks, attackers need to build and host a phishing website. In particular, the building process includes creating client-side (*e.g.*, HTML, JavaScript, and CSS) and server-side resources (*e.g.*, PHP [14]). Attackers can utilize phishing kits that contain pre-configured server-side and client-side resources for phishing websites. Phishing kits have become popular as they significantly ease the development and deployment of phishing websites [28, 32, 36, 54]. Regarding launching phishing websites, attackers may have two options based on how they create them. As the *first* option, attackers also leverage web content publishing services such as website builders (*e.g.*, wix.com [23]) and blogging services (*e.g.*, blogger.com [1]) as they provide a convenient way to create and host web content. As the *second* option, attackers may run standalone web servers running Apache [20], with the pre-configured phishing kits.

Understanding phishing websites is crucial for advancing phishing detection because these sites are central to attackers' efforts to deceive victims and steal sensitive information [48, 49, 51]. As phishing tactics continually evolve, attackers devise new evasion techniques to bypass detection, resulting in a cat-and-mouse game with detection systems. These detection systems often rely on lists of known phishing sites, which, while effective, may not keep up with the latest tactics [43, 59]. To improve detection methods and better protect users, it is essential to deeply understand the phishing ecosystem, including how phishing websites are constructed and maintained. Previous studies have examined various aspects, such as the visual characteristics and use of client-side resources like JavaScript, HTML, and CSS [24, 36, 41, 42, 48, 49, 51], evasion techniques [48, 49, 51], and the role of phishing kits in building these sites [28, 32, 36, 50, 52, 54].

Despite significant research on phishing attacks, the security configurations of phishing websites and servers remain underexplored. To address this gap, we focus on observable artifacts such as (1) identifying vulnerable points in phishing websites by analyzing insecure server configurations, such as missing or improperly set security headers; (2) detecting vulnerabilities within phishing kits that can be exploited; and (3) leveraging these insecure configurations and vulnerabilities to actively exploit phishing kits and disrupt their operations. Analyzing these areas helps us identify common weaknesses that can serve as features for detecting phishing sites. This knowledge also enables us to develop effective tools that not only detect phishing sites but also exploit these identified vulnerabilities to actively disrupt their operations. This approach shifts the paradigm from a purely defensive stance—such as detecting phishing websites and educating users to avoid them—to a proactive one where we actively seek out phishing sites and use their weaknesses to neutralize them [30, 33, 38]. These investigations have contributed to understanding the phishing ecosystem and developing countermeasures against phishing attacks by exploiting a phishing ecosystem.

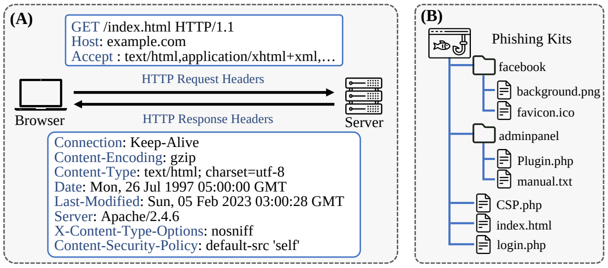
We conduct a systematic analysis of security configurations in phishing servers and the security implications of phishing kits by collecting HTTP headers, screenshots, client-side resources (*e.g.*, HTML, JavaScript, images), and phishing kits. This data is collected between Jul. 2021 and Jan. 2024 (2 years and 7 months) by accessing 16.7 million distinct phishing URLs and refining the dataset. To identify potential exploit points in phishing websites, we begin by analyzing their security configurations, focusing on HTTP headers (*e.g.*, `Content-Security-Policy` (CSP), `Set-Cookie`) and vulnerabilities in web server program versions. Then, we compare our findings with the security configurations of benign websites (Tranco top 10K websites [55]) to accommodate our analysis.

We observe that security-related HTTP headers are barely employed, or even when used, they are often improperly configured in phishing websites. For example, only 5.4% of phishing websites utilize CSP, while most benign websites (75.2%) use it. Moreover, over 98% of the phishing websites, that specify the `Set-Cookie` header, use the improper directive and its value '`Set-Cookie=/`,' allowing all directories to access the cookie. Also, 14.3% of the phishing websites are hosted on web servers with known vulnerabilities (*e.g.*, Apache/2.4.6, PHP/7.4.33). These insecure configurations and unpatched systems present further exploitation opportunities, which can be potential avenues for disrupting phishing operations.

