PURL: Safe and Effective Sanitization of Link Decoration

Shaoor Munir and Patrick Lee, *University of California, Davis;* Umar Iqbal, *Washington University in St. Louis;* Zubair Shafiq, *University of California, Davis;* Sandra Siby, *Imperial College London*

https://www.usenix.org/conference/usenixsecurity24/presentation/munir

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

PURL: Safe and Effective Sanitization of Link Decoration

Shaoor Munir
UC Davis
smunir@ucdavis.edu

Patrick Lee
UC Davis
pelee@ucdavis.edu

Umar Iqbal
Washington University in St. Louis
umar.iqbal@wustl.edu

Zubair Shafiq
UC Davis
zubair@ucdavis.edu

Sandra Siby
Imperial College London
s.siby@imperial.ac.uk

Abstract

While privacy-focused browsers have taken steps to block third-party cookies and mitigate browser fingerprinting, novel tracking techniques that can bypass existing countermeasures continue to emerge. Since trackers need to share information from the client-side to the server-side through link decoration regardless of the tracking technique they employ, a promising orthogonal approach is to detect and sanitize tracking information in decorated links. To this end, we present PURL (pronounced purel-l), a machine-learning approach that leverages a cross-layer graph representation of webpage execution to safely and effectively sanitize link decoration. Our evaluation shows that PURL significantly outperforms existing countermeasures in terms of accuracy and reducing website breakage while being robust to common evasion techniques. PURL's deployment on a sample of top-million websites shows that link decoration is abused for tracking on nearly three-quarters of the websites, often to share cookies, email addresses, and fingerprinting information.

1 Introduction

Web browsers and browser extensions are actively cracking down on new and emerging online tracking techniques. For example, almost all mainstream web browsers now already block or will soon block third-party cookies [1]–[3] and some privacy-focused browsers even deploy countermeasures against emerging tracking techniques, such as browser fingerprinting [4]–[6]. In response, online trackers continue to evolve and devise innovative tracking techniques that can bypass existing privacy-enhancing countermeasures [7]–[10]. The key limitation of existing privacy-enhancing tools is that they aim to mitigate specific types of tracking (e.g., third-party cookies, first-party cookies, email addresses, canvas fingerprinting, AudioContext fingerprinting, CNAME cloaking) [8], [11]–[13]. This approach works reasonably well to protect against known forms of tracking but fails against new or unknown forms of tracking, which routinely emerge from time to time [7], [10], [14], [15].

Our key insight is that trackers need to share information (e.g., user/device identifiers) from the client to the server side regardless of what type of tracking is employed. Therefore, we contend that a promising orthogonal approach to anti-tracking is to detect and block the sharing of tracking information in network requests. Trackers commonly include tracking information in "decorated" network request URLs (aka link decoration). Existing privacy-enhancing tools outright block network requests to endpoints using filter lists of known tracking services. However, a decorated link can include both tracking (e.g., user/device identifiers) and functional (e.g., CSRF tokens, identifying news/product subpages) information, which renders existing request blocking countermeasures ineffective – they risk breaking legitimate functionality as collateral damage if they block the request and risk allowing privacy-invasive tracking if they do not. This is fundamentally a granularity issue. To the best of our knowledge, there is an unmet need for a fine-grained approach that can precisely remove tracking information in decorated links.

To sanitize link decoration (i.e., precisely removing just the tracking information from a request URL), privacy-focused browsers and browser extensions such as Safari, Firefox, Brave, uBlock Origin, and AdGuard employ a manually-curated list of query parameters that are known to be abused for tracking [16]–[20], similar to the request-blocking filter lists such as EasyList and EasyPrivacy [21], [22]. While the manual approach might work for a small number of tracking query parameters, it cannot keep pace with the increasing adoption of query parameters by trackers [23], a problem that has also severely impacted filter lists [24]–[26]. Thus, it is not surprising that the filter lists to sanitize link decoration conservatively target only 10-100s of query parameters.

In this paper, we propose PURL (pronounced purel-l), a machine-learning approach to distinguish between tracking and functional link decorations. PURL makes use of a cross-layer graph representation to capture the complete execution of a webpage, which includes interactions between the HTML DOM structure, JavaScript execution behavior, information stored in browser storage, and network requests issued during

a webpage load. PURL then extracts distinguishing features from this rich graph representation and uses a supervised classifier to detect tracking link decorations.

Our evaluation on a sample of top-million websites shows that PURL can effectively (98.87% recall) and safely (98.62% precision) sanitize link decoration. Overall PURL achieves 98.74% accuracy, significantly outperforming existing countermeasures by at least 7.71% in terms of precision, 4.83% in terms of recall, and 6.43% overall accuracy while reducing website breakage by more than 8×. Our evaluation also shows that PURL is robust against common evasion attempts such as changing link decoration names and splitting/combining link decoration values.

We deploy PURL on a subset of top-million websites to measure the prevalence of link decorations for tracking. PURL detects that 73.02% of the sites abuse link decorations for tracking, with an average site using 10.75 tracking link decorations. We find that the most common abusers of link decoration include well-known advertising and tracking services, which use link decorations to share cookies, email addresses, and fingerprinting information.

Our key contributions are as follows:

1. We propose and evaluate **an automated machine learning approach**, called PURL, to detect tracking link decorations using features that capture interactions and flow of information across multiple layers of the web stack.

2. We **deploy PURL on top-million sites** to measure the prevalence, abusers, and type of tracking information shared via link decorations.

3. We **use PURL to generate a filter list**, which can and is already being used by privacy-focused browsers and browser extensions.

To promote reproducibility and facilitate further research, we are also open-sourcing PURL source code, including the OpenWPM patch, the machine learning pipeline, and the detected list of link decorations [27].

2 Background & Related Work

In this section, we discuss preliminaries, review related work, and survey existing countermeasures against the abuse of link decoration for tracking in industry and academia.

2.1 What is Link Decoration?

A URL is composed of the following key components: scheme, fully qualified domain name (FQDN), resource path, query parameters, and fragments. We use `https://a.site.example/YYY/ZZZ/pixel.jpg?ISBN=XXX&UID=ABC123#xyz`, as an example URL to define these segments below:

Base URL. `https://` is the scheme in this URL and `a.site.example` is the FQDN. These segments are combined to form the base URL.

Resource path. Immediately following the FQDN, `/YYY/ZZZ` is the resource path in the URL. It points to a directory, file, API endpoint, or other resource on the server. `pixel.jpg` is the name of the resource hosted on the server.

Query parameters. Immediately following the resource path after the ? delimiter, `ISBN=XXX&UID=abc123` are the query parameters in the URL. A query parameter consists of a key-value pair, where the key is separated from the value by the = delimiter. Multiple query parameters in a URL are separated from each other by the & delimiter.

Fragments. Immediately following the query parameters after the # delimiter, `xyz` is the fragment in the URL. Fragments can be a singular value or multiple key-value pairs that are separated by the & delimiter (similar to query parameters).[1]

While only query parameters are traditionally considered as link decoration [23], [28], we find that link decoration can be carried out using the resource path, query parameters, and fragments (discussed further in Section 3.1)[2]. Next, we describe the threat model we consider for tracking through link decoration.

2.2 Threat Model

Our threat model focuses on the sharing of identifying information through link decoration in third-party requests. We exclude first-party requests (i.e. requests sent from and to the first party) from our analysis, as we assume that first-parties would use identifying information to provide legitimate website functionality. However, our analysis includes first-party initiated requests sent to third-party domains, such as tracking via click identifiers used by social media and search engines. Our threat model considers the abuse of link decoration for both same-site and cross-site tracking because even same-site identifiers (e.g., first-party cookies [7], [8], [10]) can be combined with additional information for cross-site tracking. We consider two main entities in our threat model: the victim (users) and the adversary (third-party trackers). While third-party trackers require some cooperation from the website publisher (i.e., embedding a script on the page), we do not consider the publisher to be an active part of the threat model.

We assume that the victim/user:

• has third-party and first-party cookies enabled in the browser[3]

[1] In cases where fragments contain multiple key-value pairs, we treat these key-value pairs similarly as query parameters to facilitate easier comparison and analysis across different URL components.

[2] While fragments are not sent alongside a request to a server, the server can send a redirect to another page, which can access the fragments from the URL, e.g., through `window.location.hash`.

[3] We allow third-party cookies as they are supported in major browsers such as Chrome and Edge. However, PURL's approach is agnostic to using first- or third-party cookies for tracking.

• may provide personally identifiable information (PII) such as email address on the website (e.g., to log in)

We assume that the adversary/third-party tracker:

• may be present in a third-party context or a first-party context (i.e., a script embedded in the main frame) on the websites visited by the user

• aims to collect identifying information such as identifier cookies and email addresses for tracking

As described below, third-party trackers can use link decorations to share identifying information in several ways:

A user visits a website *website.example*, where tracker A is present in a first-party context (e.g., a script that sets a first-party identifier cookie) and tracker B is present in a third-party context (e.g., a pixel that sets a third-party identifier cookie). The user provides their email address on a form field that is accessible to only tracker A[4]. To send the email address to its server, tracker A has to append the email address as a link decoration in the request URL to its server. To send the first-party cookie to its server, tracker A again has to append the first-party cookie as a link decoration in the request URL to its server because first-party cookies would not be automatically sent to its third-party domain. Upon subsequent visits to the website by the same user, tracker A can associate its first-party cookie with the email address for same-site tracking, even though the user may not provide the email address in subsequent visits. Tracker A can also share the email address with tracker B by appending the email address as a link decoration in the request URL to tracker B's server. Tracker B can cross-site track the user with its third-party cookie, while also being able to associate it with the email address shared by tracker A.

2.3 Abuse of Link Decorations for Tracking

The abuse of link decoration for tracking is not a new phenomenon. To the best of our knowledge, the earliest evidence of link decoration abuse is from 1996, when Webtrends (an analytics service) used the `WT.mc_id` query parameter for click tracking in advertising campaigns [30], [31]. Since then, link decorations in general, and query parameters specifically, have been widely used for creating personalized links to track the success of advertising campaigns. For example, Urchin Tracking Module (UTM) parameters, such as `utm_source` identify the source of traffic on a website and attribute it to specific advertising campaigns [32]–[35].

Prior research has shown that trackers abuse link decoration to implement various tracking techniques [10], [36]–[42]. While prior work has proposed approaches to detect and block specific tracking techniques, which in turn rely on link decoration, these studies do not specifically study the abuse of link decorations for tracking. To the best of our knowledge, Randall *et al.* [23] is the first study to specifically study tracking query parameters. The authors found that 8.1% of the navigation URLs are decorated with identifiers as query parameters for tracking.

With new restrictions [1], [2], [43] on third-party cookies, trackers are moving towards alternative techniques of tracking, which include the use of first-party cookies [7], [8], [10], personally identifiable information (PII) [44], [45], and device/browser fingerprinting [46]–[48]. In contrast to third-party cookies that are automatically included as headers in outgoing HTTP requests, these alternative tracking techniques must rely on link decoration for sharing identifying information. As trackers shift their focus towards these alternative tracking techniques, it is reasonable to assume that the abuse of link decoration for tracking will also continue to increase.

