Referrer Policy: Implementation and Circumvention

Luqman Muhammad Zagi
Radboud University
Nijmegen, The Netherlands
luqman.zagi@ru.nl

Zahra Moti
Radboud University
Nijmegen, The Netherlands
zahra.moti@ru.nl

Gunes Acar
Radboud University
Nijmegen, The Netherlands
g.acar@cs.ru.nl

ABSTRACT

The Referrer Policy (RP) standard makes it possible for websites to control how much information will be shared in the Referer *[sic]* header. In this study, we investigate the implementation and circumvention of the Referrer Policy standard across 27,750 distinct websites and over 100K pages from three vantage points: the United States, Singapore and the Netherlands. Our findings reveal that 48.38% of websites implement document-wide referrer policies, and 13.39% apply element-specific referrer policies. The majority of the sites (43.81%) use the Referrer-Policy HTTP response header to set a document-wide policy, while 11.09% use HTML meta tags. Even on websites with restrictive referrer policies, scripts can access the full page URL and exfiltrate it — which we label as a *referrer policy circumvention*. We identified RP circumventions on 77.20% of websites often carried out by third-party advertising and analytics scripts, including Google Analytics, Facebook, and TikTok Pixel. While the ability to manage referrer information and the adoption of more privacy-focused default policies represent positive gains for user privacy, the widespread circumvention of these measures by third-party script remains to be a problem. We recommend implementing technical measures to restrict script access in order to address this privacy and security issue.

KEYWORDS

Referer, Referrer Policy, Online tracking, Privacy, Circumvention

1 INTRODUCTION

Modern webpages increasingly incorporate third-party content, including content delivery networks (CDNs), analytics services and advertising networks to monetize their content, offer new functionalities and deliver content efficiently [69]. The Referer *[sic]* header[1], an HTTP header that enables the browser to capture and transmit the URLs of referring pages to servers, has become an essential component for facilitating this integration. The referrer allows websites to monitor referral traffic for analytics and debugging purposes. It also helps website owners to block traffic from unknown sources. The referrer may include an empty string, an origin, or a full URL, but it omits *fragments* and *userinfo* subcomponent (e.g., now deprecated *username:password* format) [19, 38]. While the header enables functionalities such as analytics, logging and caching [58], it also raises significant privacy concerns. The header may disclose sensitive information such as health conditions and order confirmations [48, 64], without users' awareness. Moreover, the referrer can reveal a user's name, gender, address, and passport number to third parties embedded on a website [28, 48, 51, 64]. The header can also reveal online behaviors and preferences. For instance, specific products being browsed or news articles being viewed can easily be extracted from the Referer header.

The Referer header can also reveal privileged *capability* or *telltale* URLs [1, 50]. These URLs may contain sensitive information themselves or act as a gateway to manage an account or order. Websites including Emirates, Lufthansa, GrubHub and Spotify were found to contain telltale URLs [64]. In a particular example, the trainline.eu website inadvertently leaked unique user tokens to grant direct access to booking details through the Referer header [50]. This allowed third-party entities to access sensitive booking details without requiring additional login credentials. Similar referrer leaks can also be used to take over user accounts when password reset tokens are included in the URLs. When users request a password reset, the reset token may be sent in the referrer to third parties—if the token is included in the reset link. A malicious third party can then use the token to reset the password, gaining unauthorized access to the account. In 2021, this very vulnerability was discovered on UPchieve, an online tutoring and counseling site, whose users likely contain many high school students [3].

The "Referrer Policy" (abbreviated as **RP**), introduced as candidate standard in 2017, aims to limit the amount of URL information sent to third-parties via the Referer header when retrieving the subresources, prefetching, or navigation [34]. The policy allows a website to determine what part of the URL should be sent under different conditions, such as cross-origin or insecure requests. For instance, to completely omit the Referer header, a website may set its RP to `no-referrer`. The RP may also be set for individual elements such as anchors (`a`) or images (`img`), which may either trigger requests when clicked or loaded as a subresource.

Since November 2020, many browsers have adopted a more privacy-focused default RP, limiting cross-origin request referrers to the origin only, rather than the full page URL [24]. While this shift offers clear privacy benefits, the effectiveness of RP is challenged by the complexity of modern web applications. Third-party advertising and analytics scripts can often bypass the website owner's preferred RP [2], accessing and transmitting the full page URL to their servers. To the best of our knowledge there has not been an empirical study focusing on the circumvention of the RP standard.

Our study has two primary objectives: (1) to assess the prevalence of RP usage on the web, and (2) to identify instances of RP *circumventions*—cases where a complete URL (or its components) is shared despite a restrictive policy. To achieve the first goal, we develop a custom crawler that collects HTTP headers, POST bodies,

[1]The header was misspelled "Referer" in the early HTTP specification. We use the misspelled version only when referring to the header. [26].

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–21

https://doi.org/XXXXXXX.XXXXXXX

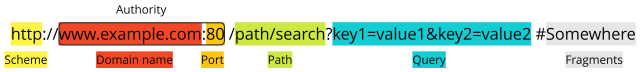

Figure 1: The URL components.

HTML source and JavaScript property accesses to `location.href` and `document.url` — two host object properties that reveal the current document's URL. To detect RP circumventions, we identify cases where the page URL or its components were sent in a third-party request in circumvention of the active RP. Determining the active RP for each request is a complex process. We use the RP provided for each request by the Chrome DevTools Protocol [42], which our crawler uses under the hood.

Circumvention, in this context, does not refer to third parties altering the Referer header. Rather, third parties typically transmit the webpage URL through various vectors, such as in the POST body, either with or without encoding. As a result, while the Referer header itself may not directly violate the RP, the URL is still transmitted using alternative methods.

In summary, we make the following contributions

- We measure the implementation of RP on more than 27K websites using vantage points from three continents (North America, Asia, and Europe).
- We study how websites implement RP, by analyzing HTTP headers, `meta` element attributes, element-specific policies and relational attributes (`rel`).
- We develop a method to detect and measure *RP circumventions*, building on a leak detection method from prior work [36, 65].

2 BACKGROUND AND RELATED WORK

In this section, we present technical preliminaries and related past research.

2.1 Uniform Resource Locator (URL)

URL represents a subset of the Uniform Resource Identifier (URI) that points to a specific resource on the World Wide Web [25]. A URL consists of the following components, depicted in Figure 1 [19, 25]:

- **Scheme:** Specifies the protocol employed for communication, such as HTTP, HTTPS, or FTP.
- **Authority:** An optional component, denoted by a double slash at the beginning and terminated by a slash, question mark, or hash symbol. It may consist of three sub-components: (a) user info (omitted from Figure 1 for simplicity), (b) host, and (c) port.
- **Path:** Specifies the location of a resource on a web server. Historically, it corresponded to a physical file location, but modern web servers often treat paths as abstract identifiers without direct physical mapping.
- **Query:** Contains additional parameters passed to the server, typically used for filtering or customization.
- **Fragments:** Points to a specific part of a resource, usually for navigation within a webpage, and is denoted by a hash symbol (#).

2.2 Referrer

MDN Web Docs[2] describes the Referer header as "an HTTP header, containing either the full or partial URL that denotes the source from which a specific resource has been requested" [11]. For example, when a user clicks a link to a target site from a source site, the target site may receive the URL of the source site through the Referer header in the HTTP request. This helps the target site identify where the traffic is coming from, which can be useful for tracking referrals, behavioral profiling and logging [58]. The referrer information is sent by default in each HTTP request and can be accessed via JavaScript using the `document.referrer` property.

While RFC 9110 [38] specifies that the Referer header should exclude sensitive information such as URL fragments and the userinfo subcomponent [19], it may still contain privacy-sensitive information about a user. To mitigate these concerns, the World Wide Web Consortium (W3C) has introduced the Referrer-Policy standard [30], allowing website owners to control the level of detail shared in the Referer header.

2.3 Referrer Policy (RP)

Referrer Policy (RP) governs the extent of data sent in the Referer header [24, 30]. The header may contain no information, the origin (*scheme* and *authority*) or the full URL (*scheme*, *autority*, *path*, and *query*). The exact data sent in the referrer is determined by factors such as the origin of the request (same-origin or cross-origin) and the security protocol (HTTPS or HTTP). RP can take a total of eight policy values. These policies and the associated behavior are displayed in Table 1. In November 2020, the default policy in the standard was changed from `no-referrer-when-downgrade` to `strict-origin-when-cross-origin` [24]. Since this change, conforming browsers only send the page's origin in cross-origin requests, unless a more permissive RP is specified. While the RP standard defined eight possible values, Firefox and Safari do not support the most permissive (insecure) three policies and upgrade them to the default policy instead. A summary of the supported RP values for each major browser is given in Table 2.

Implementation. The RP can be specified in five different ways, as outlined below [30]. Example code for each method is provided in Listing 1. When multiple RPs are encountered at different layers, the precedence rules given in Figure 2 apply. In short, browsers prioritize element-specific policies over document-wide policies. Moreover, documents (e.g., frames) inherit their RPs from their parent documents.

- **Referrer-Policy Headers:** The `Referrer-Policy` HTTP response header defines how a user agent should handle referrer information when making requests or creating new, document-wide browsing contexts.
- **HTML `Meta`-tags:** A dedicated `<meta>` element with a `name` attribute set to `referrer` and the desired policy value in the `content` attribute determines the default behavior for the entire document.
- **Element-Specific Referrer Policies:** Individual elements such as anchors (`a`), images (`img`), inline frames (`iframe`) or scripts (`script`) can have their own RP using the `referrerpolicy` attribute.

[2]Formerly known as Mozilla Developer Network.

Table 1: The referrer policy's effect on what part of the URL is sent in the referrer under different conditions. *None*: empty referrer; *Origin*: only the origin is sent; *Full*: all URL parts except fragments and user information are sent.

Referrer Policy	same origin	cross origin	HTTPS to HTTP
no-referrer	None	None	None
same-origin	Full	None	None
strict-origin	Origin	Origin	None
strict-origin-when-cross-origin	Full	Origin	None
origin	Origin	Origin	Origin
origin-when-cross-origin	Full	Origin	Origin
no-referrer-when-downgrade	Full	Full	None
unsafe-url	Full	Full	Full

Table 2: Referrer policy types supported by browsers. Safari and Firefox intentionally upgrade the permissive policies listed under *Ignores* to the more private default policy. Safari only sends the origin in the referrer in cross-site requests [12].

Browser	Notes
Chrome	Supports all types
Edge	Supports all types
Opera	Supports all types
Safari	Cross-site: origin Ignores: unsafe-url, origin-when-cross-origin no-referrer-when-downgrade
Firefox	Ignores: unsafe-url, origin-when-cross-origin, no-referrer-when-downgrade

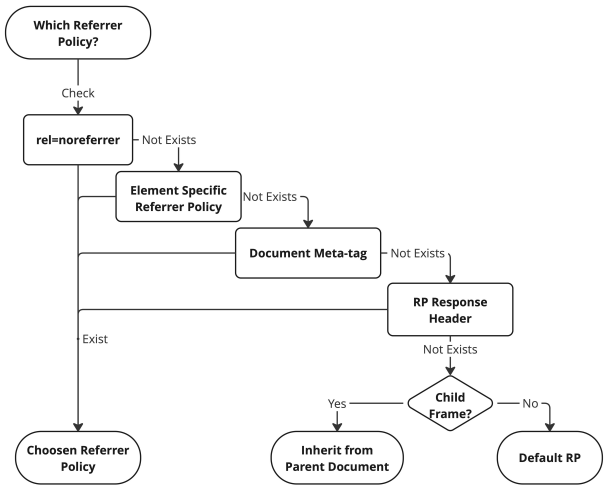

Figure 2: Following the HTML standard, this flowchart shows how browsers choose a referrer policy for fetching resources when multiple, potentially conflicting policies are present [44].

```
<meta name="referrer" content="origin" />

<a href="http://example.com" referrerpolicy="origin"></a>

<a href="http://example.com" rel="noreferrer"> </a>
```

Listing 1: Referrer policy implementation using HTML elements and attributes. Using <meta> offers an alternative to setting a document-wide policy using the *Referrer-Policy* HTTP response header. The other two methods can be used to set element-specific policies.

- **Element-Specific Link Relations (`rel`):** The `rel=` "`noreferrer`" *link relation* on anchor (`a`) or `area` elements act as a shortcut to set their policy explicitly to `no-referrer`.
- **Inheritance:** Referrer policy is inherited according to the policy container inheritance mechanism defined in the HTML standard [43].

Legacy referrer policies. Now deprecated legacy referrer policies such as `origin` and `none-when-downgrade` are defined in the first draft of the RP standard, released in 2014 [35]. These policies are automatically transformed to their modern counterparts by modern browsers (Appendix A.1). We apply the same transformation to legacy RPs we encounter in our measurements — which is a rare occurrence.