Furthermore, we analyze our collected 13,344 distinct phishing kits (2.68M PHP scripts) using two static analysis tools (`Semgrep` [17] and `progpilot` [19]), identifying a wide range of security issues, including vulnerabilities that can be immediately exploitable (*e.g.*, backdoors) by security entities. Specifically, we find 689,963 Common Weakness Enumerations (CWEs) from 11,853 phishing kits (88.8% out of 13,344). In particular, we find potentially exploitable security weaknesses, such as CWE-79 (Cross-site scripting) [8] and CWE-89 (SQL injection) [9], are commonly observed in our dataset as well as readily exploitable vulnerabilities (or backdoors) in 1,668 (12.5% out of 13,344) phishing kits.

Our contributions are summarized as follows:

- We conduct a longitudinal analysis of the security configurations of the phishing websites and vulnerabilities within phishing kits by investigating our collected dataset of HTTP headers, client-side resources, and phishing kits for 31 months (Jul. 15, 2021, to Jan. 31, 2024).



```
GET /index.html HTTP/1.1
Host: example.com
Accept : text/html,application/xhtml+xml,...
```

```
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Sun, 05 Feb 2023 03:00:28 GMT
Server: Apache/2.4.6
X-Content-Type-Options: nosniff
Content-Security-Policy: default-src 'self'
```

Figure 1: Example of (A) HTTP Header and (B) Phishing Kit.

- We discover that phishing websites have relatively weak security configurations from their HTTP headers (*e.g.*, CSP and Set-Cookie). In particular, we find that self-hosted phishing websites often misconfigure security-related headers.
- We find 689,963 CWEs (Common Weakness Enumerations) and 1,668 backdoors (or readily exploitable vulnerabilities) in our phishing kit dataset. Additionally, our analysis shows that the phishing kit examined in the case study, which was involved in 204 phishing campaigns, can be easily exploited, suggesting inadequate attention to security measures.
- We publicly share our source code and data collection at "https://moa-lab.net/security-configurations-measurement/" to support and advance future research in the community.

2 Background

Phishing attacks represent a sophisticated social engineering attack where cybercriminals fool victims into disclosing sensitive information. In this section, we introduce how a phishing ecosystem can be configured using HTTP headers and phishing kits.

2.1 HTTP Headers and Security Configurations

HTTP Headers. HTTP headers transmit additional information between clients and servers during requests and responses, specifying details about the requested resource or desired behaviors. For example, a request header may include the resource's location (*e.g.*, '`GET /index.html`') and acceptable data formats ('`Accept: text/html`'). Similarly, response headers provide details about the server's response, such as status codes ('`200 OK`'), content types ('`Content-Type: application/json;`'), and content-encoding ('`Content-Encoding: gzip`'). This information, formatted as key-value pairs, helps ensure accurate data handling by the client. Standard header fields are defined in the IANA registry [13].

Security Configuration in HTTP Headers. HTTP headers help enhance web application security, particularly against Cross-site Scripting (XSS) attacks [4], where attackers inject malicious scripts (*e.g.*, JavaScript) into trusted web pages. For end-users, these scripts appear as legitimate content, leading to the execution of malicious code within the trusted site. Once executed, the scripts can access sensitive information, such as session cookies, potentially hijacking sessions, defacing websites, or redirecting users to harmful sites. To prevent XSS attacks, the `CSP` header is used to control the sources of permitted content. For example, the directive "`Content-Security-Policy: default-src 'self'`" restricts content to the local origin, as shown in Figure 1–(A).

Security Configuration in HTML. HyperText Markup Language (HTML) is the standard language for structuring web pages using tags, elements, and attributes. It can also help mitigate web security attacks, such as XSS. For instance, a `Content-Security-Policy (CSP)` can be defined in the HTML '`<head>`' to restrict scripts to the same origin, using a directive like `<meta http-equiv="Content-Security-Policy" content="default-src 'self';">`.

2.2 Phishing Kit

Phishing kits serve as comprehensive toolsets for deploying phishing sites on web servers [45]. While some creators closely guard their kits, others offer them as part of the cybercrime-as-a-service ecosystem [44]. Specialized criminals develop and sell these kits, often accommodating custom requests [29].