2.4 Countermeasures Against the Abuse of Link Decorations for Tracking

Given the increased focus on alternative techniques to track users due to restrictions on third-party cookies, privacy-focused browsers, and browser extensions have started deploying countermeasures against the abuse of link decorations. These countermeasures can largely be divided into two different categories: filter list based countermeasures that rely on a static filter list of link decorations and heuristic-based countermeasures that rely on certain properties to identify tracking link decorations. Next, we describe existing countermeasures against tracking link decoration and highlight their limitations.

2.4.1 Filter List Based Countermeasures

Filter lists of link decorations contain both site-specific and site-agnostic rules which determine which link decorations should be allowed and which link decorations should be removed. These filter lists are manually curated and maintained, which has been shown by previous research to have issues such as slow updates and being error-prone [26], [49], [50]. As discussed below, filter list based countermeasures are used by both privacy-focused browsers and browser extensions.

Brave. Since July 2020 [51], Brave browser attempts to remove tracking query parameters from URLs by matching them against a list of known tracking query parameters. This filter list is curated by analyzing the documentation provided by the trackers themselves and from the reports submitted by Brave developers and users. At the time of writing, Brave's filter list of tracking query parameters contains 59 query parameters [18], [52].

Firefox. In January 2022, Firefox introduced the query parameter stripping feature [53] in Firefox Nightly 96.

Mozilla integrated this feature in Firefox 102.0 in June 2022 [54] although it was not enabled by default – Firefox users have to set *Strict* security level in Enhanced Tracking Prevention (ETP) to enable this feature. Firefox also allows users to remove tracking link decorations from copied URLs

[4] Prior research [10], [29] shows that email identifiers help trackers monitor user activity across websites

[55]. Similar to Brave, Firefox also relies on a curated filter list of tracking query parameters. At the time of writing, Firefox's filter list of tracking query parameters contains 23 query parameters [56].

Safari. Since June 2023, Safari 17 [57] removes known tracking query parameters in Safari's private browsing mode. Similar to Brave and Firefox, Safari also relies on a curated filter list of tracking query parameters. At the time of writing, Safari's filter list of tracking query parameters contains 24 query parameters [16].

AdGuard. AdGuard introduced a new filter type named `removeparam` to remove tracking query parameters from request URLs in 2021 [19]. At the time of writing, AdGuard's filter list includes more than one thousand query parameter rules [58]–[60].

uBlock Origin. uBlock Origin introduced new filter types `queryprune` in 2020 and then switched to `removeparam` [61]. Unlike AdGuard, uBlock Origin supports regular expression-based filters to remove tracking query parameters. At the time of writing, uBlock Origin includes 46 query parameter rules [62].

Requests-based filter lists. Filter lists such as EasyList [21] and EasyPrivacy [22], which are designed to block network requests to known trackers, would indirectly also block tracking link decorations. However, blocking the whole URL is not practical where tracking and non-tracking link decorations are mixed in the same URL. As we show later, using EasyList [21] and EasyPrivacy [22] results in non-trivial false positives and false negatives.

2.4.2 Heuristic Based Countermeasures

The aforementioned filter list based countermeasures are limited because they need to be manually created and updated. These limitations are apparent in their smaller size, with only one filter list including close to a thousand rules to detect tracking link decorations.

To address these issues, Randall *et al.* [23] proposed CrumbCruncher – a semi-automated heuristic-based approach to detect query parameters involved in the sharing of identifiers. CrumbCruncher conducts parallel and consecutive crawls to identify query parameters that are distinct across parallel crawls but persistent across consecutive crawls. However, their approach is prone to false positives and false negatives. Specifically, not all persistent parameters are predisposed to be tracking, e.g., parameters such as `share_button` or `en-US` remain consistent during multiple visits by the same user, however, they are not used for tracking. The manual review by the authors showed that CrumbCruncher suffers from a 36% false positive rate that needs to be addressed through a manual review. CrumbCruncher also suffers from false negatives because it incorrectly assumes that tracking query parameters are persistent across subsequent crawls. We find that almost 80% of potential identifiers (i.e., longer than 8 characters as defined by CrumbCruncher) that are shared to known tracking endpoints (defined using EasyList [21] and EasyPrivacy [22]) via query parameters change their value across consecutive crawls. For example, the `fbp` query parameter, which is used by Meta Pixel to share identifiers stored in the `_fbp` cookie, does not maintain its value in about 87% of the cases.[5] In summary, CrumbCruncher suffers from non-trivial false positives and false negatives due to its simplistic heuristic.

3 Motivating Measurements

In this section, we motivate the need for a tailored solution to curb the abuse of link decoration, such as PURL, by demonstrating that link decoration abuse is a prevalent phenomenon. Our key idea is to crawl a large set of websites and investigate the network requests, from known advertising and tracking services that appear on those websites, for link decoration. We exclusively focus on known advertising and tracking services because they are the main culprits who engage in such practices, and also because currently there are no current approaches to effectively detect link decoration abuse.

3.1 Methodology

Crawler configuration. We use OpenWPM (v0.17.0) [11] and Firefox (v102) [64] for crawling. Our crawls are stateless – i.e., we clear all cookies and other local browser states before crawling each website. By using stateless crawling, we ensure that each crawl is independent and not biased by the residual state from previous website crawls. We turn off all built-in tracking protections provided by Firefox (Enhanced Tracking Protection [ETP]) [65]. We conduct our crawls entirely from the vantage point of an academic institution in the US, hence, we do not interact with consent banners.

Websites crawled. We crawl a 20K sample of the Tranco top-million websites between March and April 2023 [66]. As previous research has highlighted the importance of making crawls representative of sites with varying popularity [67], we crawl all of the top-1K sites, uniformly sample another 9K sites from the sites ranked 1K–100K, and a further 10K from sites ranked 100K–1M. Additionally, to capture differing content on both the landing and internal pages [68], we perform an interactive crawl that covers both types of pages. Specifically, for each site, we crawl its landing page, randomly scroll and move the cursor (for bot mitigation), and then select up to 20 internal pages to visit at random. After each page load is complete (i.e., when the onLoad event is fired), we uniformly at random wait an additional 5–30 seconds for bot mitigation and other resources to finish loading. The success rate of our crawler is 98.79%. A tiny fraction of web pages do not load correctly because of server-side errors.

[5] Bekos *et al.* [63] showed that the value of the `_fbp` cookie (and consequently the `fbp` query parameter) is randomly chosen from a list of up to 50 different identifiers, resulting in a new identifier for the same user each time.

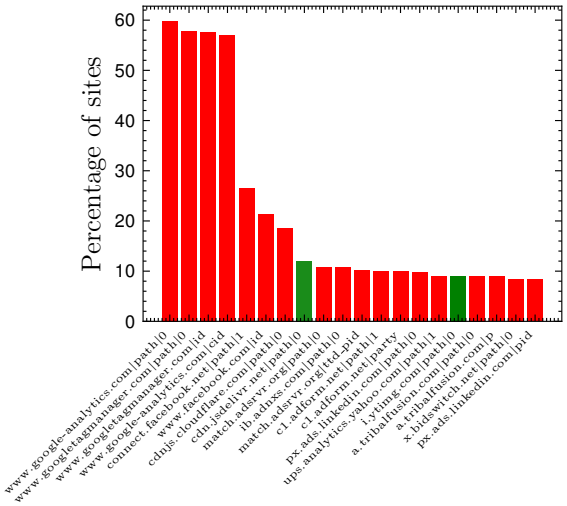

Figure 1: Percentage of sites where the same link decoration by top domain appears and their primary usage. The shades of red and green show link decoration's usage as ATS and Non-ATS, respectively.

Labeling tracking requests. To analyze the prevalence of link decoration in tracking requests, we use EasyList [21] and EasyPrivacy [22]. Specifically, we use them to label requests as Advertising and Tracking Service (ATS) or non-Advertising and Tracking Service (Non-ATS). We label a request as ATS if its URL matches the rules in either one of the lists. Otherwise, we label it as Non-ATS.

Naming link decorations. When link decorations are in the key-value format, the key can simply be combined with FQDN[6] to uniquely identify a link decoration. For example, if a link decoration with key *username* is sent to an FQDN *site.example.com*, *site.example.com+username* can be used to identify the link decoration. When link decorations are not in the key-value format (e.g., resource paths and fragments), we assign them keys based on the FQDN and their position in the URL. We identify link decorations for resource paths based on their distance (directory levels) from the root. For the example URL: `https://a.site.example/YYY/ZZZ/pixel.jpg?ISBN=ABC&UID=DEF123#xyz`, we identify the following link decorations as key-value pairs:

- `a.site.example` | $path_0$: `YYY`
- `a.site.example` | $path_1$: `ZZZ`
- `a.site.example` | *ISBN*: `ABC`
- `a.site.example` | *UID*: `DEF123`
- `a.site.example` | *fragment*: `xyz`[7]

This naming scheme allows us to compare link decoration values across different URLs.

[6] We combine FQDN with the link decoration key because different FQDNs can use the same key names.

[7] If fragments are in the key-value format like query parameters, we treat them similarly as query parameters.

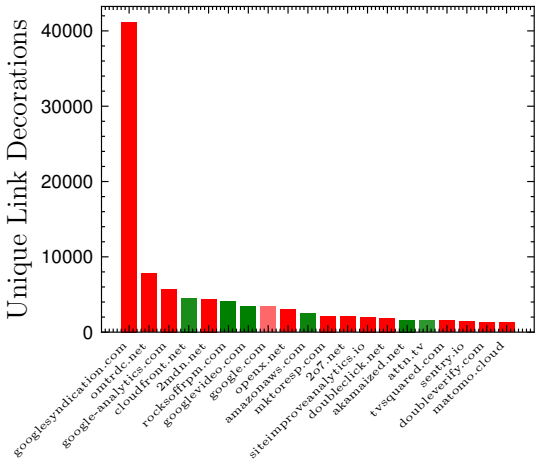

Figure 2: Total unique link decorations used by domains. The shades of red and green show link decoration's usage as ATS and Non-ATS, respectively.

3.2 Prevalence of Link Decoration

We investigate the prevalence of link decoration used on the 20K sample of the top-million websites. Of the 44,648,436 link decorations in our data, 41.22% are query parameters, 58.14% are resource paths, and 0.63% are fragments. Considering only unique link decorations, we observe a total of 584,174 decorations: 42.41% of which are query parameters, 53.85% are resource paths, and 3.73% are fragments.

Overall, 45.55% unique link decorations are in the URLs labeled as ATS, while the rest are sent to Non-ATS endpoints. We find that requests sent to ATS endpoints disproportionately contain more decorations on average (7.69) than the requests sent to Non-ATS endpoints (4.68), highlighting that link decorations are more frequently used by ATS than Non-ATS. A similar trend holds for third-party requests (4.80) vs. first-party endpoints (2.10).