2.4 Third-Partyness

The "third-party" requests commonly refer to requests that have a different eTLD+1 than the embedding page [55, 67]. In this study, we define third-party requests based on Kats et al.'s [48] notion of "ideal" third-party relationships. Using DuckDuckGo's Tracker-Radar entity -> domains map [33], we label a request as a third party if it belongs to a different entity (e.g., company or institution) and has a different eTLD+1. For example, requests to google-analytics.com are considered first-party on google.de, since both domains belong to Google/Alphabet. If a domain is not present in DuckDuckGo's entity map, we use the domain's eTLD+1 as its entity.

2.5 Related work

RP Implementations. Lavrenovs and Melon's 2018 study on HTTP security headers on one million websites [52] revealed low adoption of the RP header. Only 0.05% of HTTP responses and 0.33% of HTTPS responses included a RP header. This limited adoption is likely due to the RP specification being relatively new at the time, introduced in January 2017. While this study provided early insights into the adoption of RP, it did not consider HTML-based RP implementations and did not investigate RP circumventions.

Gadient et al. [39] extended the analysis to mobile app communication by examining 9,714 requests with unique URLs from 3,376 apps. Their findings revealed that only 7% of server responses included any security-related headers, with a mere 2% containing the RP header.

Prior studies [39, 52] on measuring the RP adoption have been limited to HTTP header-based RP implementations, neglecting HTML-based implementations. We present a more comprehensive and accurate measurement of RP usage by including HTML-based RP implementations at both document and element level.

Other research has studied RP implementations on mobile browsers and their effectiveness in online collaboration services. In a 2019 study, Luo et al. [57] identified that several popular mobile browsers (e.g., Chrome and Opera Mini) lacked full implementation of the RP standard. According to the web feature support website caniuse.com [29], all mobile browsers except Opera Mini have added RP support in the meantime.

Leakage via the Referer header. Kaleli et al. [47] studied online collaboration services where improper RP implementations and insufficient browser support led to the leakage of secret URLs via the Referer header. A secret URL is a unique link that grants access to a specific document, commonly used for facilitating easy access. Their study revealed that if a file contains a link to certain websites (e.g., an attacker's website), these services inadvertently leaked the secret URL via the Referer header to the attacker's site, compromising the confidentiality of the shared document. Their study underscored that both implementation and browser compatibility are required for effective referrer control. While their work highlighted the specific risk of secret link exposures due to insecure referrer handling, it concentrates on a narrow use case—online collaboration portals—rather than the broader web ecosystem.

Investigating sign-up flows of e-commerce websites, a 2021 study by Dao and Fukuda [28] revealed that 42% of the 307 shopping websites expose personal identifiable information (PII) to at least one third-party domain. The information was exposed through the Referer header on 2.3% of these websites, by request URL on 90.8%, payload body on 33.1%, and a cookie on 4%. They also found that most PII leaks were encoded, with only 32.3% transmitted in plaintext. The leaked data comprises email addresses, usernames, and real names. While the research revealed personal information leaks through the Referer header, it did not examine the impact of RP on reducing these leaks.

A 2022 research by Kats et al. [48] has investigated information leakage due to internal site search functionality. In their study, the researchers discovered that 81.3% of these websites inadvertently disclosed the search terms to third parties. In most (75.8%) cases, the search terms were leaked via the Referer header.

HTTP Headers. Several studies have assessed the implementation and impact of HTTP headers. Siewert et al. [66] studied the behavior of modern browsers when confronted with duplicate, conflicting, or misspelled security headers and directives. McGahagan et al. [59] ranked the significance of HTTP headers in detecting malicious websites. Lerenovs and Visky [53] showed how a single HTTP header can often identify the class or model of specific device connected to the Internet.

While numerous online tools can parse HTTP headers, we identified two useful resources for understanding RP implementations. Outside academic research, Crawler Ninja [45] provides historical measurements of various security headers, including the RP. Chrome Platform Status [40], on the other hand, offers aggregate historical usage data on specific web features based on Google Chrome telemetry data and HTTP Archive crawls [17].

Different than prior work, we take a broader approach to examining the RP, collecting data from different vantage points in the US, Europe, and Asia to offer a global perspective. We perform a large-scale measurement of how widely various websites adopt RP directives (via headers, meta tags, or element attributes) and whether these measures are subverted in practice by third-party scripts. Our findings reveal that even correctly specified RPs can be bypassed by scripts that access and transmit the full page URL, negating the intended privacy protections. Revealing these shortcomings can provide guidance for future privacy-focused web standards, inform policy making and motivate technical countermeasures.

3 METHODS

Below, we describe our methods, including the selection of the target websites, crawls, and data analysis.

3.1 Website list

In order to build a list of websites to study, we utilized the Tranco top 1 million list [54] [3]. Tranco uses a transparent ranking method, openly sharing its data sourcing, filtering, and evaluation processes. The Tranco list also offers localization and customization options with various criteria [54]. We used the October 2023 version of the Tranco list and applied two filters: retaining only pay-level domains (eTLD+1) and including only domains present in the global dataset of the Chrome User Experience Report [41]. Due to study limitations, we (randomly) sampled 30,000 websites from this list.

Website Categorization. Website categorization allows us to understand the nature and characteristics of the websites involved by grouping them into similar categories based on shared attributes. Each category encompasses websites with common content and functionalities, allowing us to compare RP implementations and circumventions within and across different categories.

The classification of websites depends on the tools and methods employed. In this study, we utilized the McAfee SiteLookup service [7], which leverages McAfee Global Threat Intelligence categories to classify websites into predefined groups, including e-commerce, news, social media, finance and entertainment.

While domain categorization services are known to have inconsistencies, as highlighted by Vallina et al. [70], McAfee was found to offer the most comprehensive coverage, making it a suitable choice for our study.

3.2 Crawls

To measure RP usage and circumventions across websites, we extended the open-source crawler Tracker Radar Collector (TRC) [32]. TRC is a modular web crawler designed for large-scale web measurements and has been used by several prior measurement studies [18, 68]. Built on top of Puppeteer and leveraging the Chrome DevTools Protocol (CDP), TRC can collect data about network traffic, JavaScript execution and web elements. We enhance TRC by adding custom modules to collect data specific to our research objectives. The data collection is conducted in two steps: 1) collecting inner page links and 2) measuring Referrer-Policy implementations and circumventions.

3.2.1 Step 1: Collecting Inner Page Links. We developed and added a `LinkCollector` module to the crawler to retrieve inner page URLs from each target website. Inner page URLs containing both a hostname and a path provide more detailed insights into referrer leaks than homepage URLs (e.g., https://example.com/path). Homepages

[3] Available at http://tranco-list.eu/list/7PL6X

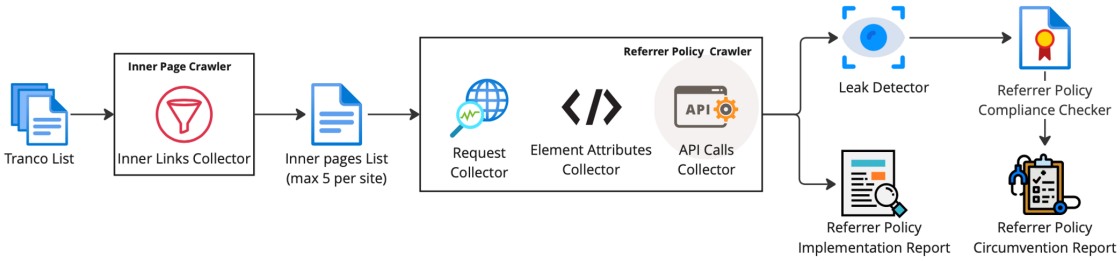

Figure 3: An overview of the data collection and analysis pipeline. The Inner Page and Referrer Policy Crawlers are based on DuckDuckGo's Tracker Radar Collector [32].

typically lack a path component (e.g., https://www.example.com), making it impossible to determine whether a referrer is leaking only the origin or the full URL. Therefore, detecting potential full URL or path leaks is more feasible on inner pages than on homepages. We applied a series of filtering rules to enhance the accuracy and relevance of the data collected. The `LinkCollector`, following an approach similar to that used by Moti et al. [61], prioritizes links near the center of the viewport to avoid links from footers or other less prominent areas. The goal of this approach is to prioritize links that users are more likely to click. Then, the crawler filters links based on several criteria: it excludes links to external domains, links with common file extensions (e.g., `'.jpg'`, `'.jpeg'`, `'.pdf'`, `'.png'`, and `'.xml'`), links without a path or parameters, and `mailto:` links. These filtering rules helped collect links to inner web pages, rather than documents or images.

3.2.2 Step 2: Measuring Referrer-Policy Implementations and Circumventions . Recall that RPs can be set through HTTP headers, a dedicated HTML meta-tag or element-specific attributes (§2.3). To capture the necessary data for studying RP usage, we use multiple collectors. In order to capture HTML-based implementations, we develop a new collector called `RPAttributeCollector`. This collector gathers RP-related attributes from various HTML elements such as links, images, iframes and scripts. The `RPAttributeCollector` identifies and extracts relevant attributes from each of these elements, including `href`, `src`, `title`, `text`, and `rel`, as well as capturing the attributes of the `<meta name="referrer">` tag. Notably, the collector processes both the main frame and any nested child frames. This allows for analysis of RP implementations at both the document and element levels while also considering cross-origin frames.

Additionally, we modify the TRC's `requestCollector` module to capture the Referer and Referrer-Policy headers. For each request, we also save the `referrerPolicy` property, which is provided by the Chrome DevTools Protocol (CDP) to indicate the specific RP that the browser uses for this request [42]. Note that this property is not an actual header and is not sent over the wire. It rather indicates the RP that the browser computed for each request after considering all document-wide and element-specific policies. We capture and use the `referrerPolicy` property to detect RP circumventions. We further extended the `requestCollector` to record HTTP POST request bodies in addition to request headers. We also added support for additional HTTP methods, such as PUT and PATCH. Capturing the POST request bodies and other request types enabled a more comprehensive search for RP circumventions.

TRC's JavaScript instrumentation (`apiCollector`) uses CDP to set conditional breakpoints that trigger when specific functions are called or properties are accessed. When a breakpoint is triggered, the `apiCollector` collects the JavaScript stack trace and metadata related to the function or property access [32, 68]. We added breakpoints for `window.location` and `document.URL` properties to intercept script access to page URLs and their components. Through testing we verified that even a script accesses a property of these objects (e.g., `window.location.href`), we could intercept the access. Capturing JavaScript access to URLs helped identify scripts that bypass the Referrer Policy by directly accessing and sharing page URLs with third parties.

Vantage Points and Crawl Dates. We ran the crawls from three distinct vantage points, to compare the RP measurements across locations: San Francisco (**SF**), Amsterdam (**Ams**), and Singapore (**Sg**). The decision to include Singapore was mainly motivated by increasing the diversity of vantage points, which are commonly limited to Europe [46, 65, 69, 73] and North America [16, 18, 37, 48, 58, 65, 69, 73]. The crawls were run in January 2024 and took roughly twenty days to complete. Crawlers did not interact with consent dialogs that may be present on websites. This approach was chosen to preserve the initial conditions as they occur when a webpage is loaded in a browser, ensuring that any circumventions observed reflect events prior to user interaction with the page. In addition, this passive approach made the results from different vantage points more comparable, since interacting with consent dialogs in the EU may lead to a different treatment, where data protection laws are stricter.

3.3 Potential URL Leakage Detection

While prior studies [28, 48] focused on personal information leakage via the Referer header, our study examines whether third parties receive the webpage's URL, to detect RP circumventions. Detecting URL leaks presents a challenge since URLs may be sent in obfuscated form to evade detection similar to other data exfiltrations [22, 28, 65]. To cope with this challenge, we adopted the Leak Detector tool developed by Senol et al.[65], which draws upon research by Englehardt et al. [36]. Leak Detector considers diverse encoding and hashing techniques such as Base64 and URL (percent) encoding

when searching for leaks in request details. In order to detect potential leaks, we searched for the components of the page URL in the third-party request details. Using Python's `urllib.urlparse` function, we parsed the URL into its components, including the hostname, path, parameters, query and fragments. These components served as search terms to identify potential URL leaks. The search terms are then searched within the request URL, request POST body, and request Referer header to detect any instances of URL components in encoded or plain-text form.

We configure the leak detector to use an extensive array of encoding-decoding techniques (list on Appendix A.3), with a depth of two layers in encoding and decoding. This approach enables the detection of double-encoded URLs, among others. We exclude hash functions from the leak detection since hashed URLs are unlikely to be useful to third parties due to the irreversibility property of hashing.

3.4 Referrer Policy Circumvention Detection

Not all URL transmissions to third parties constitute an RP circumvention. If the RP permits, even the full URL (excluding fragments) can be disclosed to third parties. To detect RP circumventions, we compare the CDP-provided `referrerPolicy` property of each third-party request with the transmitted URL parts detected in the same request. We then classify the results into three categories: 1) *circumvention (full URL)*, if it involves the transmission of the full URL or origin combined with path or query parameters, when the RP disallows the sharing of the full URL; *2) circumvention (partial)*, if it is sending path or query when RP disallows or; 3) *safe*, if no circumvention is detected.