As shown in Figure 1–(B), these kits generally include: (1) the template mimicking the legitimate website's resources (*e.g.*, HTML, images, fonts), (2) pre-compromised web servers for capturing and transmitting submitted data, and (3) optional features to filter unwanted traffic or implement anti-detection measures. These help significantly lower entry barriers for attackers, enabling individuals with minimal technical expertise to engage in successful phishing operations. A typical phishing operation involves purchasing a kit, customizing it with a designated email address, uploading and extracting it on a pre-compromised server, and using spam tools to distribute pre-crafted messages to target email lists. The phisher then awaits the influx of stolen credentials.

3 Motivation

The motivation for our research is to identify and exploit weaknesses in phishing infrastructure to actively disrupt and neutralize phishing operations. By analyzing vulnerabilities in phishing websites and kits, we uncover common flaws that can be used not only as indicators for detecting phishing sites but also as entry points for taking direct action against them. This approach moves beyond traditional defensive strategies—such as detecting phishing sites and raising user awareness—toward a more proactive stance. Instead of merely reacting to phishing threats, we aim to leverage the attackers' weaknesses to interfere with their operations, effectively turning the tables and neutralizing the threat at its source. This shift in strategy aims to significantly reduce the impact of phishing attacks by disrupting them before they can exploit victims.

4 Crawler Design & Dataset Collection

As shown in Figure 2, 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, header examination, vulnerability identification (focusing on PHP), and backdoor access investigation.

Table 1: 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;

4.1 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 [2], 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 [26, 40].

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 [32, 54].

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.

4.2 Phishing Data Collection

Collecting Phishing Website Resources. The APWG eCrime Exchange `eCX` [21] 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 [37, 48–50, 60]. Our

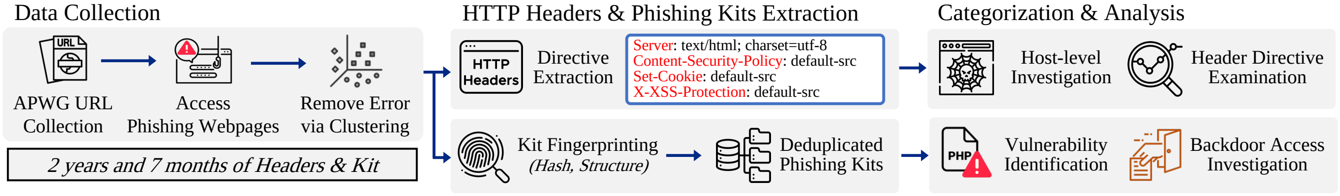

Data Collection
APWG URL Collection
Access Phishing Webpages
Remove Error via Clustering
2 years and 7 months of Headers & Kit
HTTP Headers & Phishing Kits Extraction
HTTP Headers
Directive Extraction
Server: text/html; charset=utf-8
Content-Security-Policy: default-src
Set-Cookie: default-src
X-XSS-Protection: default-src
Kit Fingerprinting (Hash, Structure)
Deduplicated Phishing Kits
Categorization & Analysis
Host-level Investigation
Header Directive Examination
PHP
Vulnerability Identification
Backdoor Access Investigation

Figure 2: Overview of Our Systematic Measurement Study.

crawler is fed real-world phishing URLs from the APWG eCrime Exchange (eCX) in real-time [21]. As illustrated in Figure 2, 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` [25] 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 1.

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) Figure 1–(B). 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 [28, 54]. 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) [28] – undergoes security analysis using `Semgrep` [17] and `progpilot` [19].

5 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 [15] 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 5.

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 2, 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 2 describes the types of attacks, 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 [55], 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 [27] 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.

5.1 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%. Compared to benign websites, the general usage of headers is similar (within a 0.1 margin), except for the `Server` header. In benign websites, only 78.4% use the `Server` header, likely to reduce the risk of information leakage and prevent potential vulnerabilities

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 3 illustrates the usage trends of security-related headers in phishing

Table 2: 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;

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 [35, 39, 53, 56]. Analyzing the top 10K domains from the Tranco 1M list [55], 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%).