We further find that the same link decorations are widely reused (use of the same link decoration key/name on more than one site) by advertising and tracking services. Figure 1 shows the top 20 link decorations and their prevalence in our dataset.[8] The color of the bar shows the usage of link decoration for either ATS or Non-ATS purpose (i.e., the red represents ATS and green represents Non-ATS). The plot shows that while both ATS and Non-ATS services show reuse of the same link decoration across multiple websites, it is ATS who predominately exhibit this behavior. For example, `www.googletagmanager.com|id` is used in around 55% of sites in our dataset and is primarily used in ATS requests. On the other hand, the most commonly used Non-ATS link decoration is `cdnjs.cloudflare.com|path|0`, which is found on slightly more than 10% of sites in our dataset.

[8] To simplify the illustration due to space constraints, we limit link decorations for each FQDN to only the top two.

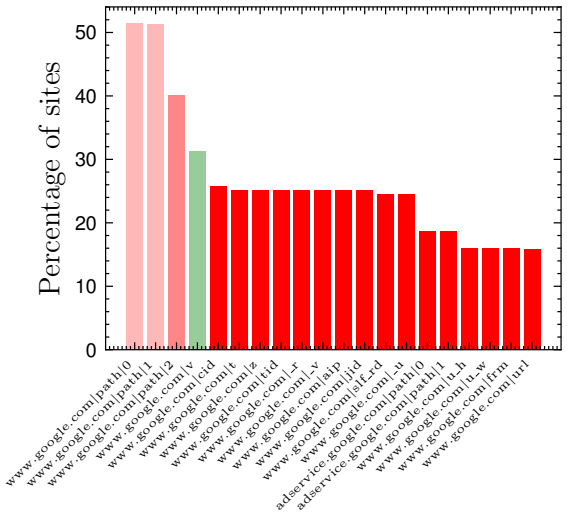

Figure 3: Average number of link decorations used by Google endpoints (minimum 1000 requests across 20K sites)

Next, we analyze the use of link decoration by ATS. Figure 2 plots top-20 tracking domains based on the number of unique link decorations they use. The intensity of the color (green for Non-ATS and red for ATS) in the figure for each domain shows its use of link decorations in ATS or Non-ATS requests. We note that *googlesyndication.com*, which is used by Google Ad Manager [69], uses the highest number of unique link decorations among the ATS domains. It is followed by *omtrdc.net*, which is used by Adobe Marketing Cloud [70], *google-analytics.com*, which aggregates and reports user stats for sites [71], and *cloudfront.net*, which is an Amazon-owned content delivery network [72]. Other well-known ATS such as Facebook, Baidu, and Microsoft are also among the domains that use the most link decorations in their requests. Our main finding is that link decorations are widely used by well-known advertising and tracking services. Crucially, Figure 2 also shows some mixed usage of link decorations. For example, link decorations used by *google.com* were part of 122,028 ATS requests and 56,026 Non-ATS requests (this is represented by a lighter shade of red in the figure as compared to other domains which are either darker shades or red or green). On the other hand, link decorations used by *amazonaws.com* were part of 863 ATS requests and 16,420 Non-ATS requests (represented by a darker shade of green in Figure 2).

To identify the reason behind significant mixed usage of link decorations by *google.com*, we take a closer look at different link decorations used by *google.com* and their prevalence in our dataset. Figure 3 plots the prevalence of the top 20 link decorations used by *google.com*. The color of each link decoration represents its use as ATS and Non-ATS, with higher shades of red representing predominant use in ATS requests and higher shades of green representing predominant use in Non-ATS requests. The top four link decorations have significant mixed usage between ATS and Non-ATS requests, with the top three: `www.google.com|path|0`, `www.google.com|path|1`, and `www.google.com|path|2` leaning towards more ATS while `www.google.com|v` slightly leaning towards Non-ATS.

```
http://go.artinstitutes.edu/search/brand/local/PGC?source
=BGNAG&ven=search&Tac=sem&school=newyork&Matchtype=Exact&
gclid=KjwKEAjwq6m3BRsdfdfsdfCP7IfMq6Oo9gsdfACRc0bN3J-fcQ1
t1DdfO5AyuTfKIyFbgTFPfCmPXyGdrKRBoCmv3w_wcB
```

Figure 4: Example URL with mixed link decorations. ■ indicates a Non-ATS link decoration while ■ indicates an ATS link decoration. Resource paths are highlighted as green and the query parameters with keys `source`, `ven`, `Tac`, `school`, and `Matchtype` are used for functional purposes, while `gclid` contains an identifier that is used to track ad clicks.

These results show that a single link decoration can be part of both ATS and Non-ATS requests. Next, we try to evaluate if a single request can also have both ATS and Non-ATS link decorations. To this end, we make use of query parameter filter lists used by Brave[18], Firefox[17], and Safari[16], as well as privacy-enhancing extensions uBlock Origin[20] and AdGuard[58], [59]. We label every link decoration not included in these filter lists as a Non-ATS link decoration. Overall, we observe 51,736 requests that contain one or more ATS link decoration, while only 248 of these requests contain no other Non-ATS link decoration. On average, an ATS link decoration is accompanied by 16.06 Non-ATS link decorations in the same request URL. An example of such mixed URLs is shown in Figure 4.

Takeaway. Our measurements show that a request URL can contain both ATS and Non-ATS link decorations. Moreover, the classification of a link decoration can change depending on which site it is used on and its destination domain. Thus, as we show later in Section 4.3, it is non-trivial to detect and block ATS link decorations using existing countermeasures.

4 PURL

In this section, we present PURL (pronounced purel-l), our machine learning approach to detect ATS link decorations. PURL's key idea is to use the execution traces of ATS link decorations as their signatures, which it learns and automatically detects with the help of a machine learning (ML) classifier. PURL captures detailed execution traces across the HTML, network, JavaScript, and storage layers of the web stack and models them in a graph representation. The graph representation captures the natural interaction between different layers of the web stack and provides a parse-able representation to extract various characteristics (i.e., features) of ATS link decoration execution, that are used to train a supervised ML classifier. Figure 5 provides an overview of PURL's design.

4.1 Design and Implementation

Browser instrumentation. PURL extends OpenWPM [11], an open-source web measurement tool, to record the execu-

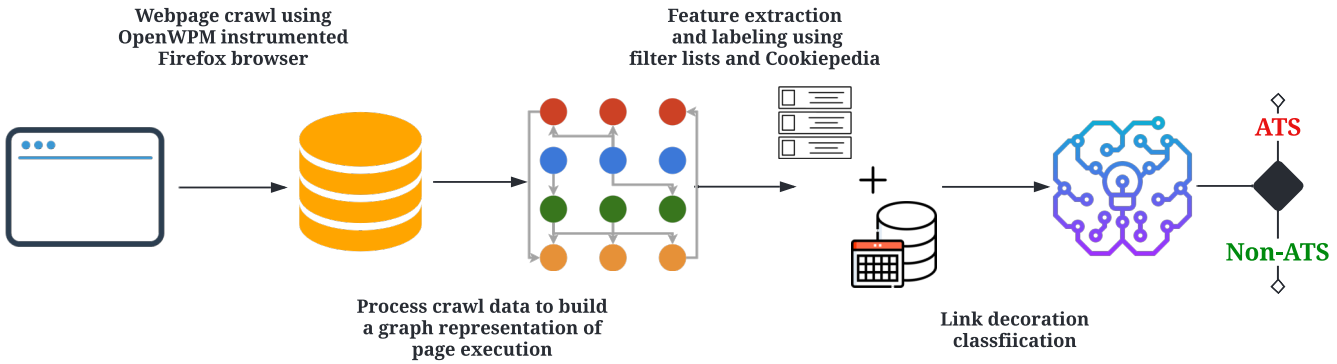

Figure 5: Overview of PURL pipeline: (1) Webpage crawl using an instrumented browser; (2) Construction of a graph representation to represent the instrumented webpage execution information; (3) Feature extraction for graph nodes that represent link decorations; and (4) Classifier training to separate out ATS and Non-ATS link decorations.

tion of a webpage across *HTML*, *network*, *JavaScript*, and *storage* layers during a webpage load. Similar to prior work on tracking detection (e.g., [10], [73], [74]), PURL captures the HTML elements created by scripts, network requests sent by scripts and HTML elements, responses received to these requests, sharing of identifiers stored in storage (local storage and cookies), and other read/write operations on storage mechanisms present in the browser.

PURL improves upon previous work by creating a more granular representation of the network layer of a webpage, which is essential for capturing characteristics of link decoration. Specifically, instead of coarsely capturing network requests and responses, PURL breaks them down and captures granular components of link decorations. For example, instead of identifying that a request contains a cookie value, PURL identifies the exact link decoration that was used to share that cookie value.

Graph Construction. There are five types of nodes in PURL's graph representation: *storage*, *HTML*, *script*, *network*, and *decoration*. *Storage* nodes refer to information stored in cookies and localStorage.*HTML* nodes are HTML Document Object Model (DOM) elements on a webpage. *Script* nodes map interactions of JavaScript execution on a webpage. *Network* nodes represent outgoing HTTP requests and incoming HTTP responses from network endpoints. *Decoration* nodes are created by splitting each *network* node into its link decorations.

We capture read and write operations performed on information in browser storage by different scripts, their sharing through network requests, and the setting of browser storage through network responses. We capture the interaction of different scripts with HTML elements and also map requests generated through HTML elements. In addition to these interactions, PURL captures actions and attributes that are specific to link decorations. Since link decorations may be used to share information stored in browser storage and the network responses received as a result of this sharing may be used to set browser storage elements using HTTP headers or JavaScript APIs, we monitor Base64-, MD5-, SHA-1-, and SHA-256-encoded[9] storage node values in decorations to associate relevant interactions between decoration, storage, network, and script nodes.

Figure 6 and 7 show how PURL constructs a graph representation for an example scenario where a script is reading/writing to browser storage and sending requests including link decorations to tracking sources. The nodes in the given example graph are storage, script, request, and decoration. The numbers on the edges represent a particular action, as represented in Figure 6. Dotted and dashed lines respectively show the flow of information from storage to decoration nodes (exfiltration) and the flow of information from request nodes to storage nodes (infiltration). PURL links the outward flow of information (exfiltration) to decoration nodes, and also maps the inward flow of information (infiltration) from a parent request/response node to the storage node. We use this graph structure to calculate features for decoration nodes that represent this flow of information.

Feature extraction. PURL leverages the graph representation to extract three different types of features that capture the execution traces of link decoration, referred to as *structural*, *flow*, and *content* features.

Structural features map the relationship of different nodes in the graph with each other, such as the connectivity of nodes and their ancestry information. For example, the connectivity of nodes captures how many different link decorations are in a request. As per Section 3, on average, ATSes use far more link decorations than Non-ATSes, resulting in stronger connectivity for ATS link decorations, which is also reflected in their structural feature values. Structural features help encode this information for our classifier using graph properties such as centrality, connectivity, and closeness[75], [76].