The distinction between *circumvention (full URL)* and *circumvention (partial)* arose during our pilot project. We observed that certain third-party requests did not include a complete URL. Instead, they transmitted only the path or query parameters, possibly because the origin was already sent in the Referer header. To account for this nuance, we introduced the *circumvention (partial)* category.

An additional challenge was due to a very short path and query parameters detected as leaks, many of which appeared to be false positives. To mitigate this issue, we only detected RP circumventions for paths and queries if they were longer than ten characters. Due to this filtering, we present the RP circumventions as percentages or indicate the number of websites included in the analysis. To test the reliability of our detection approach, we conducted a false positive evaluation on 100 *circumvention (partial)* cases and 100 *circumvention (full URL)* cases. The method for this evaluation is detailed in Appendix B, and the results are presented in § 4.2.

4 RESULTS

For simplicity, results are based on the SF crawl unless otherwise specified.

We limit our analysis to 124,202 pages (from 27,570 distinct websites), which were successfully crawled from all three vantage points. Approximately 96% of the pages were successfully visited by our RP crawler (Table 3). Details of how we detect failed visits can be found in Appendix A.2. We categorized the 27,570 websites using McAfee SiteLookup service [7], which resulted in 35 distinct categories with a minimum of 100 websites each (Figure 11 in Appendix C). We enforced the lower limit of 100 to avoid categories with very few websites. The "Business" category emerged as the most common, with 3,902 websites, while the "Content Server" category was the least common, with 104 websites.

Table 3: Number of successfully crawled websites and pages.

	SF	Sg	Ams
Successfully loaded websites	27,414	27,406	26,362
Successfully loaded pages	119,591	119,476	119,243
Successfully loaded pages (%)	96.29%	96.19%	96.01%

Table 4: Percentage of distinct websites applying a document-wide referrer policy. Document-wide referrer policies can be set by either meta-tags or RP response headers.

Referrer Policy	SF	Sg	Ams
no-referrer	7.39%	7.35%	5.77%
same-origin	6.78%	6.96%	6.68%
strict-origin	1.33%	1.32%	1.22%
strict-origin-when-cross-origin	27.14%	26.54%	23.65%
origin	7.07%	6.71%	6.13%
origin-when-cross-origin	8.21%	8.18%	6.06%
no-referrer-when-downgrade	9.95%	9.83%	9.63%
unsafe-url	4.48%	2.74%	1.60%
Total	**48.38%**	**47.73%**	**44.20%**

4.1 Referrer Policy Usage

4.1.1 Document-Wide Referrer Policies . The percentage of websites setting a document-wide RP ranged from 44.20% (Ams) to 48.38% (SF). These policies were implemented using various methods, with 11.09% utilizing meta tags and 43.81% employing response headers. The `strict-origin-when-cross-origin` policy was the most common, observed on 27.14% of the websites. Table 4 shows a detailed breakdown of these values. Across vantage points, we observe little difference in RP usage, with similar percentages overall. The most restrictive RP (`no-referrer`), for instance, was used on 7.39% and 5.77% of the websites when visited from SF and Amsterdam, respectively. The only exception to this trend is `unsafe-url`, which was used on 4.48% and 1.60% of the websites when visited from SF and Amsterdam, respectively. The lower usage of `unsafe-url` in the Amsterdam crawl might result from not interacting with consent dialogs, as we find below that `unsafe-url` is primarily employed by third-party frames. In §4.2.4, we report on a small scale investigation comparing RP implementations and circumventions in different consent modes. However, given the scope of our consent crawl, the findings did not offer conclusive evidence.

Referrer-Policy Response Headers. We observed one or more RP response headers on 43.81% of the websites. The most common policy `strict-origin-when-cross-origin` was found on 26.98% of the websites, followed by `no-referrer-when-downgrade` (9.06%). In contrast, the least popular policy `strict-origin` was observed

Table 5: Percentage of distinct websites with RP response headers. In rare cases, when a website sends multiple RP values in the same header, we determine the applicable policy based on precedence, prioritizing the rightmost value. If the rightmost value is invalid—due to a typo, browser-imposed restrictions (Table 2), or an unrecognized policy—it is disregarded, and the next valid value is selected.

Referrer Policy	SF	Sg	Ams
no-referrer	5.15%	5.18%	4.99%
same-origin	6.75%	6.93%	6.65%
strict-origin	1.32%	1.31%	1.21%
strict-origin-when-cross-origin	26.98%	26.38%	23.47%
origin	4.87%	4.71%	4.47%
origin-when-cross-origin	4.26%	4.29%	4.02%
no-referrer-when-downgrade	9.06%	8.97%	8.77%
unsafe-url	3.60%	1.82%	0.69%

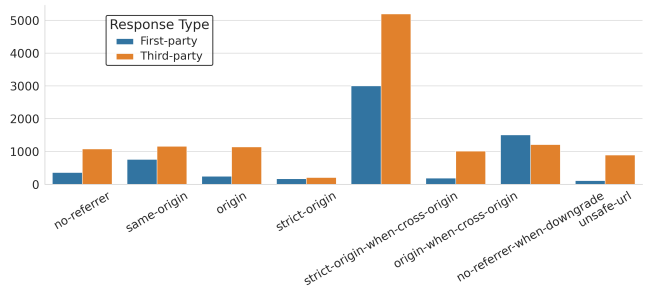

Figure 4: Total distinct websites with Referrer-Policy response headers, grouped by third-partyness (first-party: blue; third-party: orange).

on 1.32% of the websites (Table 5). Similar to aggregate document-wide policies presented above, we find a lower usage of `unsafe-url` in the Amsterdam crawl compared to SF and Sg crawls (SF: 3.60%, Sg: 1.82% Ams: 0.69%). In §4.2.4, we present an exploratory investigation on whether the low `unsafe-url` in the Amsterdam crawl could be explained by lack of consent.

Figure 4 illustrates the distribution of RPs found in response headers, categorized by first-party and third-party domains. Notably, the most permissive RP `unsafe-url` was almost exclusively found in third-party responses, appearing on 891 websites, compared to just 112 websites in first-party responses. Note that RPs found in third-party responses may or may not cause any change in what RP the browser uses. Allowing third parties to set the RP for the whole website would render the RP standard useless. However, cross-origin iframes can set the active RP for their embedded document and requests originating from that document.

Our analysis of response headers also revealed RPs that do not follow the standard. Certain RP headers, for example, contained newline characters ("\n") to separate multiple RPs (Table 22, Appendix C), while the correct syntax requires commas for separation (Table 23, Appendix C).

Table 6: Percentage of distinct websites with referrer policy in HTML `Meta` Tags.

Referrer Policy	SF	Sg	Ams
no-referrer	2.67%	2.57%	0.94%
same-origin	0.04%	0.04%	0.04%
strict-origin	0.01%	0.01%	0.01%
strict-origin-when-cross-origin	0.42%	0.42%	0.43%
origin	2.49%	2.17%	1.81%
origin-when-cross-origin	4.04%	3.98%	2.10%
no-referrer-when-downgrade	1.14%	1.12%	1.11%
unsafe-url	0.97%	0.99%	0.96%

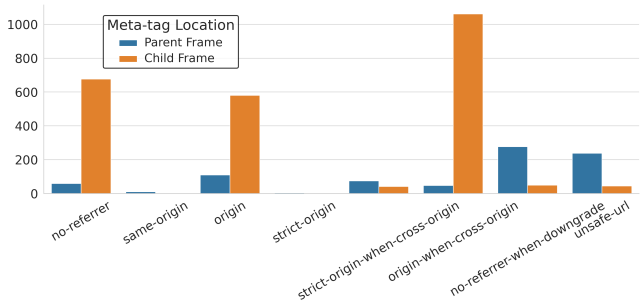

Figure 5: Total distinct websites with meta-tag referrer policy grouped by frame.

HTML `Meta`-tag. Our analysis revealed limited adoption of the RP implementation via HTML meta tag, with only 7.15% websites in Ams, 10.66% in Sg, and 11.09% in SF (Table 6). The most common policy `origin-when-cross-origin` was found on 4.04% of the websites.

We find a significant disparity in RP usage between top-level (parent, first party) and descendant (children) frames (Figure 5). The most common policy in child frame meta tags was `origin-when-cross-origin`, which was found on 1,069 distinct websites. On the other hand, the highly permissive `no-referrer-when-downgrade` policy was most prevalent in top-level document meta tags (277 websites). Recall that `no-referrer-when-downgrade` implies sending the full URL in the referrer, unless the protocol security is downgraded from HTTPS to HTTP. We note that, we only find 2.15% of the approximately 12 million requests captured in the SF crawl were sent over insecure HTTP connections.

In addition, we identified two invalid policies: "none", likely a *legacy* form (§2.3) [43], and "noreferrer", potentially due to a typo or developer familiarity with the link relation attribute. These invalid policies would be ignored by browsers. Furthermore, 3.90% of the websites continue to utilize the legacy `origin-when-crossorigin` meta tag RP, which omits the hyphen in "cross-origin". Browsers still respect legacy policies, which may disincentivize website owners to update them. Table 20 in Appendix C shows other legacy values we found in the RP meta tags.

4.1.2 Element-Specific Referrer Policies. Recall from §2.3 that website owners can use `rel` and `referrerpolicy` attributes to specify

Table 7: Distinct websites implementing `rel=noreferrer`. Note that `rel=noreferrer` applied to `<img>` and `<script>` elements is not supported and will be discarded by browsers [30].

Loc.	a	area	img	script
SF	1,342	2	7	1
Sg	1,327	2	7	1
Ams	1,340	2	7	1

Table 8: Most common referrer policies found in HTML element attributes. Indicated as the percentage of the total websites. The complete version of the table can be found in Table 21 in Appendix C

El.	Location	Most Common Referrer Policy
<a>	SF	origin (0.35%)
	Sg	origin (0.36%)
	Ams	origin (0.47%)
<iframe>	SF	unsafe-url (1.99%)
	Sg	unsafe-url (1.09%)
	Ams	no-referrer-when-downgrade (0.89%)
<img>	SF	origin (0.38%)
	Sg	origin (0.38%)
	Ams	origin (0.39%)
<link>	SF	no-referrer (1.24%)
	Sg	no-referrer (1.24%)
	Ams	no-referrer (1.23%)
<script>	SF	no-referrer (1.22%)
	Sg	no-referrer (1.22%)
	Ams	no-referrer (1.21%)

RPs for individual elements. Analyzing the data on 33 million elements we captured in the SF crawl, we find that 0.17% of the elements had an explicitly defined RP, originating from 13.39% websites. Table 7 and Table 21 (Appendix C) provide detailed breakdowns of the observed element-specific RP values. We note that we only saved information about elements that can take RP attributes (e.g., images, links and anchors), as explained in §3.2.2.

Our results showed that "`rel= noreferrer`" attribute is much more common than the `referrerpolicy` attribute. In the case of `<a>` elements, we found `rel=noreferrer` usage on 1,342 distinct websites (4.90%) (Table 7), significantly more than the combined usage of the `referrerpolicy` in `<a>` elements (0.46%) (Table 21).

While "`rel=noreferrer`" is primarily intended for `<a>` and `<area>` elements, we identified seven websites where it was used for `<img>` and `<script>` elements. Such non-standard implementations are ignored by the browser and have no effect.

Our analysis revealed that 9.03% of distinct websites have at least one element with `referrerpolicy` attribute. The most common policy implemented in each element was the same in every location, except for iframes where Ams has a different value (Table 8; detailed breakdown in Table 21 in Appendix C). In the SF and Sg crawls, `unsafe-url` was the most commonly observed RP for the iframes: 537 (SF), 295 (Sg). Notably, on 405 of the 537 websites in the SF crawl, the `src` domain of the `<iframe>` with the `unsafe-url` RP was rubiconproject.com. The domain belongs to Magnite, an advertising tech company that acts as a supply-side platform (SSP) [27]. Notably, an iframe with `unsafe-url` RP was only observed on only 15 websites in the Ams crawl (Table 21). As mentioned above, the difference in `unsafe-url` usage in iframes could be attributed to local data protection regulations and lack of interaction with consent dialogs in the crawls. We share a comparison of RP implementations and circumventions in different consent modes in §4.2.4.

Table 9: Percentage of distinct websites with extracted request referrer policies. When multiple distinct RP values were present on a single website, each distinct RP value was counted separately.