Takeaway: Phishing websites exhibit significantly lower security header adoption compared to benign sites, with minimal critical protections. While benign sites use headers like CSP (75.2% vs. 5.4%) and `X-XSS-Protection` (75.8% vs. 31.0%) more frequently, self-hosted phishing servers use `Set-Cookie` extensively (6.5% vs. 53.4%), suggesting advanced user tracking.

5.2 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 `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.

5.3 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 [16] 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` [47] (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.

5.4 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.

Table 3: Top 10 CWEs in Phishing Kits.

ID	Type	LoE*	Severity	`Semgrep`	`Progpilot`
				# Vuln. (%)	# 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

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.

6 Vulnerabilities in Phishing Kits

In this section, we look at CWE vulnerabilities and backdoors to code in phishing kits. We conducted penetration testing to scrutinize the level of vulnerability of phishing kits in a local environment, adhering to ethical considerations.

6.1 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` [17] and `progpilot` [19], 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 3, the top three CWE categories identified by `Semgrep` are CWE-79 [8] (XSS, 73.71%), CWE-89 [9] (SQL Injection, 12.85%), and CWE-918 [10] (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 [5] (Insecure Cookie, 17.19%), and CWE-89 (SQL Injection, 16.38%), representing 72.13% of the total CWEs reported.

Severity of CWEs. To better understand the severity of the CWEs we found, we leverage semgrep's `Likelihood` metric [3], 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 [7] (deserialization of untrusted data) are the two highly severe CWEs. CWE-79 (XSS) and CWE-22 [6] (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.

6.2 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.

6.2.1 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.

6.2.2 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.

```
 1 <?php                                                                  verify.php
 2 if (isset($_POST['signin'])) {
 3       echo "<script>window.location='next.php?
 4             email=".$_GET['email']."&password=".$_POST['password']
 5             ."&verify=0'</script>";} ?>
 6 <?php                                                                  next.php
 7 $message .= "Email: ".$_GET['email']."\n";
 8 $message .= "Password: ".$_GET['password']."\n";
 9 // ... collect location info and user agent ...
10 $fp = fopen('.error.htm', 'a');
11 fwrite($fp, "\n".$message); fclose($fp);
12 // ... send email to attacker ...
13 if ($_GET['verify'] == 0) {
14       echo "<script>window.location='verify.php?
15             email=".$_GET['email']."&error=true&verify=1'</script>";
16 } elseif ($_GET['verify'] == 1) {
17       session_start();
18       echo "<meta http-equiv='refresh' content='0;
19             url=https://login.live.com/'/>"; } ?>
```

Listing 1: Two Examples of XSS Vulnerability.

```
1 <?php require_once('Connections/conn.php');
2 $user = $_POST['username'];
3 $pass = $_POST['password'];
4 $result = mysql_query("SELECT * FROM login WHERE
5 username='$user' AND password='$pass'") or die(mysql_error());
6 $row = mysql_fetch_array( $result );
7 // ... rest of the code ...; ?>
```

Listing 2: Example of SQL Injection Vulnerability.

6.3 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 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` [18] and `VirusShare` [22], 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. The keywords used in this analysis are listed in Table 4. To validate our findings, we activate and penetration-test the phishing kits in a controlled environment, adhering to ethical guidelines.

Figure 4(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(b), this backdoor control panel lacks proper traversal protections, making it easily accessible via subdirectory recursion (*e.g.*, ffuf [11]) 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.

7 Discussion

Limitations. Identifying content publishing services and server owners is difficult when phishing attackers use redirects. To address this, we manually verified 100 phishing websites from content publishing services, comparing provider names from domains and HTML files, and confirming full consistency in the results.

Recommendations.

- *Detection Improvement with Fingerprinting.* Improving phishing detection through fingerprinting misconfigured headers like `CSP` or `Set-Cookie` can enhance accuracy beyond current methods based on visual similarities and suspicious URLs. Since phishing attackers have different characteristics when looking at the security-related headers compared to benign counterparts.
- *Understand the Taken Information.* Identifying and exploiting security vulnerabilities on phishing servers, such as misconfigured headers or flaws in phishing kits, can provide valuable insights into phishing operations. Accessing the servers or databases of confirmed phishing websites allows defenders to analyze the types of data collected, helping to develop targeted defense mechanisms for the most compromised categories (*e.g.*, geo-locations, IP addresses, credit card details, and login credentials). When conducted under legal and ethical guidelines, retrieving critical data can aid in profiling victims and assessing the scope of phishing attacks, ultimately strengthening mitigation efforts.
- *Exploitation Beyond Detection.* Immediate mitigation strategies for benign users (including web servers and/or website administrators) can be as follows. First, a web server system administrator may run web vulnerability scanners [49, 51] that discover and attempt known vulnerabilities such as SQL injection, command injection, and XSS on their benign, legitimate web servers. They can also be configured to scan known web shell backdoors and control panels. Second, a web developer may run static analysis tools to find known signatures (*e.g.*, function names) of those vulnerabilities and backdoors. For instance, `shell_exec()` and `python_exec()` indicate a webshell backdoor.