Flow features represent information flow across different layers of a webpage. Capturing this flow of information is important to track how user information is, extracted, stored,

[9] PURL can be extended to support other encodings.

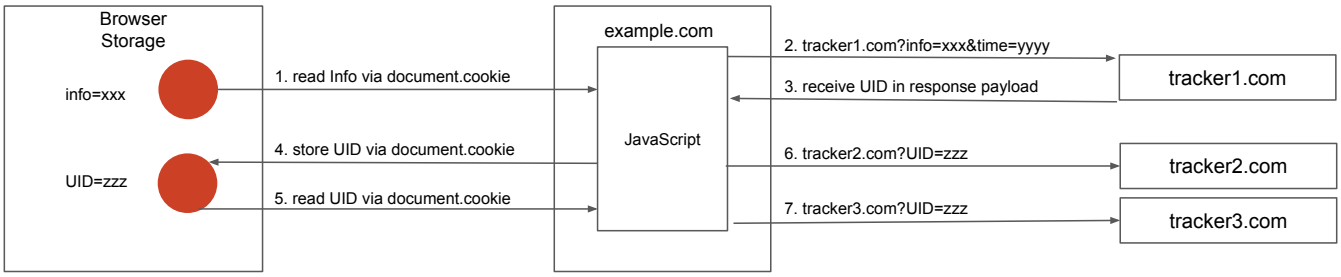

Figure 6: Example scenario to illustrate PURL's graph construction (shown in Figure 7). (1) A script on example.com reads info cookie from browser storage using `document.cookie`. (2) The script sends a network request to *tracker1.com* which includes the info cookie value and the current time in the decorated link. (3) *tracker1.com* sends a network response that contains UID in the response payload. (4) The script stores UID in the browser storage using `document.cookie`. (5) The script reads UID using `document.cookie` (6,7) The script sends the UID to *tracker2.com* and *tracker3.com* as decorated links.

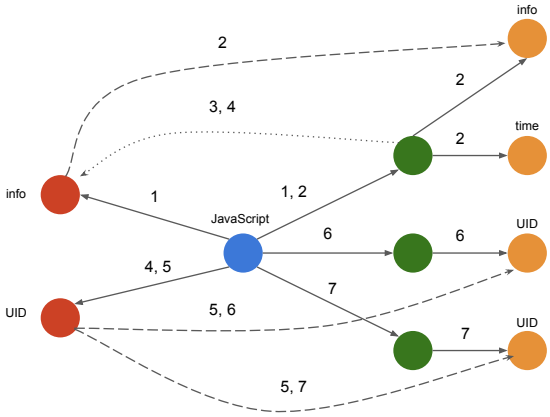

Figure 7: Graph representation of Figure 6 in PURL. ● network nodes, ● script nodes, ● storage nodes, and ● represent decoration nodes. The solid lines show the interactions of the script nodes with the storage and request nodes, while the dashed (_ _ _) and dotted (...) lines represent the flow edges captured by PURL from a storage node to a decoration node and from a network node to a storage node, respectively.

and sent out through tracking link decorations. First, PURL captures the direct sharing of storage values through the link decoration. As described in Figure 6, storage values can directly be sent out using link decorations. PURL represents this sharing as additional edges from storage nodes to decoration nodes. Second, PURL also keeps track of whether the parent request of a link decoration results in the setting of a storage node. This inward flow of information (infiltration) is usually used by ATS to set identifiers based on information sent out in requests (mainly through link decorations) [10]. PURL maps this infiltration through indirect edges connecting the parent request/response of decoration with the corresponding storage node. In addition, PURL monitors if the script sending the parent request of decoration is involved in sending storage information in non-parent requests or is part of redirects [37].

To determine the suitability of a decoration as a potential identifier, PURL also computes *content* features such as character-level Shannon entropy [77] and the relative position of the decoration in the URL. The complete list of the features used by PURL and their analysis is in the appendix.

Ground Truth Labeling. Once we capture the execution traces, we need to label them before they can be used to train a classifier. However, as discussed in Section 2, there are currently no readily available sources that can be reliably used to label link decoration abuse. Thus, we create our own set of labels by combining three different sources, which are: (i) filter lists of known advertising and tracking sources, (ii) a database of known tracking cookies, and (iii) short lists of manually curated tracking query parameters. Recall from Section 3.1 that each link decoration instance is a combination of the site where it appeared and the decoration key, and it is labeled as such. Next, we describe our ground truth labeling process that leverages the three aforementioned approaches.

1) Filter lists. Filter lists, such as EasyList and EasyPrivacy [21], [22], are the most reliable sources to identify tracking, which are used by almost all privacy-enhancing tools. However, their detection granularity is at the level of a URL and thus cannot be directly used to label individual parameters as ATS in a URL. This is because even a URL detected as ATS by filter lists might contain both tracking and non-tracking parameters [78], [79]. Despite this problem, filter lists can still be used to identify benign parameters, i.e., the parameters found in URLs from Non-ATS services, and we use them as such. Specifically, we rely on EasyList [21] and EasyPrivacy [22] filter lists to first identify Non-ATS URLs and then label all parameters in them as Non-ATS. As shown in Section 3.2, a single URL can contain both ATS and Non-ATS link decorations. In such cases, labeling all link decorations in URLs that are not blocked by filter lists as Non-ATS will result in incorrect labels for ATS link decorations. To account for this, we re-label all Non-ATS link decorations identified in this step as ATS if they are found to be involved in tracking in the next steps.

2) Cookiepedia. Link decoration can contain values stored in cookies, which are traditionally used to store and share user

identifiers [7], [10]. To identify such link decorations that can be used to exfiltrate tracking cookies, we make use of Cookiepedia [80], which is a database of cookies maintained by a well-known Consent Management Platform (CMP) called OneTrust [42], [81]. Primarily, Cookiepedia provides the purpose of each cookie in its database through its integration with OneTrust. Each cookie is provided one of the four labels: strictly necessary, functional, analytics, and advertising/tracking. We monitor the sharing of all cookies labeled as either analytics or advertising/tracking in Cookiepedia through a link decoration and label those link decorations as ATS.

3) Manually Curated Lists. Recall from Section 2 that several privacy-enhancing browsers and extensions maintain lists of known ATS link decorations parameters. Despite these lists being limited, they contain popular query parameters that are manually vetted to be used for tracking. Thus, we also make use of ATS link decoration lists, maintained by Brave [82], Firefox [17], AdGuard [83], and uBlock Origin [20].

Using a combination of these techniques, we were able to label 18.76% of our dataset, with 1.40% (60,573 instances) being labeled as ATS and 17.36% (749,553) as Non-ATS. Even though the ground truth is limited, especially for ATS samples, we argue that it is a significant improvement over the existing countermeasures. As described in Section 2.4, prior work has identified only a handful (maximum of around one thousand) ATS link decorations. Additionally, mislabelling link decorations can result in significant website breakage, necessitating a high-precision, albeit limited, ground truth.

Classifier. After curating the ground truth, we next train a supervised classifier to detect ATS link decoration. We use a random forest ensemble classifier because it is tolerant against noisy labeled data, is efficient to train, and is interpretable [84], [85]. We train the model using a balanced set of ATS and Non-ATS link decoration samples. We evaluate the accuracy of our classifier using stratified 10-fold cross-validation, to ensure that we do not train and test on the same samples. Overall, our classifier achieves 98.74% accuracy, 98.62% precision, and 98.87% recall, indicating that it is successful in detecting ATS link decorations.

The classifier accuracy is also comparable across different types of link decorations. For resource paths, it achieves an accuracy of 99.36%, 99.03% precision, and 99.69% recall. For query parameters, the accuracy is 96.40%, precision is 93.39%, and recall is 99.87%. Finally, for fragments the accuracy is 99.33%, precision is 98.75%, and recall is 100%.

4.2 Analysis of Disagreements between PURL and Ground Truth

We manually analyze PURL's false positives and false negatives to assess whether these are actual mistakes or limitations of our ground truth (recall that we curated a high-precision, albeit limited, ground truth).

First, we analyze the false positives of PURL. We manually verify the most common false positives in our dataset by first, analyzing information sent by false positive link decorations (user identifiers, values stored in cookies, etc.), and second, by analyzing available online documentation by senders/receivers of these link decorations. Our analysis of the most commonly misclassified ATS link decorations reveals that most of them are indeed used for tracking. In total, PURL classifies 8,058 Non-ATS instances (out of 749,553 total, 1.07%) as ATS, which correspond to 2,994 unique link decorations. The three most common false positive link decorations include `utk` query parameter sent to *hubspot.com*, `bsi` query parameter sent to *frog.wix.com*, and `iiqpciddate` query parameter sent to *api.intentiq.com*. We manually analyzed these three link decorations, which account for almost 10% of all false positives. Our analysis of the documentation for these three link decorations shows that these are not false positives, but rather these were falsely labeled as Non-ATS in our ground truth, and PURL actually correctly classified them as ATS. `utk` query parameter contains the HubSpot's `hubspotutk` cookie which is used to identify a user visiting a website [86]; `bsi` query parameter contains the identifier used by BSI's Customer Data Platform [87], while `iiqpciddate` accompanies `iiqpcid` query parameter which is used by IntentIQ to uniquely identify a user [88]. We conclude that PURL's false positives are, in actuality, false negatives in the ground truth, which was curated conservatively to be highly precise (rather than high recall).

Second, we analyze the false negatives of PURL. In total, PURL incorrectly classifies only 50 ATS instances as negatives (out of 60,573 total, 0.08%). In all of these instances, PURL was unable to link the sharing of stored information through these link decorations. In addition to this, values of structural features (e.g., number of edges) for these instances were also lower than true positive instances (e.g. 11,501.99 number of edges as compared to 14,762.86 for true positives). We conclude that false negatives happen when PURL is unable to trace certain tracking behaviors of ATS link decorations. We elaborate on PURL's implementation limitations in tracing storage sharing in Section 6.

Beyond these disagreements, PURL is more than the sum of its ground truth. Concretely, PURL detects 52,489 ATS link decorations that are not detected by Cookiepedia, EasyList [21], EasyPrivacy [22], or the manually curated filter lists for ATS link decorations.

4.3 Comparison with Existing Countermeasures

In this section, we compare PURL versus existing countermeasures against ATS link decorations to demonstrate that it significantly advances the state-of-the-art. We compare it against approaches that directly detect ATS link decoration and also approaches that were originally designed to detect ATS link decoration but can be repurposed to detect them.

We compare PURL against CrumbCruncher [23] and link decoration based filter lists, which are designed to detect ATS link decoration. For comparison with CrumbCruncher, we rely on the list of ATS query parameters published by Randall *et al.* [89]. For comparison with link decoration based filter lists, we rely on the lists offered by Brave [82], Firefox [17], Safari [16], uBlock Origin [20], and AdGuard [83]).