Referrer Policy	SF	Sg	Ams
no-referrer	9.56%	9.12%	7.25%
same-origin	2.33%	2.28%	2.28%
strict-origin	0.64%	0.64%	0.64%
strict-origin-when-cross-origin	100.00%	100.00%	100.00%
origin	8.08%	7.75%	7.33%
origin-when-cross-origin	5.41%	5.35%	3.54%
no-referrer-when-downgrade	9.50%	9.39%	8.86%
unsafe-url	5.26%	3.63%	2.33%

The `<a>` and `<img>` elements predominantly contained the `origin` policy, indicating that only the origin component of the URL is sent in the Referer header. In contrast, `<link>` elements contained stricter policies, with `no-referrer` being the most common choice. The `<script>` elements primarily contained `no-referrer` in all three crawls. A detailed breakdown of element-specific RPs can be found in Table 21 in Appendix C. These results show that while element-specific RPs are rare, they show a distinction based on the element type. More private policies are preferred for `<link>` and `<script>` elements, which are commonly used to embed CSS and JavaScript resources, respectively. On the other hand, least private RPs are used for `<iframe>` elements, which can be used to load cross-origin resources including advertisements.

4.1.3 Request Referrer Policy Values. Recall that CDP provides a property called "`referrerPolicy`" for each request, which reflects the cumulative effect of all applicable RPs [42]. Importantly, "`referrerPolicy`" is not an actual HTTP header, but is provided by the CDP for diagnostic purposes. Examining "`referrerPolicy`" for 12 million requests captured in the SF crawl reveals `strict-origin-when-cross-origin` as the most prevalent policy (observed on all websites), followed by `no-referrer` (9.56%) and `no-referrer-when-downgrade` (9.50%) (Table 9).

While the median website utilizes a single policy, we observed websites with up to six distinct RPs (e.g., breezeline.com). On average, websites employ 1.41 distinct RPs, with 32.09% using two or more.

4.1.4 Permissive vs. Restrictive Referrer Policies. As noted in Table 2, Safari and Firefox disregard the most *permissive* RPs; namely, `no-referrer-when-downgrade`, `origin-when-cross-origin` and

`unsafe-url`. We group and compare these permissive policies with *restrictive* RPs; namely, `no-referrer`, `same-origin`, and `strict-origin`.

Comparing the proportion of websites that use permissive and restrictive RPs across website categories, we find that the three categories that use more permissive RPs are Blogs/Wikis, Portal Sites, and General News, whereas the top three categories for *restrictive* RPs are Finance, Blogs/Wikis, and Job Search (Figure 6). A possible explanation for news websites having more permissive RPs could be that they are known to include more third-party trackers compared to other categories [37].

To assess whether the use of permissive or restrictive RPs change with websites' popularity, we compared the Tranco ranks of websites where we found these two types of RPs. Our analysis did not reveal a meaningful difference: permissive RPs are used on 18.30% of websites (median rank: 417,219.5), while restrictive RPs are used on 18.10% of websites (median rank: 422,384).

4.2 Referrer Policy Circumventions

As outlined in §3.4 our analysis of RP circumventions is based on three categories: 1) *safe*: no URL leaks; 2) *circumvention (full URL)*: full URL leak, when disallowed by the RP; and 3) *circumvention (partial)*: partial URL leak, when disallowed by the RP. Our results indicate that circumventions with full URL leaks were observed in 21,084 websites, corresponding to 76.91% of the studied websites. Circumventions with partial URL (path and/or query), on the other hand, were detected on 2,704 websites (9.86%) (Table 10).

As previously discussed, *circumvention (partial)* occurs when a third party transmits only the path and/or query parameters, excluding the origin. This method relies on setting a minimum character threshold for path or query components to mitigate false positives.

To verify the detected leaks, we selected 100 random samples from each of the *circumvention (full)* and *circumvention (partial)* categories. We checked the detected leaks for false positives. The evaluation of full URL leaks yielded no false positives. In contrast, we found 12/100 false positives when manually reviewing the circumventions with partial URL leaks. Despite this, the dominant trend remains that full URL circumvention is overwhelmingly prevalent. Among the 21,164 distinct websites where circumvention was detected, 21,084 (99.62%) involved full URL leaks. While partial circumvention was observed in 2,704 cases (12.78% of total circumventions), only 80 instances did not overlap with full URL circumventions. As a result, the 12% false positive rate associated with partial circumventions does not significantly impact the overall conclusions of our study.

From this subsequent subsection, we consolidate *circumvention (full URL)* and *circumvention (partial)* into a single category, *circumvention*, as we now classify websites in both groups as having breached the RP.

RP circumventions were identified on 77.20% of the websites (Table 10) and in 11.57% of the third-party requests (Table 14). These findings are similar across other vantage points. In Singapore, circumventions were observed on 77.02% of websites, while in Amsterdam, the rate was slightly lower at 76.27% (Table 14).

Table 10: Referrer policy circumventions in the SF crawl based on the number of distinct websites. "Path" and "Path and query" circumventions are limited to cases where a minimum of 10 characters is present to prevent false positives. "# Websites" represents the number of websites that meet specific conditions: in the "Path" row, it indicates websites where the path component exists, while in the "Path and Query" row, it refers to websites where both the path and query components exist.

Leaking	Circum.	# Web.	%
Full URL	21,084	27,414	76.91%
Path (without origin)	2,677	27,414	9.77%
Path and Query (without origin)	182	3,534	5.15%
Path or Query (without origin)	2,704	27,414	9.86%
Total	**21,164**	**27,414**	**77.20%**

Table 11: Percentage of distinct websites with referrer policy circumventions. The percentage is calculated by dividing the total number of circumventions found for a specific referrer policy by the total number of distinct websites where that policy appears (Table 9).

Referrer Policy	SF	Sg	Ams
no-referrer	11.75%	11.84%	14.17%
same-origin	60.82%	61.82%	61.60%
strict-origin	65.14%	64.00%	61.93%
strict-origin-when-cross-origin	74.60%	74.38%	73.62%
origin	22.20%	20.40%	21.75%
origin-when-cross-origin	17.05%	16.97%	23.24%

Table 11 shows the percentage of circumventions for each RP. We observed an RP circumvention on 11,75% of the websites that use the `no-referrer` policy. The most frequently circumvented policy was the `strict-origin-when-cross-origin` policy; 74.60% of the websites with that policy had an RP circumvention. While the low rate of circumventions on websites using `no-referrer` is a positive finding, we observe this most restrictive policy on only 2,622 websites, representing 9.56% of the total websites (Table 9). On the other hand, the most circumvented policy (`strict-origin-when-cross-origin`) is also the most prevalent policy, being observed on all websites.

4.2.1 Circumventing Domains and Entities. Analyzing the domains of the requests that contained RP circumventions, we find three Google domains (google-analytics.com: 89.14%, google.com: 63.64%, and doubleclick.net: 57.91%) to be the most common (Table 12). All ten domains are associated with companies offering advertisement, marketing or analytics services [7, 33, 60]. Table 13 shows the top ten entities where requests with RP circumventions sent to Google, Facebook and Microsoft top the list, while TikTok's owner ByteDance found to circumvent RPs on 88.85% of the websites it is embedded on. Note that while TikTok is known for its mobile app,

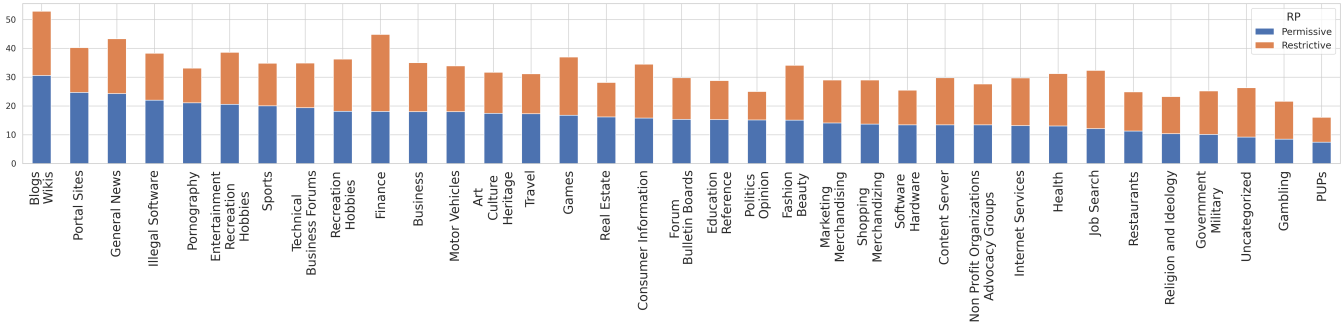

Figure 6: Comparison of website categories by referrer policy quality. Sorted by the percentage of permissive referrer policies. The detail per referrer policy can be seen in Figure 12 in Appendix C.

Table 12: Count of third-party domains circumventing referrer policy, based on their appearance on distinct websites. Sorted by distinct websites with circumventions. (Table 9)

Third-party Domain	Appears on	Circumvents on
google-analytics.com	16,821	14,995 (89.14%)
google.com	16,069	10,226 (63.64%)
doubleclick.net	14,623	8,468 (57.91%)
facebook.com	7,886	6,531 (82.82%)
bing.com	2,521	1,289 (51.13%)
linkedin.com	1,860	1,198 (64.41%)
twitter.com	1,687	1,151 (68.23%)
yandex.com	1,230	1,124 (91.38%)
googleadservices.com	2,001	1,066 (53.27%)
adnxs.com	2,099	1,055 (50.26%)

Table 13: Count of third-party entities circumventing referrer policy, based on their appearance on distinct websites. Sorted by distinct websites with circumventions.

Third-party Entity	Appears on	Circum. on
Google LLC	24,120	18,317 (75.94%)
Facebook Inc.	8,992	6,573 (73.10%)
Microsoft Corporation	5,818	3,721 (63.96%)
Yandex LLC	1,340	1,221 (91.12%)
Twitter Inc.	1,741	1,151 (66.11%)
Media.net Advertising FZ-LLC	1,359	1,043 (76.75%)
ByteDance Ltd.	1,121	996 (88.85%)
Pinterest Inc	1,047	911 (87.01%)
Akamai Technologies	1,182	812 (68.70%)
HubSpot Inc.	1,009	707 (70.07%)

it also offers TikTok Pixel for website owners who want to retarget their visitors with ads on the TikTok app [5].

4.2.2 Circumventions by Website Categories . The RP circumventions were observed across various website categories (Figure 12), with Fashion/Beauty (86.69%), Shopping/Merchandising (84.06%), and General News (83.92%) emerging as the top three categories. Conversely, Uncategorized (59.48%), Pornography (56.36%), and Gambling (55.86%) were found to contain the least number of RP circumventions. The lower rate of circumventions on websites with objectionable content may stem from the higher privacy expectations of visitors to such sites. For adult entertainment websites specifically, the results are consistent with Altaweel et al.'s finding that these sites tend to have fewer third-party trackers compared to non-adult websites with comparable popularity [15]. On the other hand, news and shopping websites are more likely to contain third-party trackers for monetization and marketing purposes [37].

Table 14: Percentage of referrer policy circumventions, grouped by leak vector. Req: Distinct third-party requests; Web: Distinct websites; Total: All vectors combined, counted once if multiple circumvention vectors appear on a request or a website. More detailed measurements are given in Table 24.

Vector	SF		Sg		Ams	
	Req	Web	Req	Web	Req	Web
Referrer	0,63%	9.15%	0.57%	8.33%	0.51%	5.28%
Post	4.14%	32.57%	3.17%	32.12%	2.03%	29.13%
URL	13.21%	74.78%	13.85%	74.64%	13.09%	73.78%
Total	**17.76%**	**77.20%**	**17.37%**	**77.02%**	**15.49%**	**76.27%**

4.2.3 Circumventions by Leak Vectors. Table 14 (detailed on Table 24) shows that URL leaks were most commonly sent in the request URL in RP circumventions (74.78% of the websites). This is followed by 32.57% of the websites where we observed a leak in the POST body. In those two cases, a script simply retrieves the current page URL and sends it in the request URL (Figure 10) or in the POST payload (Figure 9).

On 9.15% of the websites, however, we observe a RP-circumventing leak in the Referer header. This is at first counter-intuitive, since the browser should have enforced the active RP, preventing the sending of the URL as the Referer. A closer look reveals that the page URL is not sent as the referrer, but is appended to the cross-origin iframe's src attribute—which in turn was sent as the referrer.