Ethics. Our methods prioritize ethical considerations while maintaining scientific rigor in analyzing real-world phishing websites and kits. To address ethical concerns, we examine phishing kit backdoors in a controlled local environment rather than on active phishing servers. While this may not fully replicate live server configurations, it ensures a safe and responsible analysis of these malicious tools. Our recommendation – improving phishing detection through fingerprinting misconfigured headers – relies solely on security headers, which do not contain sensitive victim information. However, other parts of HTTP/HTTPS headers (*e.g.*, cookies, and credentials) may include sensitive data and must be handled with care. Any such information, even if obtained from malicious actors, should be disregarded immediately.

Additionally, while identifying and exploiting security vulnerabilities can enhance phishing defense, any such actions targeting malicious actors or infrastructure must be conducted with proper legal authorization and consultation.

8 Related Work

Previous research has largely overlooked the analysis of header information in phishing threats. Earlier studies have focused on understanding HTTP headers in benign websites and their role in mitigating security attacks.

Server Information in Web Headers. Research on security headers has focused on benign websites [31, 34, 39, 46, 58], with limited depth on security-related headers [31, 34]. In contrast, our study provides an in-depth analysis of security headers in real-world phishing attacks.

Phishing Ecosystem. Prior studies [48, 49, 51] explored phishing techniques through controlled experiments, while others [24, 36, 42] focused on mitigation and detection. These works emphasize phishing site structures and evasion tactics. In contrast, we analyze HTTP headers and phishing kits to uncover attackers' security practices.

Phishing Kit. While prior studies [50, 54] analyzed phishing kits' design and attack usage, our approach identifies vulnerabilities and examines their HTTP header configurations. We also uncover backdoors, revealing how attackers maintain control and exfiltrate data. This analysis provides insights into their tactics for sustaining malicious activities and exploiting victims.

9 Conclusion

Our measurement study provides crucial insights into the security configurations of real-world phishing websites and the vulnerabilities within phishing kits for 31 months. Despite the prevalence of HTTP headers, our findings reveal that they are frequently underutilized or incorrectly configured on phishing websites, highlighting substantial deficiencies in security practices. Our in-depth analysis of 13,344 phishing kits using advanced static analyzers uncovered a high number of vulnerabilities, with almost 90% of the kits exhibiting multiple CWE vulnerabilities.

Acknowledgments

We sincerely thank the anonymous shepherd and all the reviewers for their constructive comments and suggestions. This work is supported in part by the NSF (2210137, 2335798, 2426653, and 2427783), a seed grant from the AI Tennessee Initiative at the University of Tennessee Knoxville, Science Alliance's StART program, gifts from Google exploreCSR, and IITP grants from South Korean government (RS-2024-00439762 and RS-2024-00419073). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] 2024. Blogger: Posts. https://www.blogger.com/. (Accessed on 09/07/2024).

[2] 2024. ChromeDriver. https://chromedriver.chromium.org. (Accessed on 09/13/2024).

[3] 2024. Contributing rules | Semgrep. https://semgrep.dev/docs/contributing/contributing-to-semgrep-rules-repository. (Accessed on 05/15/2024).

[4] 2024. Cross Site Scripting (XSS) | OWASP Foundation. https://owasp.org/www-community/attacks/xss/. (Accessed on 09/07/2024).

[5] 2024. CWE - CWE-1004: Sensitive Cookie Without 'HttpOnly' Flag. https://cwe.mitre.org/data/definitions/1004.html. (Accessed on 09/14/2024).