Additionally, we also compare PURL against Cookiepedia [80] and request-based filter lists (i.e., EasyList [21] and EasyPrivacy [22]), which we repurpose to detect ATS link decoration. We compare with Cookiepedia cookie labels by identifying link decorations that are used to exfiltrate values of cookies labeled as ATS by Cookiepedia. For comparison with request-based filter lists, we consider all link decorations in the requests labeled as ATS by filter lists to be ATS link decorations. We compare these approaches across two axes: (i) accuracy and (ii) breakage.

Accuracy. Table 1 shows the comparison of accuracy, precision, and recall of all four countermeasures against PURL. It can be seen from the table that PURL outperforms the runner-up countermeasure (i.e., request-based filter lists) by 6.43% in terms of accuracy and by 7.71% in terms of precision. Some of the most common ATS link decorations detected by PURL and missed by the runner-up countermeasure are `sync.intentiq.com|pcid`, `pr-bh.ybp.yahoo.com|path|2`, and `partner.mediawallahscript.com|uid`. As we discuss next, the higher precision results in a measurable reduction in website breakage when PURL is used as compared to runner-up countermeasures.

Classifier	Accuracy	Precision	Recall
PURL	98.74%	98.62%	98.87%
CrumbCruncher	50.16%	59.09%	10.67%
Cookiepedia	80.99%	99.01%	62.63%
Filter lists (Requests)	92.31%	90.91%	94.04%
Filter lists (Decorations)	50.50%	100.0%	10.15%

Table 1: Classification accuracy of PURL, CrumbCruncher, Request Filter lists, Decoration Filter lists, and Cookiepedia

Classifier	Navigation		SSO		Appearance		Miscellaneous	
	Minor	Major	Minor	Major	Minor	Major	Minor	Major
PURL	0%	0%	2%	0%	2%	1%	2%	0%
Filter lists	0%	2%	1%	2%	8%	6%	4%	1%

Table 2: Website breakage comparison of all three countermeasures.(▬) signifies no breakage, (▬) minor breakage, and (▬) major breakage. Each cell represents the percentage of sites on which breakage was observed.

Website Breakage. To determine what effect each countermeasure has on the usability of a site, we compare website breakage caused by PURL and the countermeasure with the second-highest accuracy and recall (request-based filter lists) on 100 sites. We use 20K sites from section 3, out of which 50 sites were sampled from those ranked 1-1000 and 50 from the rest to ensure the sample is representative. A list of sites used for breakage analysis is available at [90]. Our breakage analysis is divided into four different categories of how a website is used: navigation (moving from one page of the website to another), SSO (third-party login integrations), appearance (visual consistency and acuity), and miscellaneous (additional functionality such as shopping carts, chatbots, etc.). We also categorize each breakage into minor or major, the former implying that the underlying functionality is disrupted but still usable, and the latter implying that the functionality is completely unusable from the user's perspective. Two reviewers interact with the webpage while ATS link decorations detected by PURL or filter lists are removed. Any disagreements between the two reviewers are resolved after careful discussion.

Our evaluation shows that out of the 100 sites, PURL caused minor breakage on 5 sites and major breakage due to a failure to load CSS on *autodesk.com*[10]. On the other hand, requests-based filter lists caused minor breakage on 12 different sites and major breakage on 9 sites, including a complete breakdown of navigation on *directunlocks.com* and CSS issues on *engadget.com* and *mysuncoast.com*.

These results show that PURL not only significantly outperforms existing countermeasures in terms of accurately sanitizing more ATS link decorations, but it also does so without causing additional website breakage.

5 Deployment

In this section, we deploy PURL on a 20K sample of top-million sites to understand the prevalence and the nature of information shared in ATS link decorations.

5.1 Prevalence of ATS Link Decorations

We first analyze the breakdown of PURL's classification of link decorations. PURL classifies 4.56% (196,890) of link decorations in our dataset as ATS. 73.02% (14,604) of the tested sites contain at least one request with an ATS link decoration. Overall, an average site employs 10.75 ATS and 44.59 Non-ATS link decorations. Table 3 provides the breakdown of different types of ATS and Non-ATS link decorations. Out of the 196,890 link decorations labeled as ATS, 6.62% are resource paths, 93.36% are query parameters, and 0.02% are fragments. Our findings indicate that while query parameters account for the majority of ATS link decorations, trackers also abuse resource paths and fragments that are ignored in prior work [23](e.g., CMID cookie set by Casalemedia and shared through resource path in request to Yahoo: *https://pr-bh.ybp.yahoo.com/sync/casale/**ZBlPAJ28Qf7u...***).

Table 4 lists the top-50 most prevalent ATS link decorations. We note that a majority of the top ATS link decorations are used by various Google advertising and tracking endpoints

[10] PURL misclassified the second resource path in *https://static-dc.autodesk.net/etc.clientlibs/**v605.20230316.1511**/autodesk/clientlibs/clientlib-dhig.min.css*

	Resource Paths	Query Parameters	Fragments	Total
ATS	13,030 (6.62%)	183,813 (93.36%)	47 (0.02%)	196,890 (100.00%)
Non-ATS	1,295,246 (31.42%)	2,824,549 (68.52%)	2,233 (0.05%)	4,122,028 (100.00%)
Total	1,308,276 (30.29%)	3,008,362 (69.65%)	2,280 (0.05%)	4,318,918 (100.00%)

Table 3: Distribution of link decorations between ATS and Non-ATS across the 20K sample of top-million sites

FQDN	Key	Storage Keys	% Sites
www.google-analytics.com	cid	251	56.84
www.google-analytics.com	_gid	206	50.90
stats.g.doubleclick.net	cid	220	36.60
stats.g.doubleclick.net	_gid	133	34.04
www.google.com	cid	193	23.43
www.facebook.com	fbp	5	19.14
googleads.g.doubleclick.net	auid	46	13.41
analytics.google.com	cid	98	11.03
googleads.g.doubleclick.net	cookie	1	7.72
googleads.g.doubleclick.net	gpic	1	7.71
googleads.g.doubleclick.net	ga_vid	112	6.97
securepubads.g.doubleclick.net	ga_vid	54	6.69
partner.googleadservices.com	cookie	1	5.60
partner.googleadservices.com	gpic	1	5.59
bat.bing.com	vid	2	5.55
bat.bing.com	sid	2	5.54
securepubads.g.doubleclick.net	cookie	1	5.45
securepubads.g.doubleclick.net	gpic	1	5.42
www.google-analytics.com	sid	802	4.99
x.bidswitch.net	user_id	139	4.70
idsync.rlcdn.com	partner_uid	208	4.61
pixel.tapad.com	partner_device_id	174	4.50
cm.g.doubleclick.net	google_hm	77	4.28
ups.analytics.yahoo.com	uid	182	4.19
pixel.rubiconproject.com	put	185	3.96
www.facebook.com	ts	202	3.84

Table 4: Prevalence of Top-25 ATS link decorations and the number of cookies they share

on *doubleclick.net*, *google-analytics.com*, and *google.com*. For example, `cid` and `_gid` are used by all of the aforementioned Google domains, with `cid` and `_gid` on *www.google-analytics.com* shared on more than half of the sites. After Google, `fbp` on *www.facebook.com* used by Meta pixel [91] is present on nearly 20% of the sites. After Facebook, `vid` and `sid` on *bat.bing.com* used by Microsoft/Bing Ads Conversion tracking [92] are present on about 5% of the sites. All of the remaining top-50 ATS link decorations also belong to well-known advertising and tracking organizations such as BidSwitch [93], LiveRamp [94], Yahoo! [95], Magnite [96], Amazon [97], Criteo [98], OpenX [99], Oracle/BlueKai [100], The Trade Desk [101], Sovrn Holdings [102], Index Exchange [103], TripleLift [104], and LiveIntent [105]. The presence of these advertising and tracking organizations in top-50 ATS link decorations (and even beyond top-50, not shown here due to space constraints) highlights PURL's effectiveness in detecting both popular and relatively lesser known ATS link decorations.

Next, we investigate different types of information shared through these ATS link decorations.

5.2 Sharing of Browser Storage through ATS Link Decorations

Prior work has shown that browser storage, such as cookies and local storage, is widely used for tracking by ATS [7], [10], [74]. To investigate whether ATS link decorations are used to share browser storage, we look for the presence of browser storage in link decorations. Table 4 shows the number of distinct browser storage keys shared in the plaintext or encoded formats (Base64, SHA1, SHA-256, and MD5) by top-50 ATS link decorations. We find that most of the ATS link decorations detected by PURL are used to share a large number of browser storage keys (cookies and local storage). For example, `google_hm` on *cm.g.doubleclick.net* is used by Google's cookie matching service [106] to receive cookies of its advertising partners such as `CMID` (set by *casalemedia.com*), `suid` (set by *simpli.fi*), `__mguid_` (set by *mediago.io*), and `tuuid` (set by *adsrvr.com*). Other such well-known ATS link decorations that are similarly used to share browser storage keys set by various organizations include `partner_uid` (LiveRamp), `partner_device_uid` (Tapad), `uid` (Yahoo!), `put` (RubiconProject), and `ttd_puid` (The Trade Desk). We also note that some ATS link decorations are used to share only a few browser storage keys. For example, `fbp` (*www.facebook.com*) is mainly used to share ghosted [7], [10] `_fbp` cookie set by Meta pixel [91]. As another example, `vid` and `sid` (*bat.bing.com*) are mainly used to share ghosted `_uetvid` and `_uetsid` cookies set by Microsoft UET tracking tag [92]. A related interesting example is `cid` (*www.google-analytics.com*), which is mainly used to share ghosted `_ga` cookie set by Google Analytics [107], [108]. However, it is also used to share 250 other browser storage keys, potentially due to cookie name conflicts [109].

5.3 Sharing of Deterministic Information through ATS Link Decorations

Next, we look at the sharing of deterministic identifiers through ATS link decoration. Deterministic identifiers include email addresses and any commonly used user identifiers such as username, phone number, etc. provided by the user to log into a site. In contrast with identifiers stored in third-party cookies, these deterministic identifiers are not automatically sent with requests, necessitating their sharing through mechanisms such as link decorations. To study this sharing of deterministic identifiers, we crawl the 20K sample of top-million websites twice, both with and without providing deterministic identifiers such as email addresses in text input fields.[11] Similar to the aforementioned storage analysis, we analyze whether link decorations contain these deterministic identifiers in plaintext or encoded format using well-known hashing techniques (Base64, SHA1, SHA256, and MD5). We repeat these parallel crawls two additional times for a total of three

[11] To automatically fill in the text fields on web pages, we extended a crawler released by prior work [45].