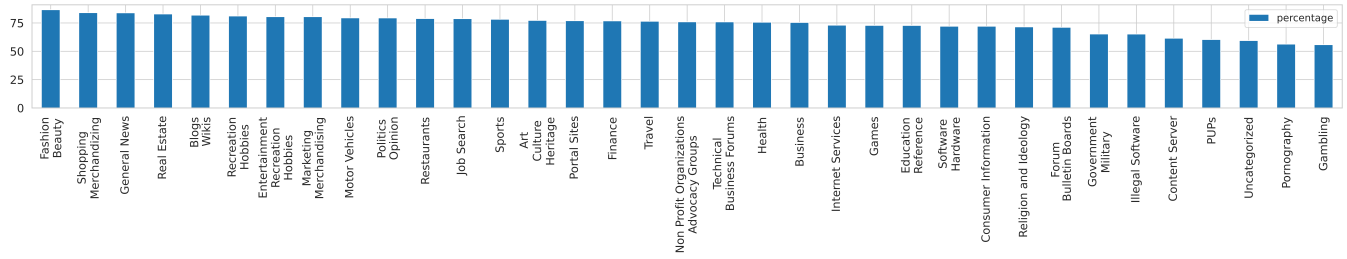

Figure 7: Percentage of website categories with referrer policy circumventions

An example of this circumvention found on voyagingtheworld.com is shown in Figure 8. The request originates from a cross-origin iframe (ferryscanner.com) and sends the iframe's src (or URL) in the Referer header. However, the address bar URL (voyagingtheworld.com) is appended to the iframe's src, likely when the iframe is injected to the page. Therefore, despite the active RP (`strict-origin-when-cross-origin`) the third-party domain of the iframe receives the full address bar URL. We note that this circumvention is only possible with the cooperation of a script running with the top-level document's privileges, since scripts from the cross-origin iframe cannot access the page URL due to Same-Origin Policy. While the third party, in this case, seems to provide functionality to the website, this method can be used for tracking-related purposes as well.

4.2.4 Interacting with Consent Dialogs. We conducted a small-scale additional crawl from Amsterdam to explore how consent affects RP implementations and circumvention. We randomly selected 100 websites and followed their inner links, yielding 309 total pages in our sample (See Appendix B, for details). To automatically handle consent dialogs on these pages, we employed DuckDuckGo's autoconsent library [31], which is bundled with the Tracker Radar Collector. Our analysis indicates that interacting with consent dialogs has only a minor impact on RP implementations (Table 17, Appendix B) and circumventions (Table 19, Appendix B). Recall that the use of `unsafe-url` was found to be lower in the Amsterdam crawl compared to other vantage points (Table 4). We do not observe an increase in `unsafe-url` usage in the optIn crawl. This may indicate that third parties using the `unsafe-url` RP may not be operating within the EU or on the websites we have crawled for this small investigation. Turning to circumventions, when no interaction with the consent dialog occurred, we observed that 72 of the 100 websites exhibited at least one instance of RP circumvention. In the optOut crawl, the number of websites with circumventions slightly increased to 74. Finally, in the optIn crawl, we observed RP circumvention on 79 of the 100 websites (Table 18, Appendix B). Variations in circumvention across the three crawls may partly stem from website dynamism. However, with measurements spanning 309 webpages across 100 sites, the likelihood of these differences being solely due to dynamism is low. Overall, in this limited sample, consent impacts circumvention by only a single-digit percentage.

4.2.5 Encodings Used in Referrer Policy Circumventions. In many cases of circumvention, the URL is transmitted in an encoded form. We identified 20,612 websites (out of 21,164) where at least one

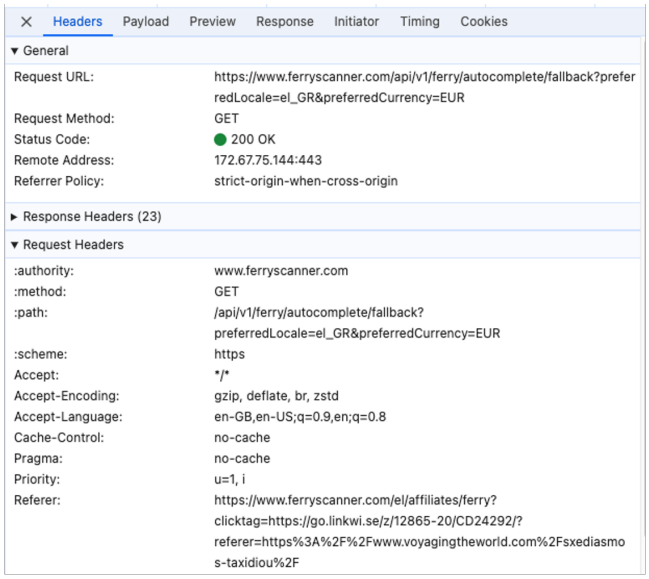

Figure 8: An example referrer policy circumvention via the Referer header. The full URL of the address bar URL (https://www.voyagingtheworld.com/sxe...) is appended to the cross-origin iframe's src attribute, which is sent in the Referer header. The active referrer policy (`strict-origin-when-cross-origin`) only allows for origin to be sent in cross-origin requests, but this method bypasses the policy. Note that this bypass method still requires the cooperation of a script running with the first-party's privileges to access the top-level document's URL.

circumvention instance across the three vectors utilized URL encoding, while 2,791 websites employed Base64 encoding. Additionally, we observed a third party (Yandex LLC) using HTML escaping (`html.escape`), which appeared on three websites. A compression method, LZString [63], was detected on 13 websites, with the highest usage attributed to CleverTap, a company specializing marketing and personalization, which implemented it on four websites.

4.2.6 Script Accesses to Document URLs. Recall that our crawler intercepted access to JavaScript object properties that reveal the current document URL such as `location.href` and `document.URL`. Note that for scripts running in cross-origin iframes, these properties will return the iframe's URL, not the address bar URL. Table15

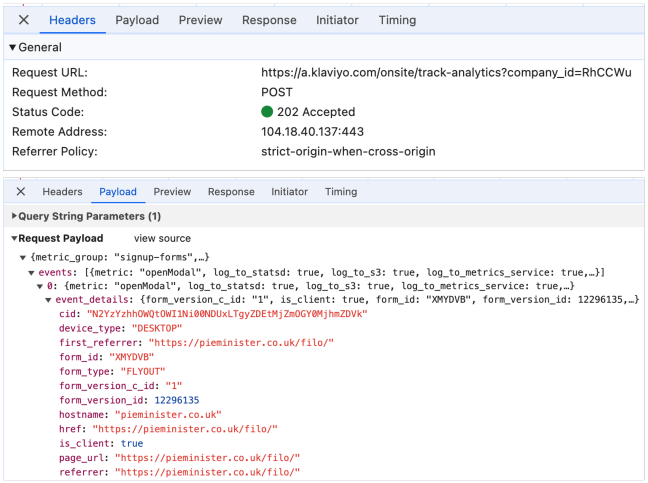

Figure 9: Example of referrer policy circumvention via POST body. The full page URL (https://pieminister.co.uk/filo/) is sent in the POST body, circumventing the `strict-origin-when-cross-origin` policy.

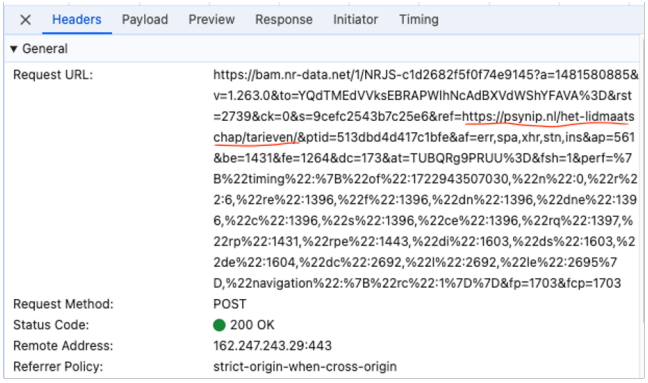

Figure 10: Example of referrer policy circumvention via request URL. The request URL contains the full page URL (https://psynip.nl/het-lidmaatschap/tarieven), circumventing the `strict-origin-when-cross-origin` policy.

shows the percentage of websites where we observed one or more access to these properties. The `location.href` property, which contains the full URL, was accessed on 89.63% of the websites. The port component of the URL, which is rarely used since default ports 443 or 80 are assumed, is the least commonly accessed (13.19% of the websites). Accessing these object properties does not necessarily translate to RP circumventions, but it is a necessary condition. The monitoring of access to these properties can also inform defenses against RP circumventions. For instance, in the most extreme case, we observe approximately 29,788 accesses (SF) to `location.href` on the same website, with the highest count reaching over 53,000 access (Ams). These accesses were performed by a script from infinity-tracking.com, which offers marketing "phone call analytics" services [6].

Table 15: Percentage of distinct websites where one or more third-party script accesses to `window.location` or `document.URL` was observed.

	SF	Sg	Ams
location.href	89.63%	89.53%	89.49%
location.protocol	85.41%	85.28%	84.81%
location.host	56.58%	56.50%	55.82%
location.hostname	84.83%	84.69%	84.36%
location.port	13.19%	12.68%	12.52%
location.pathname	79.47%	79.34%	78.96%
location.hash	71.49%	71.31%	70.75%
document.URL	34.37%	34.30%	30.21%

5 DISCUSSION

Our analysis revealed widespread circumvention of website-imposed RPs by advertising and analytics scripts such as Google Analytics, Facebook Pixel and TikTok Pixel. While these third parties do not exploit a vulnerability in the RP standard, they misuse the unfettered access given to scripts even when the website uses a restrictive RP. In that respect, the circumventions we identified in this study are by design: third-party scripts running in the top-level document's context enjoy the same privileges as the first party. This distinction even confuses some developers, who expect RP to apply to third-party scripts when restrictive RPs are in place [13].

We propose an opt-in mechanism to manage script access to referrer information, including the current top-level document URL. Specifically, a new RP value could allow websites to opt-in to trim `document.referrer`, `location.href`, and `document.URL` for scripts, aligning with the document-wide RP. We acknowledge that this approach may be challenging to implement. For instance, in Chromium-based browsers, scripts in cross-origin frames can access *location.ancestorOrigins*, regardless of the active RP. The property exposes ancestor origins even when the RP prohibits it (e.g., *no-referrer*) [10]. This issue led Firefox to forgo implementing the *location.ancestorOrigins* property, resulting in inconsistencies in browser support [4, 9]. Moreover, the *location* object is notoriously complex and any changes to it would be challenging to implement securely[4].

We observed minor differences in RP implementations across vantage points — especially a lower `unsafe-url` usage in the Amsterdam crawl (§ 4.1.1). This reduction may stem from local data protection regulations and limited interaction with consent dialogs, as `unsafe-url` is predominantly used by third-party frames (Table 21, Appendix C). Overall, these differences are expected since websites and advertisers dynamically localize their content based on the user's geographic location [16, 20, 56]. The number of websites with RP circumventions did not differ across vantage points (SF: 77.20%, Sg: 77.02%, Ams: 76.27%) (§ 4.2). Similarly, our exploratory analysis on the impact of consent revealed minimal differences across three consent modes. The only exception was that opting

[4]The HTML standard contains the following warning: "The Location exotic object is defined through a mishmash of IDL, invocation of JavaScript internal methods post-creation, and overridden JavaScript internal methods. Coupled with its scary security policy, please take extra care while implementing this excrescence." [8]

in resulted in a single-digit increase in observed circumventions (Appendix B).

The ability of third-party scripts to bypass the Referrer Policy (RP) could have legal implications. If URLs containing personal information are exfiltrated without user consent, they may potentially breach the EU's General Data Protection Regulation (GDPR) or other relevant data protection laws. In fact, the similar number of leaks observed in the Amsterdam (EU) crawl suggests that websites do not require user consent to load the third-party scripts responsible for exfiltrating the page URL.

Gambling websites, which often handle sensitive personal and financial information [14, 62], showed a relatively low circumvention rate (55%) and minimal use of permissive RPs (8.43%) (§ 4.2.2). However, this result does not imply that such websites are safe since they are prime targets for malicious actors given the sensitive data they possess [21, 23].

The differences in RP usage between parent and children frames suggest that similar future measurement studies should capture element-specific RP implementations in all frames, not just the top-level document.

5.1 Limitations and Future Work

Consent Dialogs. Prior research [46, 49, 73] indicates that consent dialogs are often designed to be easy to accept but difficult to decline, and giving consent influences the type and amount of data shared with third parties.

In our primary study, the decision to not interact with consent dialogs allowed for more consistent comparisons across vantage points, as all measurements were conducted under similar conditions. However, this methodological choice likely rendered the Amsterdam crawl (where GDPR applies) less representative of real user experiences, as actual users may be required to interact with consent dialogs to access content. Consequently, we argue that the RP circumventions detected in our study—particularly in the Amsterdam crawl—likely represent lower bounds of the actual circumvention rates. In order to address this shortcoming, we conducted an additional small-scale (100 website) crawl to assess the potential impact of consent dialogs on RP circumventions (§4.2.4). While our crawl provides a limited overview, future work can analyze this effect more comprehensively.

Leak Detection False Positives. Leak detection module looks for a URL or its components in millions of requests, leading to potential false positives. To minimize false positives, we enforced a lower limit of ten characters when searching the path and query components. We also quantified the false positives in a random sample (§ 4.2). Analyzing 200 detected circumventions (100 partial and 100 full URL leaks) we found no false positives in full URL leaks and 12 false positives in circumventions involving partial URL leaks. Since full URL leaks constitute 99.6% of the circumventions, the effect of false positives due to partial URL leaks on our results is negligible.