[6] 2024. CWE - CWE-22: Improper Limitation of a Pathname to a Restricted Directory. https://cwe.mitre.org/data/definitions/22.html. (Accessed on 09/14/2024).

[7] 2024. CWE - CWE-502: Deserialization of Untrusted Data. https://cwe.mitre.org/data/definitions/502.html. (Accessed on 09/14/2024).

[8] 2024. CWE - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'). https://cwe.mitre.org/data/definitions/79.html. (Accessed on 09/14/2024).

[9] 2024. CWE - CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'). https://cwe.mitre.org/data/definitions/89.html. (Accessed on 09/14/2024).

[10] 2024. CWE - CWE-918: Server-Side Request Forgery (SSRF). https://cwe.mitre.org/data/definitions/918.html. (Accessed on 09/14/2024).

[11] 2024. Fast web fuzzer. https://github.com/ffuf/ffuf (Accessed on 09/19/2024).

[12] 2024. HTTP headers - HTTP | MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers. (Accessed on 09/07/2024).

[13] 2024. IANA Message Headers. https://www.iana.org/assignments/message-headers/message-headers.xhtml. (Accessed on 09/19/2024).

[14] 2024. PHP. https://www.php.net/. (Accessed on 09/13/2024).

[15] 2024. publicsuffix/list: The Public Suffix List. https://github.com/publicsuffix/list?tab=readme-ov-file. (Accessed on 09/18/2024).

[16] 2024. RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. https://datatracker.ietf.org/doc/html/rfc2616#section-15.1.1. (Accessed on 09/08/2024).

[17] 2024. Semgrep. https://semgrep.dev/ (Accessed on 09/06/2024).

[18] 2024. shellray. https://shellray.com/. (Accessed on 09/13/2024).

[19] 2024. Stegosploit. https://stegosploit.info/ (Accessed on 09/06/2024).

[20] 2024. The Apache HTTP Server Project. https://httpd.apache.org/. (Accessed on 09/07/2024).

[21] 2024. The APWG eCrime Exchange (eCX). https://apwg.org/ecx/. (Accessed on 09/13/2024).

[22] 2024. VirusShare.com. https://virusshare.com/. (Accessed on 09/13/2024).

[23] 2024. Wix.com. https://wix.com/. (Accessed on 09/07/2024).

[24] Zainab Alkhalil, Chaminda Hewage, Liqaa Nawaf, and Imtiaz Khan. 2021. Phishing attacks: A recent comprehensive study and a new anatomy. *Frontiers in Computer Science* (2021).

[25] Amir Alush, Dickson Neoh, and Danny Bickson et al. 2024. Fastdup. GitHub.Note: https://github.com/visuallayer/fastdup. (Accessed on 09/13/2024).

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

[27] Archive. 2024. Wayback Machine. https://web.archive.org/ [Online; accessed 2024-08-30].

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

[29] Dominik Birk, Sebastian Gajek, Felix Grobert, and Ahmad-Reza Sadeghi. 2007. Phishing phishers-observing and tracing organized cybercrime. In *Second International Conference on Internet Monitoring and Protection (ICIMP 2007).* IEEE.

[30] Juan Caballero, Pongsin Poosankam, Stephen McCamant, Domagoj Babi ć, and Dawn Song. 2010. Input generation via decomposition and re-stitching: Finding bugs in malware. In *Proceedings of the 17th ACM conference on Computer and communications security.* 413–425.

[31] Maria Carla Calzarossa and Luisa Massari. 2014. Analysis of header usage patterns of HTTP request messages. In *Proc. of the IEEE Intl Conf on High Performance Computing and Communications.* IEEE.

[32] Marco Cova, Christopher Kruegel, and Giovanni Vigna. 2008. There Is No Free Phish: An Analysis of" Free" and Live Phishing Kits. *Proc. of the WOOT* (2008).

[33] Birhanu Eshete, Abeer Alhuzali, Maliheh Monshizadeh, Phillip A Porras, Venkat N Venkatakrishnan, and Vinod Yegneswaran. 2015. EKHunter: A Counter-Offensive Toolkit for Exploit Kit Infiltration.. In *NDSS.*

[34] Pascal Gadient, Oscar Nierstrasz, and Mohammad Ghafari. 2021. Security header fields in http clients. In *Proc. of the IEEE International Conference on Software Quality, Reliability and Security.*

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

[36] Xiao Han, Nizar Kheir, and Davide Balzarotti. 2016. Phisheye: Live monitoring of sandboxed phishing kits. In *Proc. of the 2016 ACM SIGSAC Conference on Computer and Communications Security.*

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

[38] Paul Knickerbocker, Dongting Yu, and Jun Li. 2009. Humboldt: A distributed phishing disruption system. In *2009 eCrime Researchers Summit.* IEEE, 1–12.