FQDN	Key	Sites
p.adsymptotic.com	_expected_cookie	90
idsync.reson8.com	userid	47
api-2-0.spot.im	ayl_id	33
comcluster.cxense.com	glb	29
polo.feathr.co	ttd_id	21
rtb.mfadsrvr.com	_	20
cs.iqzone.com	puid	18
pixel-geo.prfct.co	xid	13
sync.richaudience.com	pmUserId	11
rp.liadm.com	ext__pubcid	8
cdn-p.cityspark.com	b	6
ssl.connextra.com	path | 0	6
aps.zqtk.net	url	5
trackingapi.trendemon.com	CookieId	5
cs-tam.yellowblue.io	aid	5
b6.im-apps.net	vid	4
m.trafmag.com	id	4
sync.upravel.com	uid	3
cdn.gladly.com	q	3
cms.getblue.io	appnexusid	3

Table 5: Top-20 link decorations which were only present in at least 2 of the crawls where user email address was entered in text fields

crawls with and without entering deterministic identifiers. We find 538 link decorations that are present in multiple crawls where we enter deterministic identifiers, but are absent in crawls where we do not enter deterministic identifiers. PURL labels 62 of these 538 link decorations as ATS.

Table 5 shows the top-20 such link decorations. Most notable of these ATS link decorations are `ttd_id` used by Feathr (provides marketing campaign solutions to non-profits [110]), `pmUserId` used by Rich Audience (an advertising solution which works with major universal identifiers such as by ID5 [111] and The Trade Desk unified ID [112], [113]), and `ext__pubcid` used by LiveIntent (provides "cookieless email-based solutions" [114]).

5.4 Sharing of Probabilistic Information through ATS Link Decorations

Finally, we look at the exfiltration of probabilistic information through ATS link decorations. Probabilistic information includes features such as screen resolution, fonts, etc. that can be combined to extract a browser fingerprint [10], [46]. To determine if link decoration is potentially being used to send out probabilistic information, we use FP-Inspector an ML-based tool proposed by prior work [46] to detect whether the initiator scripts of link decorations are fingerprinters. We run FP-Inspector to detect 1,528 fingerprinting scripts on the tested websites. These fingerprinting scripts initiate requests containing 1,800 unique link decorations, out of which 200 are labeled as ATS by PURL. Table 6 shows the top 20 most common ATS link decorations sent by fingerprinting scripts. While it is challenging to reverse-engineer the scripts, names (e.g., `aduid`, `uuid`) of some of these link decorations make it fairly obvious that they contain some sort of identifiers.

FQDN	Key	Sites
securepubads.g.doubleclick.net	ippd	212
kraken.rambler.ru	tid	146
sofire.baidu.com	t	127
ib.adnxs.com	path | 0	117
kraken.rambler.ru	top100_id	115
kraken.rambler.ru	lv	104
log.rutube.ru	sid	92
kraken.rambler.ru	aduid	55
trackingapi.trendemon.com	vid	41
connect.facebook.net	path | 0	41
kraken.rambler.ru	adtech_uid	39
hexagon-analytics.com	uu	37
api.segment.io	path | 1	35
trackingapi.trendemon.com	MarketingAutomationCookie	27
idr.cdnwidget.com	bxvid	24
events.bouncex.net	visitid	23
sofire.baidu.com	h	22
unseenreport.com	uuid	20
ids.cdnwidget.com	path | 0	18
hexagon-analytics.com	h	17

Table 6: Top-20 ATS link decorations used by scripts involved in fingerprinting, along with their presence count on websites.

6 Discussion

In this section, we further discuss the robustness of PURL to evasion, PURL's deployment in browsers and browser extensions, and coverage of PURL's dynamic analysis.

6.1 Robustness to Evasion

We evaluate the robustness of PURL against three different evasion techniques: manipulation of link decoration key names, splitting link decoration values, and combining of link decoration values.

First, we evaluate whether changing link decoration keys impacts PURL's accuracy. To this end, we randomly change the query parameter names as well as the position of resource paths and fragments. Our evaluation shows that there is no change in PURL's accuracy due to this randomization. This is because PURL does not directly use name features for query parameters and fragments, and thus there is no impact on the features. Moreover, changing the key names of resource paths by changing their position in the URL impacts just one feature (maximum depth of decoration) but does not end up changing the classification outcome for any link decoration.

Second, we evaluate whether splitting a link decoration into multiple link decorations impacts PURL's accuracy. If the individual character lengths of the new smaller link decorations are shorter than 8 characters, PURL's pre-processing would remove such link decorations from the classification pipeline – resulting in a successful evasion. To mitigate this issue, we can exclude this pre-processing step, but it might result in more false positives. We evaluate PURL (without this pre-processing step) against this evasion technique by splitting the link decorations longer than 8 characters into multiple smaller link decorations. Splitting link decorations results in only a 0.4% drop in accuracy, which corresponds to an increase of false positive rate by 0.17%.

Finally, we evaluate whether combining all link decorations into a single encrypted string impacts PURL's accuracy. This approach has been attempted by Facebook [115] to circumvent query parameter stripping. In Facebook's case, the obfuscated URL (e.g., *https://www.facebook.com/user/posts/pfbid0RjTS7KpBA...*) contains a single encrypted resource path that essentially combines multiple query parameters. A consequence of combining link decorations in an encrypted string is that PURL would be unable to attribute storage exfiltration features to this new link decoration. However, this change also results in higher entropy and, as discussed in Section A, increases the likelihood that the link decoration will be labeled as ATS. We evaluate PURL against this evasion technique by combining the link decorations in an SHA-256 encoded string for 156,348 requests containing both ATS and Non-ATS link decorations. Here, PURL only uses the features whose values are the same across all the combined link decorations (e.g., number of ancestors, descendants, presence of ad keyword in the ascendant). Our evaluation shows that PURL is still able to detect 83.4% of new link decorations as ATS.

We do not consider any attacks that adversarially modify the graph structure, as previous research [74] has found these attacks to be non-trivial and likely to cause collateral damage, deterring attackers from using such attacks.

6.2 PURL's implementation in privacy-focused browsers or browser extensions

While PURL's implementation is not suitable for runtime deployment (mainly due to the performance overheads of the browser instrumentation and subsequent dynamic analysis), it can be used in existing privacy-focused browsers and browser extensions as follows. Concretely, we run PURL on live webpages to detect ATS link decorations offline, and then add the detected link decorations to a filter list used in Brave, Firefox, Safari, uBlock Origin, or AdGuard for runtime URL sanitization [16], [17], [20], [82], [83]. We generated a filter list compatible with popular privacy-focused extensions such as uBlock Origin and AdBlock Plus by running PURL on our dataset [27]. Our filter list is incorporated in the `adfilt` filter list [116], which is used by AdGuard. Note that PURL's classifier can be periodically rerun to generate a new filter list in case a tracker frequently changes their link decoration keys/names. If a tracker randomly generates link decoration keys/names each time, PURL's filter list can applied based on their relative position in the URL. Note that it is challenging in practice for a tracker to change its link decorations at a fast pace because it requires changing both client-side and server-side logic across different servers and organizations.

6.3 Coverage

PURL builds a graph representation of the webpage execution. The number of interactions captured depends on the intensity and variety of user activity on a webpage (*e.g.*, scrolling activity, number of internal pages clicked). PURL may not detect certain ATS link decorations if its graph representation does not capture certain interactions between different elements in the webpage because it does not sufficiently emulate different user interactions. We attempt to mitigate this issue by randomly scrolling and clicking, but it might not be always sufficient. The coverage of PURL's dynamic analysis can be improved, if needed, using various techniques from prior research [117], [118].

7 Conclusion

In this paper, we investigated the abuse of link decoration for tracking. We found that link decoration is used by known trackers for both functional and tracking purposes, even within a single URL, necessitating a fine-grained approach to detect tracking link decorations. We proposed PURL— a machine learning approach to detect and sanitize tracking link decorations. PURL leverages a graph representation that captures interactions and the flow of information across multiple layers of the web stack.

Our evaluation showed that PURL significantly outperformed existing countermeasures in its ability to detect link decorations accurately and in minimizing website breakage. Our deployment of PURL on top-million sites showed that link decoration is abused for tracking on almost three-quarters of the websites by well-known advertising and tracking services to exfiltrate first-party cookies, email addresses, and fingerprints. We also showed that PURL is robust to common evasion attempts and is readily deployable in privacy-focused browsers and browser extensions as a filter list. While PURL is orthogonal to existing countermeasures that focus on detecting specific types of tracking, it can be deployed alongside them for a defense-in-depth strategy against new and emerging online tracking techniques. We also note that there are future research directions worth exploring for detecting tracking link decorations, namely the use of LLMs (large language models). Our exploration of existing LLMs shows that out-of-the-box solutions provide an accuracy between 79-81%; however, future work can explore better prompting, context, and fine-tuning to improve this performance.

8 Acknowledgments

This work was supported in part by the National Science Foundation under grant numbers 2103439, 2103038, and 2138139.

References

[1] WebKit, *Tracking prevention in webkit*, https://webkit.org/tracking-prevention/.

[2] MDN, *Storage access policy: Block cookies from trackers*, https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Privacy/Storage_access_policy.

[3] *Privacy Sandbox for the Web*, https://privacysandbox.com/open-web/.

[4] *Tracking prevention in webkit*, https://webkit.org/tracking-prevention/#anti-fingerprinting.

[5] *Firefox's protection against fingerprinting*, https://web.archive.org/web/20230716154817/https://support.mozilla.org/en-US/kb/firefox-protection-against-fingerprinting.

[6] *Brave privacy updates*, https://brave.com/privacy-updates/.

[7] I. Sanchez-Rola, M. Dell'Amico, D. Balzarotti, P.-A. Vervier, and L. Bilge, "Journey to the center of the cookie ecosystem: Unraveling actors' roles and relationships", in *S&P 2021, 42nd IEEE Symposium on Security & Privacy, 23-27 May 2021, San Francisco, CA, USA*, 2021.

[8] Q. Chen, P. Ilia, M. Polychronakis, and A. Kapravelos, "Cookie swap party: Abusing first-party cookies for web tracking", in *Proceedings of the Web Conference*, 2021.

[9] I. Fouad, C. Santos, A. Legout, and N. Bielova, "My cookie is a phoenix: Detection, measurement, and lawfulness of cookie respawning with browser fingerprinting", in *Privacy Enhancing Technologies Symposium*, 2022.

[10] S. Munir, S. Siby, U. Iqbal, S. Englehardt, Z. Shafiq, and C. Troncoso, "Cookiegraph: Understanding and detecting first-party tracking cookies", in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023.

[11] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis", in *Proceedings of ACM CCS 2016*, 2016.

[12] H. Dao, J. Mazel, and K. Fukuda, "Cname cloaking-based tracking on the web: Characterization, detection, and protection", *IEEE Transactions on Network and Service Management*, 2021.

[13] S. Englehardt, J. Han, and A. Narayanan, "I never signed up for this! privacy implications of email tracking", *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 1, pp. 109–126, 2018.

[14] J. Wilander, *Bounce Tracking Protection*, https://github.com/privacycg/proposals/issues/6.

[15] P. N. Bahrami, U. Iqbal, and Z. Shafiq, "Fp-radar: Longitudinal measurement and early detection of browser fingerprinting", *Proceedings on Privacy Enhancing Technologies*, 2022.

[16] *Open-source tests of web browser privacy*, https://web.archive.org/web/20230731221803/https://privacytests.org/.

[17] *List of query parameters removed by brave*, https://web.archive.org/web/20230714022136/https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/query-stripping/records.