Cookies as a Leak Vector. This study focused on three leakage vectors: request URLs, POST body data, and the Referer header. However, prior research [28, 55, 72] has demonstrated that cookies can also be used as a vector for data leakage. Future studies can extend the leak detection analysis to include cookies. Finding circumventions on over 77% of the websites, we believe our study adequately demonstrated RP's shortcomings.

Script Access to `document.referrer`. We intercepted and logged script access to `location.href` and `document.URL`—two host object properties that reveal the current document's URL. However, we did not similarly monitor *document.referrer*, which was unnecessary for our study and methods.

6 CONCLUSION

We investigated implementations and circumventions of the Referrer Policy standard across 27,750 distinct websites. Based on crawls from three vantage points, we found that 48.38% of the websites implemented document-wide referrer policies, while 17.19% used element-specific policies. On 43.81% of the websites the Referrer-Policy HTTP response header was used, while only 11.09% of the sites opted to set their referrer policy using an HTML meta tag. The default `strict-origin-when-cross-origin` policy also found to be the most commonly preferred (27.14%), while less than eight percent of the sites opted to not send any referrer at all (`no-referrer`: 7.39%).

Our findings reveal that referrer policy circumventions are widespread, with 77.20% of websites having at least one third-party request breaching their declared policies. Most circumventions stem from third-party advertising and analytics scripts directly accessing and disclosing full page URLs. The domain google-analytics.com accounted for the highest number of referrer policy circumventions; 14,996 out of 16,999 websites, where it is embedded on. We did not find any meaningful differences in referrer policy circumvention across vantage points, but we observed slightly lower use of `unsafe-url` policy in the Amsterdam and Singapore crawls (Ams: 1.60%, Sg: 2.74%, SF: 4.48%). Additionally, the Fashion/Beauty category consistently ranked among the highest in circumvention rates, exceeding 86% across all vantage points.

Our findings underscore the need for stricter access control mechanisms to prevent circumvention of referrer policies that websites implement. We recommend that browser vendors explore providing websites with an opt-in mechanism to restrict script access to referrer information. In the meantime, website operators and public should be better informed about the limitations of the Referrer Policy standard, an issue that our study contributes to.

CODE AND DATA

The source code and the dataset from our study are publicly available at https://github.com/referrer-policy-pets-25.

ACKNOWLEDGMENTS

The first author is supported by the scholarship and research funding provided by the Center for Financing of Higher Education (BPPT) and the Indonesia Endowment Fund for Education (LPDP) under the 'Beasiswa Pendidikan Indonesia (BPI)' scholarship program. The preliminary investigation of Referrer Policy circumventions was performed by Thomas van Ouwerkerk in his master thesis [71]. The authors used generative AI based writing assistant software to correct typos, grammatical errors, and awkward phrasing.

REFERENCES

[1] 2018. Good Practices for Capability URLs. https://w3ctag.github.io/capability-urls [Online; accessed 30. Nov. 2024].

[2] 2020. Improving Privacy by Limiting Referrers. https://textslashplain.com/2019/10/16/privacy-tweaks-limiting-referrer [Online; accessed 1. Dec. 2024].

[3] 2021. UPchieve Disclosed on HackerOne: Password Reset Token Leak on Third... https://hackerone.com/reports/1177287.

[4] 2024. 1085214 - Implement Location.ancestorOrigins. https://bugzilla.mozilla.org/show_bug.cgi?id=1085214#c23 [Online; accessed 30. Nov. 2024].

[5] 2024. About TikTok Pixel | TikTok Ads Manager. https://ads.tiktok.com/help/article/tiktok-pixel [Online; accessed 30. Nov. 2024].

[6] 2024. Call Tracking | Infinity - Europe's best call tracking tool. https://www.infinity.co/uk/call-tracking [Online; accessed 30. Nov. 2024].

[7] 2024. Customer URL Ticketing System. https://sitelookup.mcafee.com/en. [Accessed 20-05-2024].

[8] 2024. HTML Standard. https://html.spec.whatwg.org/multipage/nav-history-apis.html#location [Online; accessed 30. Nov. 2024].

[9] 2024. Meta: add ancestorOrigins warning due to continued disagreement by annevk · Pull Request #2251 · whatwg/html. https://github.com/whatwg/html/pull/2251/commits/3c9c428a9750066bbbd3ce517952e3de938312e2 [Online; accessed 1. Dec. 2024].

[10] 2024. redact location.ancestorOrigins according to Referrer Policy · Issue #1918 · whatwg/html. https://github.com/whatwg/html/issues/1918 [Online; accessed 30. Nov. 2024].

[11] 2024. Referer - HTTP | MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer.

[12] 2024. Referrer Policy | Can I Use... Support Tables for HTML5, CSS3, Etc. https://caniuse.com/referrer-policy.

[13] 2024. third-party JavaScript access to document.location when a restrictive Referrer-Policy is set? https://stackoverflow.com/questions/75860851/third-party-javascript-access-to-document-location-when-a-restrictive-referrer-p [Online; accessed 30. Nov. 2024].

[14] Lawrance Abrams. 2019. Online Casino Database Leaks Details of Over 100 Million Bets. https://www.bleepingcomputer.com/news/security/online-casino-database-leaks-details-of-over-100-million-bets/.

[15] Ibrahim Altaweel, Maximillian Hils, and Chris Jay Hoofnagle. 2016. Privacy on adult websites. In *Altaweel et al., Privacy on Adult Websites, Workshop on Technology and Consumer Protection (ConPro'17), co-located with the 38th IEEE Symposium on Security and Privacy, San Jose, CA (2017)*.

[16] I. Anastácio, B. Martins, and P. Calado. 2010. Using the Geographic Scopes of Web Documents for Contextual Advertising. In *Proceedings of the 6th Workshop on Geographic Information Retrieval, GIR'10*. https://doi.org/10.1145/1722080.1722103

[17] HTTP Archive. 2024. The HTTP Archive. https://httparchive.org [Online; accessed 29. Nov. 2024].

[18] Muhammad Abu Bakar Aziz and Christo Wilson. 2024. Johnny Still Can't Opt-out: Assessing the IAB CCPA Compliance Framework. *Proceedings on Privacy Enhancing Technologies* (2024).

[19] Tim Berners-Lee, Roy T. Fielding, and Larry M. Masinter. 2005. *Uniform Resource Identifier (URI): Generic Syntax*. Request for Comments RFC 3986. Internet Engineering Task Force. https://doi.org/10.17487/RFC3986

[20] V. Bhatia and V. Hasija. 2016. Targeted Advertising Using Behavioural Data and Social Data Mining. In *International Conference on Ubiquitous and Future Networks, ICUFN*, Vol. 2016-August. 937–942. https://doi.org/10.1109/ICUFN.2016.7536934

[21] Harry Bowles and Darragh McGee. 2023. Data Ownership, Athlete Rights and the Global Sports Gambling Industry. In *Gambling and Sport in a Global Age*, Darragh McGee and Dunn Christopher (Eds.). Emerald Group Publishing Ltd.

[22] Manolis Chatzimpyrros, Konstantinos Solomos, and Sotiris Ioannidis. 2019. You Shall Not Register! Detecting Privacy Leaks Across Registration Forms. In *Computer Security: ESORICS 2019 International Workshops, IOSec, MSTEC, and FINSEC, Luxembourg City, Luxembourg, September 26–27, 2019, Revised Selected Papers*. Springer-Verlag, Berlin, Heidelberg, 91–104. https://doi.org/10.1007/978-3-030-42051-2_7

[23] Wolfie Christl. 2022. *Digital Profiling in the Online Gambling Industry*. Technical Report. Clean Up Gambling.

[24] MDN Contributor. 2023. Referrer Policy | MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy#unsafe-url_2 "Access Date: 3/09/2023".

[25] MDN Contributor. 2023. What is a URL? | MDN. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL "Access Date: 3/09/2023".

[26] Contributors to Wikimedia projects. 2024. HTTP referer - Wikipedia. https://en.wikipedia.org/w/index.php?title=HTTP_referer&oldid=1254661761 [Online; accessed 1. Dec. 2024].

[27] Contributors to Wikimedia projects. 2024. Magnite Inc - Wikipedia. https://en.wikipedia.org/w/index.php?title=Magnite_Inc&oldid=1237463521 [Online; accessed 30. Nov. 2024].

[28] Ha Dao and Kensuke Fukuda. 2021. Alternative to third-party cookies: Investigating persistent PII leakage-based web tracking. *CoNEXT 2021 - Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies* (12 2021), 223–229. https://doi.org/10.1145/3485983.3494860

[29] Alexis Deveria. 2024. Can I use... https://caniuse.com. [Online; accessed 29. Nov. 2024].

[30] Jochen Eisinger Dominic Farolino and Emily Stark. 2023. *Referrer Policy*. Editor's Draft. W3C. https://w3c.github.io/webappsec-referrer-policy/.

[31] DuckDuckGo. 2023. autoconsent. https://github.com/duckduckgo/autoconsent. [Accessed 28 Feb. 2023].

[32] Duckduckgo. 2023. Duckduckgo/tracker-radar-collector:Modular, multithreaded, puppeteer-based crawler. https://github.com/duckduckgo/tracker-radar-collector

[33] Duckduckgo. 2024. GitHub - duckduckgo/tracker-radar: Data set of top third party web domains with rich metadata about them — github.com. https://github.com/duckduckgo/tracker-radar?tab=readme-ov-file. [Accessed 20-04-2024].

[34] Jochen Eisinger and Emily Stark. 2017. *Referrer Policy*. Candidate Recommendation. W3C. https://www.w3.org/TR/2017/CR-referrer-policy-20170126/.

[35] Jochen Eisinger and Mike West. 2014. *Referrer Policy First Working Draft*. Working Draft. W3C. https://www.w3.org/TR/2014/WD-referrer-policy-20140807/.

[36] Steven Englehardt, Jeffrey Han, and Arvind Narayanan. 2018. I never signed up for this! Privacy implications of email tracking. *Proc. Priv. Enhancing Technol.* 2018, 1 (2018), 109–126.

[37] Steven Englehardt and Arvind Narayanan. 2016. Online Tracking: A 1-million-site Measurement and Analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1388–1401.

[38] Roy T. Fielding, Mark Nottingham, and Julian Reschke. 2022. *HTTP Semantics*. Request for Comments RFC 9110. Internet Engineering Task Force. https://doi.org/10.17487/RFC9110

[39] Pascal Gadient, Oscar Nierstrasz, and Mohammad Ghafari. 2021. Security Header Fields in HTTP Clients. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. 93–101. https://doi.org/10.1109/QRS54544.2021.00020

[40] GoogleChrome. 2024. HTML and JavaScript usage metrics. https://chromestatus.com/metrics/feature/timeline/popularity/1755.

[41] GoogleChrome. 2024. Overview of CrUX | Chrome UX Report | Chrome for Developers. https://developer.chrome.com/docs/crux.

[42] Web Hypertext Application Technology Working Group. 2024. Chrome DevTools Protocol - Network Domain. https://chromedevtools.github.io/devtools-protocol/tot/Network/#type-Request. [Accessed 2024-10-15].

[43] Web Hypertext Application Technology Working Group. 2024. HTML Standard - Semantics - Meta Referrer. https://html.spec.whatwg.org/multipage/semantics.html#meta-referrer. [Accessed 2024-05-23].

[44] Web Hypertext Application Technology Working Group. 2024. HTML Standard - URLs and Fetching - Referrer Policy Attributes. https://html.spec.whatwg.org/multipage/urls-and-fetching.html#referrer-policy-attributes. [Accessed 2024-05-23].

[45] Scott Helme. 2024. Referrer Policy Report. https://crawler.ninja/files/rp-values.txt.

[46] N. Jha, M. Trevisan, L. Vassio, and M. Mellia. 2022. The Internet with Privacy Policies: Measuring the Web Upon Consent. *ACM Transactions on the Web* 16, 3 (2022). https://doi.org/10.1145/3555352

[47] Beliz Kaleli, Manuel Egele, and Gianluca Stringhini. 2019. On the Perils of Leaking Referrers in Online Collaboration Services. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Roberto Perdisci, Clémentine Maurice, Giorgio Giacinto, and Magnus Almgren (Eds.). Springer International Publishing, Cham, 67–85. https://doi.org/10.1007/978-3-030-22038-9_4

[48] Daniel Kats, David Luz Silva, and Johann Roturier. 2022. Who Knows I Like Jelly Beans? An Investigation Into Search Privacy. *Proceedings on Privacy Enhancing Technologies* 2022, 2 (April 2022), 426–446. https://doi.org/10.2478/popets-2022-0053

[49] Simon Koch, Benjamin Altpeter, and Martin Johns. 2023. The {OK} Is Not Enough: A Large Scale Study of Consent Dialogs in Smartphone Applications. In *32nd USENIX Security Symposium (USENIX Security 23)*. 5467–5484.

[50] Modi Konark. 2019. Watching Them Watching Us - How Websites Are Leaking Sensitive Data to Third-Parties. https://dev.to/konarkmodi/watching-them-watching-us-how-websites-are-leaking-sensitive-data-to-third-parties-1nn3.