[39] Arturs Lavrenovs and F Jesús Rubio Melón. 2018. HTTP security headers analysis of top one million websites. In *Proc. of the International Conference on Cyber Conflict.*

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

[41] Kyungchan Lim, Jaehwan Park, and Doowon Kim. 2024. Phishing Vs. Legit: Comparative Analysis of Client-Side Resources of Phishing and Target Brand Websites. In *Proceedings of the ACM on Web Conference 2024.* 1756–1767.

[42] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *Proc. of the USENIX Security Symposium.*

[43] Ruofan Liu, Yun Lin, Xiwen Teoh, Gongshen Liu, Zhiyong Huang, and Jin Song Dong. 2019. Less Defined Knowledge and More True Alarms: Reference-based Phishing Detection without a Pre-defined Reference List. 7 (2019).

[44] Derek Manky. 2013. Cybercrime as a service: a very modern business. *Computer Fraud & Security* 2013, 6 (2013), 9–13.

[45] Heather McCalley, Brad Wardman, and Gary Warner. 2011. Analysis of back-doored phishing kits. In *Advances in Digital Forensics VII: 7th IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 31–February 2, 2011, Revised Selected Papers 7.* Springer, 155–168.

[46] Abner Mendoza, Phakpoom Chinprutthiwong, and Guofei Gu. 2018. Uncovering HTTP header inconsistencies and the impact on desktop/mobile websites. In *Proc. of the International World Wide Web Conference.*

[47] NVD. 2024. NVD - CVE-2023-25690. https://nvd.nist.gov/vuln/detail/CVE-2023-25690. (Accessed on 09/30/2024).

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

[49] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. 2020. {PhishTime}: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *Proc. of the USENIX Security Symposium.*

[50] Adam Oest, Yeganeh Safei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Gary Warner. 2018. Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *Proc. of the APWG Symposium on Electronic Crime Research.*

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

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

[53] Sebastian Roth, Timothy Barron, Stefano Calzavara, Nick Nikiforakis, and Ben Stock. 2020. Complex security policy? a longitudinal analysis of deployed content security policies. In *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS).*

[54] Bhaskar Tejaswi, Nayanamana Samarasinghe, Sajjad Pourali, Mohammad Mannan, and Amr Youssef. 2022. Leaky kits: The increased risk of data exposure from phishing kits. In *Proc. of the APWG Symposium on Electronic Crime Research.*

[55] Tranco. 2024. Tranco. https://tranco-list.eu/. (Accessed on 09/19/2024).

[56] Lukas Weichselbaum, Michele Spagnuolo, Sebastian Lekies, and Artur Janc. 2016. 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.* 1376–1387.

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

[58] Craig E Wills and Mikhail Mikhailov. 1999. Towards a better understanding of web resources and server responses for improved caching. (1999).

[59] Peng Yang, Guangzhen Zhao, and Peng Zeng. 2019. Phishing website detection based on multidimensional features driven by deep learning. *IEEE access* 7 (2019).

[60] 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. 2021. CrawlPhish: Large-scale Analysis of Client-side Cloaking Techniques in Phishing. In *Proc. of the IEEE Symposium on Security and Privacy.*

A 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 [53]. 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.

B 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 Listing 3.

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

Listing 3: Example of Header Manipulation in PHP.

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

Table 4: Web Shell & System Access Keywords in Kits.