[18] Brave-Core, *Brave query filter*, https://web.archive.org/web/20240207231358/https://raw.githubusercontent.com/brave/brave-core/master/components/query_filter/utils.cc.

[19] *Adguard url tracking filter*, https://web.archive.org/web/20230415152141/https://adguard.com/en/blog/adguard-url-tracking-filter.html.

[20] *List of query parameters removed by ublock origin*, https://web.archive.org/web/20230321033919/https://github.com/uBlockOrigin/uAssets/blob/master/filters/filters.txt.

[21] *Easylist*, https://easylist.to/easylist/easylist.txt.

[22] *Easyprivacy*, https://easylist.to/easylist/easyprivacy.txt.

[23] A. Randall, P. Snyder, A. Ukani, *et al.*, "Measuring uid smuggling in the wild", in *Proceedings of the 22nd ACM Internet Measurement Conference (IMC '22)*, Association for Computing Machinery, 2022, pp. 230–243.

[24] M. Alrizah, S. Zhu, X. Xing, and G. Wang, "Errors, misunderstandings, and attacks: Analyzing the crowdsourcing process of ad-blocking systems", in *Proceedings of the 2019 Internet Measurement Conference (IMC)*, 2019.

[25] P. Snyder, A. Vastel, and B. Livshits, "Who filters the filters: Understanding the growth, usefulness and efficiency of crowdsourced ad blocking", in *ACM SIGMETRICS*, 2020.

[26] U. Iqbal, Z. Shafiq, and Z. Qian, "The ad wars: Retrospective measurement and analysis of anti-adblock filter lists", in *IMC*, 2017.

[27] PURL Sanitizer, *Purl: Tracking link decorations*, https://github.com/purl-sanitizer/purl.

[28] Y. Takata, D. Ito, H. Kumagai, and M. Kamizono, "Risk analysis of cookie sharing by link decoration and cname cloaking", *Journal of Information Processing*, vol. 29, pp. 649–656, Jan. 2021.

[29] S. Englehardt, J. Han, and A. Narayanan, "I never signed up for this! privacy implications of email tracking", *Proceedings on Privacy Enhancing Technologies*, 2018.

[30] WebTrends, *Webtrends*, https://web.archive.org/web/19961226204618/http://webtrends.com/, Dec. 1996.

[31] *Webtrends analytics software implementation and maintenance guide*, https://web.archive.org/web/20091122130817/http://www.heureka.com:80/upload/WebTrendsAnalyticsSoftwareImplementationGuide.pdf.

[32] *Collect campaign data with custom urls*, https://web.archive.org/web/20230621153745/https://support.google.com/analytics/answer/1033863#zippy=%2Cin-this-article.

[33] *Understanding the basics of utm parameters*, https://web.archive.org/web/20230329071707/https://blog.hubspot.com/customers/understanding-basics-utm-parameters, 2018.

[34] *Learn how to create utm links and understand why you need them*, https://web.archive.org/web/20230619202741/https://mailchimp.com/resources/utm-links/.

[35] *A beginners guide to utm parameters (and how to use them)*, https://web.archive.org/web/20230529162103/https://www.monsterinsights.com/a-beginners-guide-to-utm-parameters/.

[36] P. Papadopoulos, N. Kourtellis, and E. P. Markatos, "Cookie synchronization: Everything you always wanted to know but were afraid to ask", in *Proceedings of the World Wide Web (WWW) Conference*, 2019.

[37] U. Iqbal, C. Wolfe, C. Nguyen, S. Englehardt, and Z. Shafiq, "Khaleesi: Breaker of advertising and tracking request chains", in *USENIX Security Symposium*, 2022.

[38] I. Fouad, N. Bielova, A. Legout, and N. Sarafijanovic-Djukic, "Missed by filter lists: Detecting unknown third-party trackers with invisible pixels", *Proceedings on Privacy Enhancing Technologies*, vol. 2020, pp. 499–518, Apr. 2020.

[39] Y. Dimova, G. Acar, L. Olejnik, W. Joosen, and T. V. Goethem, "The cname of the game: Large-scale analysis of dns-based tracking evasion", *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, 2021.

[40] T. R. (P. Institute) *et al.*, "An analysis of first-party cookie exfiltration due to cname redirections", in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021.

[41] M. Koop, E. Tews, and S. Katzenbeisser, "In-Depth Evaluation of Redirect Tracking and Link Usage", *Proceedings on Privacy Enhancing Technologies*, 2020.

[42] D. Bollinger, K. Kubicek, C. Cotrini, and D. Basin, "Automating cookie consent and GDPR violation detection", in *Security Symposium*, 2022.

[43] J. Schuh, *Building a more private web: A path towards making third party cookies obsolete*, https://blog.chromium.org/2020/01/building-more-private-web-path-towards.html.

[44] G. Acar, S. Englehardt, and A. Narayanan, "No boundaries: Data exfiltration by third parties embedded on web pages.", *Proc. Priv. Enhancing Technol.*, vol. 2020, no. 4, pp. 220–238, 2020.

[45] A. Senol, G. Acar, M. Humbert, and F. Z. Borgesius, "Leaky forms: A study of email and password exfiltration before form submission", in *USENIX Security Symposium*, 2022.

[46] U. Iqbal, S. Englehardt, and Z. Shafiq, "Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors", in *IEEE Symposium on Security and Privacy*, 2021.

[47] I. Fouad, C. Santos, A. Legout, and N. Bielova, "My cookie is a phoenix: Detection, measurement, and lawfulness of cookie respawning with browser fingerprinting", in *Privacy Enhancing Technologies Symposium (PETS)*, 2022.

[48] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: A survey", *ACM Transactions on the Web (TWEB)*, vol. 14, no. 2, pp. 1–33, 2020.

[49] L. Hieu, M. Athina, and S. Zubair, "Cv-inspector: Towards automating detection of adblock circumvention", in *Network and Distributed System Security Symposium*, 2021.

[50] M. Alrizah, S. Zhu, X. Xing, and G. Wang, "Errors, misunderstandings, and attacks: Analyzing the crowdsourcing process of ad-blocking systems", in *Proceedings of the Internet Measurement Conference*, 2019, pp. 230–244.

[51] *Grab bag: Query stripping, referrer policy, and reporting api*, https://brave.com/privacy-updates/5-grab-bag/, 2020.

[52] Brave-Core, *Brave filter list source code*, https://github.com/brave/brave-core/blob/master/browser/net/brave_site_hacks_network_delegate_helper.cc#L29.

[53] M. Nightly, *These weeks in firefox: Issue 117*, https://blog.nightly.mozilla.org/2022/06/02/these-weeks-in-firefox-issue-117/, 2022.

[54] M. Foundation, *Firefox 102.0 release notes*, https://www.mozilla.org/en-US/firefox/102.0/releasenotes, 2022.

[55] Mozilla, *Enhanced tracking protection in firefox for desktop*, https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop.

[56] *Firefox query stripping*, https://web.archive.org/web/20230714022136/https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/query-stripping/records, 2023.

[57] *News from wwdc23:webkit features in safari 17 beta*, https://webkit.org/blog/14205/news-from-wwdc23-webkit-features-in-safari-17-beta/.

[58] *Adguard general url tracking filter rules*, https://raw.githubusercontent.com/AdguardTeam/AdguardFilters/master/TrackParamFilter/sections/general_url.txt.

[59] *Adguard specific url tracking filter rules*, https://raw.githubusercontent.com/AdguardTeam/AdguardFilters/master/TrackParamFilter/sections/specific.txt.

[60] *Adguard allowlist url tracking filter rules*, https://raw.githubusercontent.com/AdguardTeam/AdguardFilters/master/TrackParamFilter/sections/allowlist.txt.

[61] *Implement $queryprune parameter*, https://web.archive.org/web/20230614232000/https://github.com/uBlockOrigin/uBlock-issues/issues/760.

[62] *Ublock origin assets github repository*, https://web.archive.org/web/20230802013435/https://github.com/uBlockOrigin/uAssets/tree/master/filters.

[63] P. Bekos, P. Papadopoulos, E. P. Markatos, and N. Kourtellis, "The hitchhiker's guide to facebook web tracking with invisible pixels and click ids", in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23, Austin, TX, USA: Association for Computing Machinery, 2023.

[64] Mozilla Foundation, *Firefox 102.0, see all new features, updates and fixes*, https://www.mozilla.org/en-US/firefox/102.0/releasenotes, 2022.

[65] *Enhanced tracking protection in firefox for desktop*, https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop.

[66] V. L. Pochat, T. V. Goethem, S. Tajalizadehkhoob, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation", in *Network and Distributed Systems Security (NDSS) Symposium 2019*, 2019.

[67] K. Ruth, A. Fass, J. Azose, *et al.*, "A world wide view of browsing the world wide web", in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22, Association for Computing Machinery, 2022.

[68] W. Aqeel, B. Chandrasekaran, A. Feldmann, and B. M. Maggs, "On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement", in *Proceedings of the ACM Internet Measurement Conference*, 2020.

[69] *Referrals are from googleads.g.doubleclick.net or tpc.googlesyndication.com*, https://web.archive.org/web/20221119020135/https://support.google.com/analytics/answer/1011829?hl=en.

[70] *Adobe marketing cloud*, https://web.archive.org/web/20220117161726/https://better.fyi/trackers/omtrdc.net/.

[71] *How google analytics works*, https://web.archive.org/web/20230730023525/https://support.google.com/analytics/answer/12159447?hl=en.

[72] *What is amazon cloudfront?*, https://web.archive.org/web/20230713064336/https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html.

[73] U. Iqbal, P. Snyder, S. Zhu, B. Livshits, Z. Qian, and Z. Shafiq, "Adgraph: A graph-based approach to ad and tracker blocking", in *IEEE Symposium on Security and Privacy*, IEEE, 2020.

[74] S. Siby, U. Iqbal, S. Englehardt, Z. Shafiq, and C. Troncoso, "Webgraph: Capturing advertising and tracking information flows for robust blocking", in *USENIX Security Symposium*, USENIX Association, 2022.

[75] S. Maurya, X. Liu, and T. Murata, "Graph Neural Networks for Fast Node Ranking Approximation", *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2021.

[76] M. Roy, S. Schmid, and G. Trédan, "Modeling and Measuring Graph Similarity: The Case for Centrality Distance", *ArXiv*, 2014.

[77] C. E. Shannon, "A mathematical theory of communication", *The Bell System Technical Journal*, pp. 379–423, 1948.

[78] *Prevent tracking based on link decoration via query string or fragment*, https://web.archive.org/web/20220310181355/https://github.com/brave/brave-browser/issues/4239.

[79] *Security/anti tracking policy*, https://web.archive.org/web/20230520014736/https://wiki.mozilla.org/Security/Anti_tracking_policy.

[80] *Onetrust. cookiepedia*, https://cookiepedia.co.uk.

[81] M. Hils, D. W. Woods, and R. Böhme, "Measuring the emergence of consent management on the web", in *Proceedings of the ACM Internet Measurement Conference*, 2020.