[51] Balachander Krishnamurthy and Craig E. Wills. 2009. On the Leakage of Personally Identifiable Information via Online Social Networks. In *Proceedings of the 2nd ACM Workshop on Online Social Networks (WOSN '09)*. Association for Computing Machinery, New York, NY, USA, 7–12. https://doi.org/10.1145/1592665.1592668

[52] Arturs Lavrenovs and F. Jesus Rubio Melon. 2018. HTTP security headers analysis of top one million websites. *2018 10th International Conference on Cyber Conflict (CyCon)*, 345–370. https://doi.org/10.23919/CYCON.2018.8405025

[53] Arturs Lavrenovs and Gabor Visky. 2019. Investigating HTTP Response Headers for the Classification of Devices on the Internet. In *2019 IEEE 7th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*. 1–6. https://doi.org/10.1109/AIEEE48629.2019.8977115

[54] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*. https://doi.org/10.14722/ndss.2019.23386
[55] Timothy Libert. 2015. Exposing the Invisible Web: An Analysis of Third-Party HTTP Requests on 1 Million Websites. *International Journal of Communication* 9, 0 (Oct. 2015), 18.
[56] T.T.C. Lin and J.R. Bautista. 2020. Content-Related Factors Influence Perceived Value of Location-Based Mobile Advertising. *Journal of Computer Information Systems* 60, 2 (2020), 184–193. https://doi.org/10.1080/08874417.2018.1432995
[57] Meng Luo, Pierre Laperdrix, Nima Honarmand, and Nick Nikiforakis. 2019. Time Does Not Heal All Wounds: A Longitudinal Analysis of Security-Mechanism Support in Mobile Browsers. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. https://doi.org/10.14722/ndss.2019.23149
[58] Masood Mansoori, Yuichi Hirose, Ian Welch, and Kim-Kwang Raymond Choo. 2016. Empirical Analysis of Impact of HTTP Referer on Malicious Website Behaviour and Delivery. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. 941–948. https://doi.org/10.1109/AINA.2016.113
[59] John McGahagan, Darshan Bhansali, Margaret Gratian, and Michel Cukier. 2019. A Comprehensive Evaluation of HTTP Header Features for Detecting Malicious Websites. In *2019 15th European Dependable Computing Conference (EDCC)*. 75–82. https://doi.org/10.1109/EDCC.2019.00025
[60] Mercedes Killeen. 2024. 28 Best Marketing Analytics Tools. https://agencyanalytics.com/blog/best-marketing-analytics-tools [Online; accessed 28. February. 2025].
[61] Z. Moti, A. Senol, H. Bostani, F. Zuiderveen Borgesius, V. Moonsamy, A. Mathur, and G. Acar. 2024. Targeted and Troublesome: Tracking and Advertising on Children's Websites. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 121–121. https://doi.org/10.1109/SP54263.2024.00118
[62] Phil Muncaster. 2020. Casino App Clubillion Leaks PII on "Millions" of Users. https://www.infosecurity-magazine.com/news/casino-app-clubillion-leaks-pii/.
[63] Pieroxy. 2013. lz-string: JavaScript compression, fast! https://pieroxy.net/blog/pages/lz-string/index.html.
[64] Tara Seals. 2018. DEF CON 2018: Telltale URLs Leak PII to Dozens of Third Parties. https://threatpost.com/def-con-2018-telltale-urls-leak-pii-to-dozens-of-third-parties/134960/
[65] Asuman Senol, Gunes Acar, Mathias Humbert, and Frederik Zuiderveen Borgesius. 2022. Leaky Forms: A Study of Email and Password Exfiltration Before Form Submission. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1813–1830. https://www.usenix.org/conference/usenixsecurity22/presentation/senol
[66] Hendrik Siewert, Martin Kretschmer, Marcus Niemietz, and Juraj Somorovsky. 2022. On the Security of Parsing Security-Relevant HTTP Headers in Modern Browsers. In *2022 IEEE Security and Privacy Workshops (SPW)*. 342–352. https://doi.org/10.1109/SPW54247.2022.9833880
[67] Oleksii Starov, Phillipa Gill, and Nick Nikiforakis. 2016. Are You Sure You Want to Contact Us? Quantifying the Leakage of PII via Website Contact Forms. *Proceedings on Privacy Enhancing Technologies* 2016, 1 (Jan. 2016), 20–33. https://doi.org/10.1515/popets-2015-0028
[68] Christof Ferreira Torres, Fiona Willi, and Shweta Shinde. 2023. Is Your Wallet Snitching On You? An Analysis on the Privacy Implications of Web3. In *32nd USENIX Security Symposium (USENIX Security 23)*. 769–786.
[69] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2020. Beyond the Front Page:Measuring Third Party Dynamics in the Field. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA, 1275–1286. https://doi.org/10.1145/3366423.3380203
[70] Pelayo Vallina, Victor Le Pochat, Álvaro Feal, Marius Paraschiv, Julien Gamba, Tim Burke, Oliver Hohlfeld, Juan Tapiador, and Narseo Vallina-Rodriguez. 2020. Mis-Shapes, Mistakes, Misfits: An Analysis of Domain Classification Services. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Virtual Event USA, 598–618. https://doi.org/10.1145/3419394.3423660
[71] Thomas van Ouwerkerk, Eelco Herder, and Gunes Acar. 2022. *Evading the Policy: A Measurement on Referrer Policy Circumvention in 3k e-Commerce Websites*. Master's thesis. Radboud University, Nijmegen.
[72] Huandong Wang, Chen Gao, Yong Li, Zhi-Li Zhang, and Depeng Jin. 2020. Revealing Physical World Privacy Leakage by Cyberspace Cookie Logs. *IEEE Transactions on Network and Service Management* 17, 4 (Dec. 2020), 2550–2566. https://doi.org/10.1109/TNSM.2020.3013335
[73] Logan Warberg, Vincent Lefrere, Cristobal Cheyre, and Alessandro Acquisti. 2023. Trends in Privacy Dialog Design after the GDPR: The Impact of Industry and Government Actions. In *Proceedings of the 22nd Workshop on Privacy in the Electronic Society*. ACM, Copenhagen Denmark, 107–121. https://doi.org/10.1145/3603216.3624963

Table 16: Conversion of legacy policy values to standard referrer policy values according to the HTML standard [43].

Legacy value	Referrer policy
`never`	`no-referrer`
`default`	`strict-origin-when-cross-origin`
`always`	`unsafe-url`
`origin-when-crossorigin`	`origin-when-cross-origin`

A APPENDICES

A.1 The Algorithm to Parse Referrer Policy on `<meta>` Elements

When a `<meta>` element is encountered within an HTML document, the user agent (web browser) executes a specific algorithm defined in the HTML standard [43] to determine what will be sent in the Referer header. The algorithm prioritizes explicit instructions provided by the `<meta>` element over the default behavior.

The algorithm follows these steps:

(1) **Element Validation:** It checks if the `<meta>` element is present within the document tree (i.e., part of the HTML structure). If not, it disregards the element.
(2) **Name Attribute Check:** It verifies if the element has a `name` attribute set to "referrer" (case-insensitive). If not, it moves on without processing the element.
(3) **Content Attribute Check:** It ensures the element has a `content` attribute and its value is not empty. If either condition is not met, the element is ignored.
(4) **Content Value Interpretation:** If all the above conditions are satisfied, the value of the `content` attribute is converted to lowercase ASCII characters for further processing.
(5) **Legacy Handling:** If the lowercase value is a legacy value, it is converted to a standard value according to the mapping given in Table 16.
(6) **Policy Mapping:** Finally, the algorithm compares the policy value to a list of recognized referrer policies. If the policy is recognized, it is set for the entire document.

A.2 Failed Visit Detection

Building on Le Pochat et al.'s method [54], we categorize a visit as failed if one of the following is true: (1) the initial request resulted in a 4xx or 5xx error code; (2) no successful (200 OK) response was received; (3) the first non-3xx response (root document) was less than 512 bytes in size; or (4) the landing page URL was "about:blank".

A.3 Supported Encoding Methods for Leak Detector

Base16, Base32, Base58, Base64, Urlencode, Entity, Deflate, Zlib, Gzip, LZstring, Custom Map
(`kibp8A4EWRMKHa7gvyz1dOPt6UI5xYD3nqhVwZBXfCcFe...`
`0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghi...`)

Table 17: Percentage of distinct websites applying a document-wide referrer policy under each consent-interaction mode. Document-wide referrer policies can be set by either meta-tags or RP response headers.

Referrer Policy	noAct	optOut	optIn
no-referrer	6.00%	6.00%	6.00%
same-origin	7.00%	7.00%	7.00%
strict-origin	3.00%	3.00%	3.00%
strict-origin-when-cross-origin	30.00%	30.00%	31.00%
origin	8.00%	8.00%	8.00%
origin-when-cross-origin	2.00%	2.00%	1.00%
no-referrer-when-downgrade	15.00%	15.00%	16.00%
unsafe-url	1.00%	1.00%	1.00%
Total	**51.00%**	**51.00%**	**53.00%**

B INTERACTING WITH CONSENT DIALOGS

In order to characterize how consent affects RP implementations and circumvention, we performed a small-scale crawl from Amsterdam—the only EU vantage point in our study. We first picked a random sample of 500 websites, of which our crawler detected 178 as displaying consent dialogs. To enhance the crawler's effectiveness, we filtered out generic consent dialogs and retained only those recognized by TRC's `autoconsent` [31], reducing the likelihood of unsuccessful interactions. After this refinement, 161 websites remained.

We randomly selected 100 websites from this subset and obtained the inner pages for those sites that we extracted earlier in the inner links collection crawl. Subsequently, we redeployed the crawler using the same IP address to perform three separate crawls with different interaction modes: no interaction (noAct), opting out (optOut), and opting in (optIn). Overall, 309 inner pages per consent mode were successfully crawled.

Table 17 summarizes the overall adoption rates of document-wide RP for each consent-interaction mode. We observe no meaningful difference across the crawls. Table 18 then examines how RP circumventions change in each consent mode, for different leak vectors (Referrer, POST body, and Request URL). Finally, Table 19 provides insights into which specific RP values were circumvented. In sum, the differences across consent modes remained minimal for both implementations and circumventions, with the exception of slight increase of circumventions in the optIn mode.

C ADDITIONAL RESULTS

Table 18: Referrer policy circumvention by consent mode and leak vector. Req: Distinct third-party requests; Web: Distinct websites; Total: Combined distinct number of requests or websites). More detailed measurements are given in Table 19.

Vector	noAct		optOut		optIn	
	Req	Web	Req	Web	Req	Web
Referrer	0.05%	1.00%	0.12%	2.00%	0.11%	6.00%
Post	3.09%	35.00%	2.60%	34.00%	2.89%	39.00%
URL	10.12%	70.00%	11.48%	72.00%	11.40%	77.00%
Total	**13.18%**	**72.00%**	**14.13%**	**74.00%**	**14.32%**	**79.00%**

Table 19: Distinct websites with specific referrer policies value that circumvented referrer policy while interacting with consent dialog. This table provides detailed information about the "web" portion of data found in Table 14. Clustered by leak vector: Referrer (R), Post body (P), and request URL (U)

	Referrer Policy	noAct	optOut	optIn
R	`no-referrer`	0	0	0
	`same-origin`	0	0	0
	`origin`	0	0	0
	`strict-origin`	0	0	0
	`strict-origin-when-cross-origin`	1	2	6
	`origin-when-cross-origin`	0	0	0
R	**Total**	**1**	**2**	**6**
P	`no-referrer`	0	0	0
	`same-origin`	0	0	0
	`origin`	1	1	1
	`strict-origin`	2	2	2
	`strict-origin-when-cross-origin`	32	31	36
	`origin-when-cross-origin`	1	1	1
P	**Total**	**35**	**34**	**39**
U	`no-referrer`	0	0	0
	`same-origin`	1	1	1
	`origin`	2	2	2
	`strict-origin`	2	2	2
	`strict-origin-when-cross-origin`	66	68	73
	`origin-when-cross-origin`	1	1	1
U	**Total**	**70**	**72**	**77**

Table 20: Legacy Referrer Policy found in Meta-tags. *: `none` would be ignored, while other legacy policies would be tranformed according to the rules given in Table 16.

	SF		Sg		Ams	
Legacy RP	**Count**	**% websites**	**Count**	**% websites**	**Count**	**% websites**
`origin-when-crossorigin`	1,069	3.90%	1,052	3.84%	534	1.95%
`always`	108	0.40%	108	0.40%	106	0.39%
`never`	15	0.06%	14	0.05%	14	0.05%
`none`	1	0.00%	1	0.00%	1	0.00%
`default`	1	0.00%	1	0.00%	1	0.00%

Table 21: Total distinct websites implementing element-specific referrer policies.