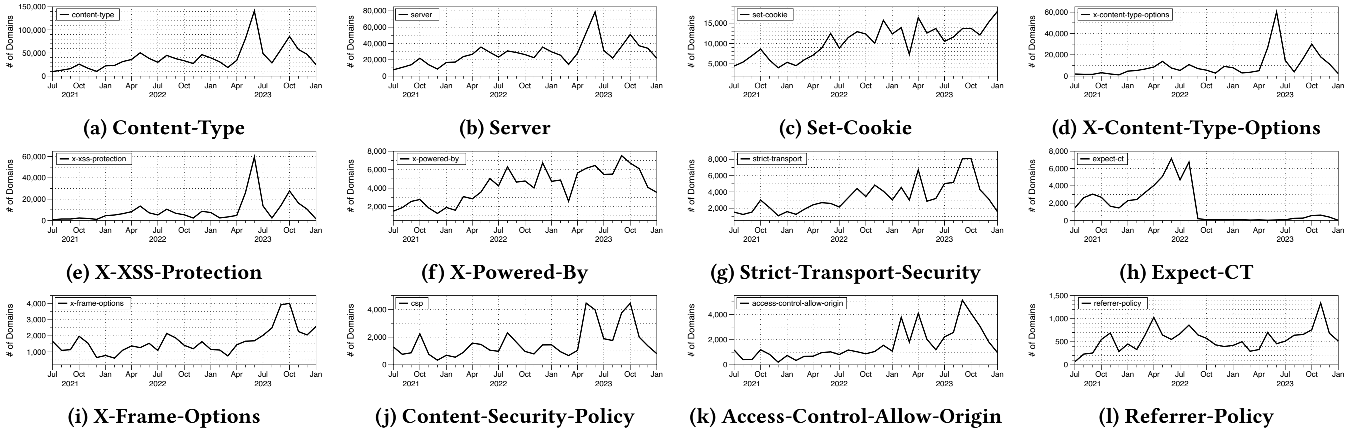
Type	Keyword Value
Shell	"angel"; "b374k"; "bv7binary"; "c99"; "c100"; "r57"; "webroot"; "kacak"; "symlink"; "h4cker"; "webadmin"; "gazashell"; "locus7shell"; "syrianshell"; "injection"; "cyberwarrior"; "ernebypass"; "g6shell"; "pouyaserver"; "saudishell"; "simattacker"; "sosyeteshell"; "tryagshell"; "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";

C Phishing CPanel and Backdoor in Kits

Figure 4 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(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(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 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 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."

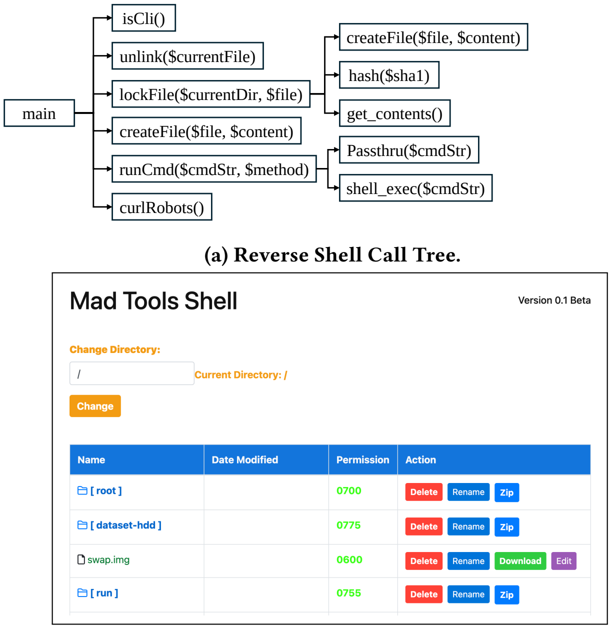


(a) Content-Type (b) Server (c) Set-Cookie (d) X-Content-Type-Options

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

(i) X-Frame-Options (j) Content-Security-Policy (k) Access-Control-Allow-Origin (l) Referrer-Policy

Figure 3: Top 12 Security-related Headers Usage



(a) Reverse Shell Call Tree.

Mad Tools Shell

Version 0.1 Beta

Change Directory:

/ Current Directory: /

Change

Name	Date Modified	Permission	Action
[root]		0700	Delete Rename Zip
[dataset-hdd]		0775	Delete Rename Zip
swap.img		0600	Delete Rename Download Edit
[run]		0755	Delete Rename Zip

(b) Example of Phishing Campaign Control Panel.

Figure 4: Phishing Control Panel and Backdoor Call Tree.

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.

D PHP Code Analysis of a Phishing Kit

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.

```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.

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.

Table 5: 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 [12]; Insecure HTTP headers are highlighted in Red.