[82] *List of query parameters removed by brave*, https://web.archive.org/web/20230314220905/https://github.com/brave/brave-core/blob/master/browser/net/brave_query_filter.cc.

[83] *List of query parameters removed by adguard*, https://web.archive.org/web/20230702140315/https://raw.githubusercontent.com/AdguardTeam/FiltersRegistry/master/filters/filter_17_TrackParam/filter.txt.

[84] M. S. Uddin, G. Chi, M. A. Janabi, and T. Habib, "Leveraging random forest in micro-enterprises credit risk modelling for accuracy and interpretability", *International Journal of Finance & Economics*, vol. 27, no. 3, pp. 3713–3729, 2022.

[85] A. Mills, T. Spyridopoulos, and P. Legg, "Efficient and interpretable real-time malware detection using random-forest", in *2019 International conference on cyber situational awareness, data analytics and assessment (Cyber SA)*, IEEE, 2019, pp. 1–8.

[86] *Cookies set in a visitor's browser by hubspot*, https://web.archive.org/web/20230729122826/https://knowledge.hubspot.com/privacy-and-consent/what-cookies-does-hubspot-set-in-a-visitor-s-browser.

[87] *Bsi customer data platform*, https://web.archive.org/web/20230601052226/https://www.bsi-software.com/en/cdp.

[88] *Cookieless advertising solution*, https://web.archive.org/web/20230729220324/https://www.intentiq.com/.

[89] *List of query parameters removed by crumbcruncher*, https://web.archive.org/web/20230724081035/https://github.com/ucsdsysnet/crumbcruncher/blob/master/data/simple_results.csv.

[90] PURL Sanitizer Contributors, *Breakage analysis sites – purl sanitizer*, https://raw.githubusercontent.com/purl-sanitizer/purl/main/data/breakage-analysis-sites.md.

[91] *fbp and fbc Parameters*, https://web.archive.org/web/20220722220344/https://developers.facebook.com/docs/marketing-api/conversions-api/parameters/fbp-and-fbc/.

[92] Microsoft, *Conversion tracking – microsoft advertising*, url = https://about.ads.microsoft.com/en-us/solutions/tools/conversion-tracking,

[93] BidSwitch, *User matching – bidswitch*, https://protocol.bidswitch.com/features/user-matching.html.

[94] LiveRamp, *Implementing liveramp's cookie sync tag*, https://docs.liveramp.com/identity/en/implementing-liveramp-s-cookie-sync-tag.html.

[95] DuckDuckGo, *Yahoo.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/yahoo.com.json.

[96] DuckDuckGo, *Rubiconproject.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/rubiconproject.com.json.

[97] DuckDuckGo, *Amazon-adsystem.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/amazon-adsystem.com.json.

[98] DuckDuckGo, *Criteo.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/criteo.com.json.

[99] DuckDuckGo, *Openx.net tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/openx.net.json.

[100] DuckDuckGo, *Bluekai.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/bluekai.com.json.

[101] DuckDuckGo, *Adsrvr.org tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/adsrvr.org.json.

[102] DuckDuckGo, *Lijit.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/lijit.com.json.

[103] DuckDuckGo, *Casalemedia.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/casalemedia.com.json.

[104] DuckDuckGo, *3lift.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/3lift.com.json.

[105] DuckDuckGo, *Liadm.com tracking data – duckduckgo tracker radar*, https://github.com/duckduckgo/tracker-radar/blob/main/domains/US/liadm.com.json.

[106] Google, *Cookie guide – authorized buyers real-time bidding*, https://developers.google.com/authorized-buyers/rtb/cookie-guide.

[107] *How Google Uses Cookies*, https://web.archive.org/web/20230731174820/https://policies.google.com/technologies/cookies?hl=en-US.

[108] *Our advertising and measurement cookies*, https://web.archive.org/web/20230730054216/https://business.safety.google/adscookies/.

[109] M. Zhang and W. Meng, "Detecting and understanding javascript global identifier conflicts on the web", in *European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020.

[110] *Reach your people where they are with digital ads*, https://www.feathr.co/platform/digital-advertising.

[111] *Id5 - first party ids and identity resolution methods explained*, https://web.archive.org/web/20220408035339/https://id5.io/news/index.php/2022/03/24/first-party-ids-and-identity-resolution-methods-explained/.

[112] *Unified id adoption guidelines for ssps*, https://web.archive.org/web/20230801095601/https://www.thetradedesk.com/assets/global/Unified-ID-Adoption-Guidelines-for-SSPs-v1.8.pdf.

[113] *Our marketplace, a solution for premium publishers*, https://web.archive.org/web/20230615100615/https://richaudience.com/en/publishers/.

[114] *Discover a new advertising channel for powering profitable growth: Email.* https://web.archive.org/web/20230529064659/https://www.liveintent.com/advertiser-solutions/.

[115] *Facebook is now encrypting links to prevent url stripping*, https://web.archive.org/web/20230527091045/https://www.schneier.com/blog/archives/2022/07/facebook-is-now-encrypting-links-to-prevent-url-stripping.html.

[116] *Actually legitimate url shortener tool*, https://github.com/DandelionSprout/adfilt/blob/291223719312625767 05adc68620a5fd2da6b501/LegitimateURLShortener.txt#L1482.

[117] J. Kupoluyi, M. Chaqfeh, M. Varvello, *et al.*, "Muzeel: Assessing the impact of javascript dead code elimination on mobile web performance", in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022.

[118] X. Hu, Y. Cheng, Y. Duan, A. Henderson, and H. Yin, "Jsforce: A forced execution engine for malicious javascript detection", in *Security and Privacy in Communication Networks: 13th International Conference, SecureComm 2017, Niagara Falls, ON, Canada, October 22–25, 2017, Proceedings 13*, Springer, 2018, pp. 704–720.

[119] *Interpreting random forests*, https://web.archive.org/web/20230519192208/http://blog.datadive.net/interpreting-random-forests/.

A Feature Analysis

We rank the most important features for classifying ATS and Non-ATS link decorations by summing the feature contribu-

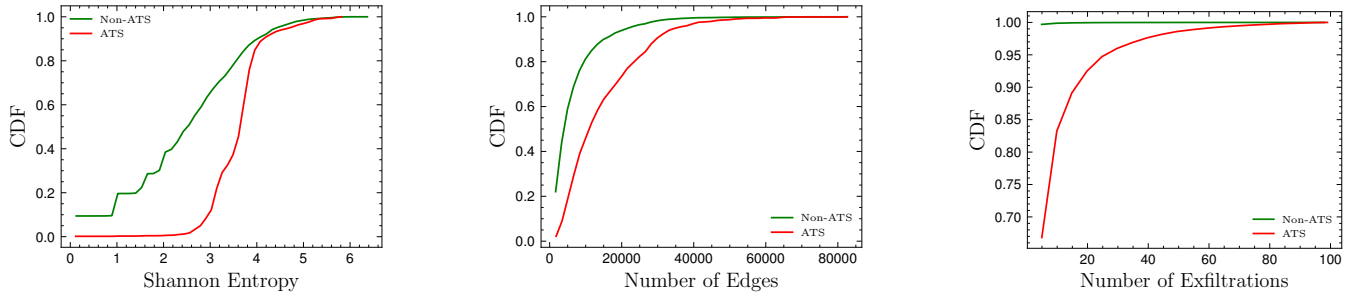

Figure 8: Distribution of the number of edges, Shannon entropy and number of exfiltrations for ATS and Non-ATS link decorations. ATS link decorations interact more with the neighboring nodes, have a higher Shannon entropy and are exfiltrated more as compared to Non-ATS link decorations.

tions during classification [119]. Table 7 reports the top-10 features ranked based on the percentage of instances where the feature was important for ATS classification. For ATS instances, Shannon Entropy, the number of nodes, edges, predecessors, and ancestors are the most important. Figure 8 plots the conditional distribution of two top-ranked features: Shannon entropy and the number of edges. The Shannon entropy for ATS decorations is higher than Non-ATS decorations – more than 70% of Non-ATS link decorations and only 8% of ATS link decorations have Shannon entropy lower than 3, respectively. The usage of higher entropy strings by ATS link decorations is expected, as they are more suitable for storing unique identifiers.

Figure 8 shows that ATS decorations are more connected as compared to Non-ATS decorations – less than 50% of ATS link decorations and more than 85% of Non-ATS link decorations have less than 10,000 edges, respectively. ATS link decorations tend to interact more with other elements of the webpage, which is expected as ATS link decorations are expected to be fetched and updated more frequently from storage and shared more frequently through network requests. Additionally, the flow of information from storage to link decoration nodes also affects the decision due to the importance of indirect features, which are calculated by exfiltration and infiltration to and from the storage nodes. ATS link decorations are more likely to exfiltrate storage values, with an ATS decoration averaging 7.4 exfiltrations (standard deviation of 20.56) while a Non-ATS decoration averaging only 0.06 (standard deviation of 1.41).

For Non-ATS instances, the lack of information flow was a strong indicator, with the top three features related to exfiltration and infiltration of storage values. The classifier also considered string entropy and advertisement-related keywords in ascendants as important features for classifying Non-ATS link decorations. Our analysis shows that information flow features, entropy, and the position of link decoration within the URL are the most critical for classifying link decorations into ATS and Non-ATS categories.

Feature	Percentage
Shannon Entropy	17.43%
Number of Nodes	16.38%
Closeness Centrality of Indirect Edges	13.22%
Ratio of Nodes over Edges	12.4%
Number of Edges	11.54%
Number of Script Predecessors	5.28%
Number of Ancestors	3.22%
Ratio of Edges over Nodes	3.06%
Number of Indirect Ancestors	2.83%
Closeness Centrality	2.03%

Table 7: Percentage of instances where a feature was the most important for ATS link decoration classification

Feature	Type
Graph size (# of nodes, # of edges, and nodes/edge ratio)	Structure
Degree (in, out, in+out, and average degree connectivity)	Structure
Centrality (closeness centrality, eccentricity)	Structure
Ascendant's attributes	Structure
Descendant of a script	Structure
Ascendant's script properties (Ad keyword, FP keyword, length of script)	Structure
Parent is an eval script	Structure
Depth of Link Decoration in URL	Content
Shannon Entropy	Content
Local storage access by parent (# of sets, # of gets)	Flow
Cookie accesses by parent (# of sets, # of gets)	Flow
Requests (sent, received) by parent	Flow
Redirects (sent, received, depth in chain) by parent	Flow
Common access to the same storage node	Flow
Cookie exfiltration	Flow
Cookie infiltrations by parent	Flow
Cookie Setter (# of exfiltration, # redirects) by parent	Flow
Graph size (# of nodes, # of edges, and nodes/edge ratio)	Flow
Degree (in, out, in+out, and average degree connectivity)	Flow
Centrality (closeness centrality, eccentricity)	Flow

Table 8: Features used by PURL. PURL calculates Graph size, Degree, and Centrality features using both normal and shared information edges. The former comes under structural features while the latter comes under flow features.