	Referrer Policy	**a**	**iframe**	**img**	**link**	**script**
SF	`no-referrer`	4	53	35	340	334
	`same-origin`	1	4	5	1	
	`strict-origin`		5		1	2
	`strict-origin-when-cross-origin`	1	10		7	148
	`origin`	96	125	104	167	129
	`origin-when-cross-origin`	1	2	10		2
	`no-referrer-when-downgrade`	21	318	48	3	278
	`unsafe-url`	2	537	9		294
Sg	`no-referrer`	4	84	20	339	333
	`same-origin`	1	4	5	1	
	`strict-origin`		5		1	2
	`strict-origin-when-cross-origin`	1	10		7	156
	`origin`	97	65	105	139	128
	`origin-when-cross-origin`		2	10	2	2
	`no-referrer-when-downgrade`	21	242	50	3	310
	`unsafe-url`	2	295	10		244
Ams	`no-referrer`	4	57	21	337	330
	`same-origin`	1	4	5	1	
	`strict-origin`		5		1	2
	`strict-origin-when-cross-origin`	1	9		6	147
	`origin`	100	60	108	126	133
	`origin-when-cross-origin`		2	10	2	2
	`no-referrer-when-downgrade`	22	237	44	3	240
	`unsafe-url`	2	15	9		167

Table 22: Total distinct websites implementing an invalid response referrer policy. These response referrer policies will be disregarded by the browser.

Response Referrer Policy	SF	Sg	Ams
strict-origin-when-cross-origin\nno-referrer-when-downgrade	57	59	59
strict-origin-when-cross-origin\nstrict-origin-when-cross-origin	30	32	32
same-origin\nsame-origin	24	24	24
no-referrer-when-downgrade\nno-referrer-when-downgrade	22	23	23
origin-when-cross-origin\norigin-when-cross-origin	11	11	12
nosniff	11	11	11
strict-origin-when-cross-origin\nsame-origin	9	9	9
strict-origin-when-cross-origin\n	7	7	6
no-referrer-when-downgrade\nstrict-origin-when-cross-origin	5	5	4
same-origin\nstrict-origin-when-cross-origin	5	5	5
strict-origin-when-cross-origin\norigin	5	5	6
no-referrer\nstrict-origin-when-cross-origin	4	4	5
strict-origin-when-cross-origin\nsame-origin\nsame-origin	3	3	3
strict-origin-when-cross-origin\nno-referrer	3	3	3
strict-origin-when-cross-origin\nstrict-origin	3	4	3
no-referrer-when-downgrade\nno-referrer	2	2	2
no-referrer\norigin-when-cross-origin	2	2	2
no-referrer-when-downgrade\n	2	2	2
no-referrer\nno-referrer-when-downgrade	2	2	2
no-referrer-when-downgrade\nsame-origin	2	2	2
origin-when-cross-origin\nno-referrer-when-downgrade	2	2	2
same-origin\nno-referrer-when-downgrade	2	2	2
strict-origin\nno-referrer-when-downgrade	1	1	1
unsafe-url\nunsafe-url	1	1	
same-origin\nno-referrer	1	1	1
no-referrer-when-downgrade\norigin-when-cross-origin	1	1	1
strict-origin-when-cross-origin\nunsafe-url	1	1	1
same-orign	1	1	1
strict-origin\nstrict-origin-when-cross-origin	1	1	1
no-referrer\nno-referrer	1	1	1
strict-orign-when-cross-origin	1	1	1
origin\norigin	1	1	1
*	1	1	1
origin\nstrict-origin	1	1	1
strict-origin\nno-referrer	1	1	1
strict-origin-when-cross-origin\nno-referrer-when-downgrade\nno-referrer-when-downgrade	1	1	1
strict-origin\norigin	1	1	1
origin\nstrict-origin-when-cross-origin	1	1	1
no-referrer-when-downgrade\nstrict-origin	1	1	1
strict-origin-when-cross-origin\norigin-when-cross-origin\norigin-when-cross-origin	1	1	1
use strict-origin-when-cross-origin	1	1	1
same-origin\nno-referrer-when-downgrade\nno-referrer	1	1	1
same-origin\norigin-when-cross-origin\norigin-when-cross-origin	1	1	1

Table 23: Total distinct websites implementing a valid response referrer policy. For policies marked "multiple," the browser will choose the last valid option from right to left. For (blank), the browser will choose the default referrer policy.

	Response Referrer Policy	SF	Sg	Ams
Multiple	origin-when-cross-origin, strict-origin-when-cross-origin	245	247	234
	no-referrer, strict-origin-when-cross-origin	69	69	68
	no-referrer-when-downgrade, strict-origin-when-cross-origin	16	16	16
	origin,strict-origin-when-cross-origin	11	11	11
	origin, origin-when-cross-origin, strict-origin-when-cross-origin	5	5	
	no-referrer, same-origin	3	3	3
	same-origin,strict-origin-when-cross-origin	2	2	2
	strict-origin-when-cross-origin, no-referrer-when-downgrade	2	2	2
	no-referrer, no-referrer-when-downgrade	2	2	2
	origin, strict-origin-when-cross-origin	2	2	2
	no-referrer,same-origin,strict-origin-when-cross-origin	2	2	2
	no-referrer-when-downgrade,strict-origin-when-cross-origin, no-referrer, strict-origin	1	1	1
	no-referrer, strict-origin-when-cross-origin\norigin	1	1	1
	same-origin, strict-origin-when-cross-origin	1	1	1
	strict-origin, strict-origin-when-cross-origin	1	1	1
	strict-origin-when-cross-origin, strict-origin-when-cross-origin, strict-origin-when-cross-origin	1	1	1
	no-referrer-when-downgrade\nno-referrer-when-downgrade, strict-origin-when-cross-origin	1	1	1
	origin,unsafe-url	1	1	1
	no-referrer-when-downgrade, no-referrer-when-downgrade	1	1	1
	same-origin,no-referrer-when-downgrade	1	1	1
	origin, strict-origin	1	1	1
	no-referrer,no-referrer-when-downgrade,origin,origin-when-cross-origin,same-origin,strict-origin,strict-origin-when-cross-origin,unsafe-url\nstrict-origin-when-cross-origin	1	1	1
	no-referrer-when-downgrade,strict-origin-when-cross-origin	1	1	1
	origin,no-referrer-when-downgrade	1	1	1
	never, no-referrer	1	1	1
	no-referrer,strict-origin-when-cross-origin	1	1	1
Single	strict-origin-when-cross-origin	7036	6868	6059
	no-referrer-when-downgrade	2468	2441	2384
	same-origin	1852	1899	1818
	no-referrer	1404	1412	1358
	origin	1332	1288	1219
	origin-when-cross-origin	1167	1174	1100
	unsafe-url	984	488	184
	strict-origin	357	355	326
	(blank)	82	81	81

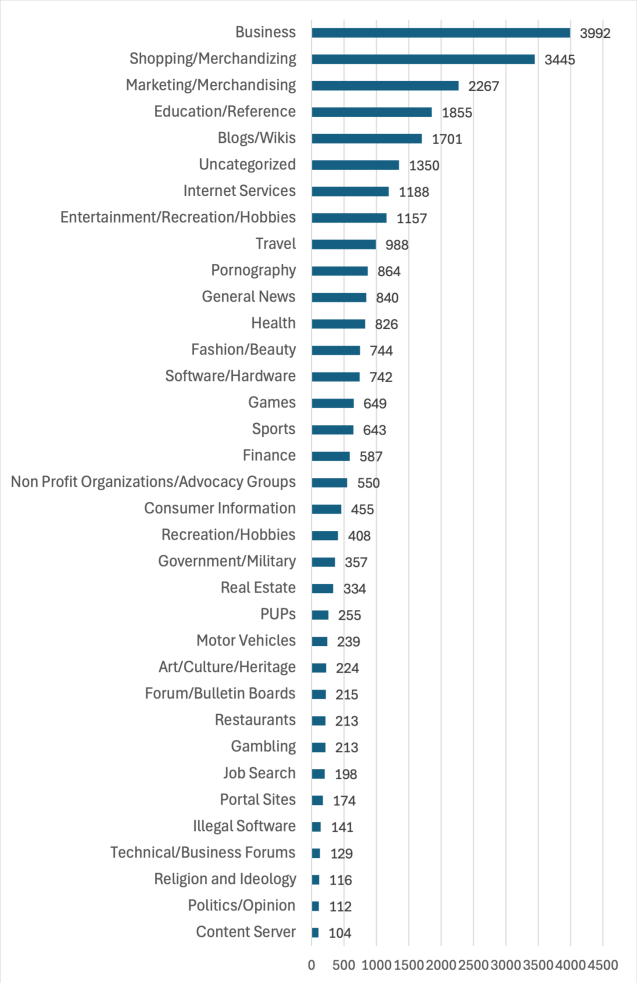

Figure 11: The number of studied websites in each category, excluding categories with fewer than 100 websites.

Table 24: Distinct websites with specific referrer policies value that circumvented referrer policy. This table provides detailed information about the "web" portion of data found in Table 14. Categorize in Referrer (R), Post body (P), and request URL (U)

	Referrer Policy	SF	Sg	Ams
R	no-referrer	11	8	10
R	same-origin	7	6	8
R	origin	2	3	3
R	strict-origin	2	1	1
R	strict-origin-when-cross-origin	2,443	2,192	1,360
R	origin-when-cross-origin	82	86	79
R	**Total**	**2,553**	**2,283**	**1,445**
P	no-referrer	143	139	124
P	same-origin	165	160	147
P	origin	322	318	325
P	strict-origin	50	50	45
P	strict-origin-when-cross-origin	8,484	8,365	7,558
P	origin-when-cross-origin	89	86	69
P	**Total**	**8,928**	**8,802**	**7,971**
U	no-referrer	227	222	216
U	same-origin	372	372	369
U	origin	291	234	226
U	strict-origin	108	106	103
U	strict-origin-when-cross-origin	19,801	19,750	19,480
U	origin-when-cross-origin	143	141	136
U	**Total**	**20,501**	**20,455**	**20,189**

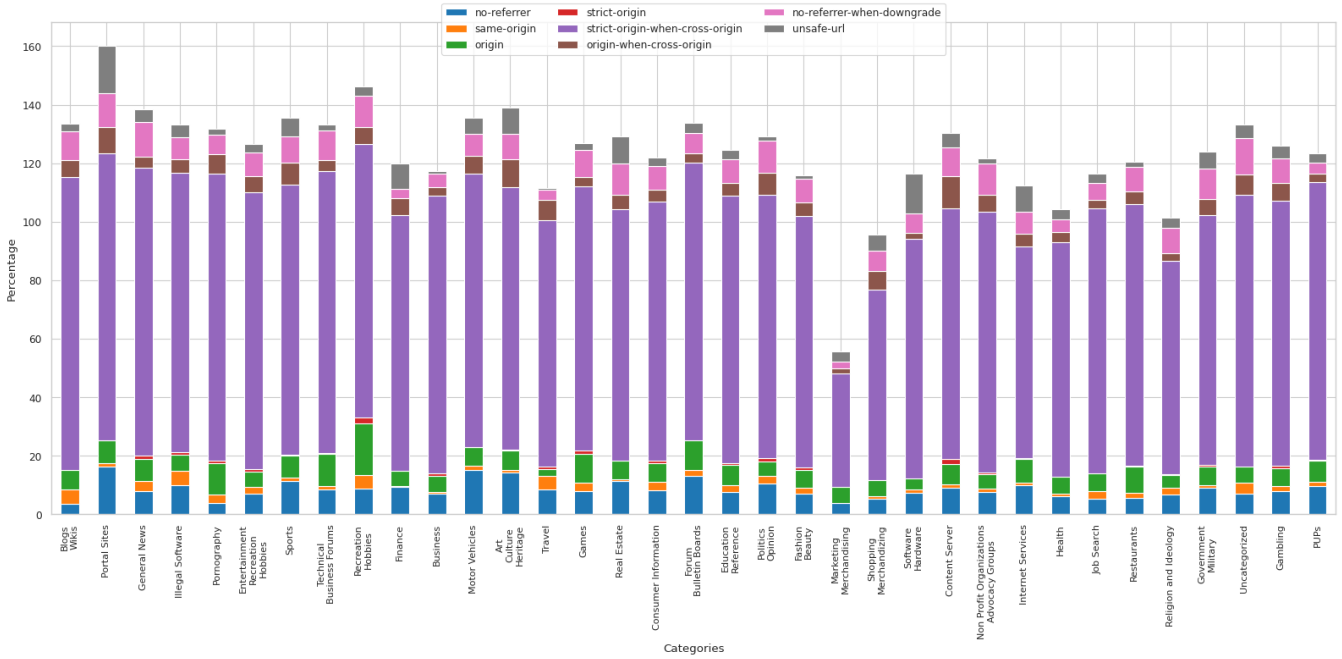

Figure 12: Comparison of website categories percentage by referrer policy values. This figure provides detailed information from Figure 6. Categories may not add to exactly 100% because requests observed on a single website may operate under different referrer policies due to, for example, element-specific referrer policies.