Measuring UID Smuggling in the Wild

Audrey Randall
aurandal@eng.ucsd.edu
UC San Diego

Peter Snyder
pes@brave.com
Brave Software

Alisha Ukani
aukani@ucsd.edu
UC San Diego

Alex C. Snoeren
snoeren@cs.ucsd.edu
UC San Diego

Geoffrey M. Voelker
voelker@cs.ucsd.edu
UC San Diego

Stefan Savage
savage@cs.ucsd.edu
UC San Diego

Aaron Schulman
schulman@cs.ucsd.edu
UC San Diego

ABSTRACT

This work presents a systematic study of UID smuggling, an emerging tracking technique that is designed to evade browsers' privacy protections. Browsers are increasingly attempting to prevent cross-site tracking by partitioning the storage where trackers store user identifiers (UIDs). UID smuggling allows trackers to synchronize UIDs across sites by inserting UIDs into users' navigation requests. Trackers can thus regain the ability to aggregate users' activities and behaviors across sites, in defiance of browser protections.

In this work, we introduce CrumbCruncher, a system for measuring UID smuggling in the wild by crawling the Web. CrumbCruncher provides several improvements over prior work on identifying UIDs and measuring tracking via Web crawling, including in distinguishing UIDs from session IDs, handling dynamic Web content, and synchronizing multiple crawlers. We use CrumbCruncher to measure the frequency of UID smuggling on the Web, and find that UID smuggling is present on more than eight percent of all navigations that we made. Furthermore, we perform an analysis of the entities involved in UID smuggling, and discuss their methods and possible motivations. We discuss how our findings can be used to protect users from UID smuggling, and release both our complete dataset and our measurement pipeline to aid in protection efforts.

CCS CONCEPTS

• **Networks** → **Network measurement**; • **Security and privacy** → **Privacy protections**.

ACM Reference Format:
Audrey Randall, Peter Snyder, Alisha Ukani, Alex C. Snoeren, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. 2022. Measuring UID Smuggling in the Wild. In *ACM Internet Measurement Conference (IMC '22), October 25–27, 2022, Nice, France.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3517745.3561415

1 INTRODUCTION

Over the past few years, tensions have deepened between those collecting detailed user behavior data for advertising purposes and privacy-conscious users who do not want to be monitored. While there are some efforts to find a compromise between these positions (e.g., allowing the collection of aggregated, anonymized data [5, 17]), none have yet managed to satisfy advertisers or privacy advocates [8, 37, 49]. In the absence of such a solution, privacy-focused browsers (i.e., browsers for which privacy is seen as a competitive advantage) have rolled out changes that block one of the core mechanisms used by *third-party trackers* to aggregate information about a user *across different websites*, thereby building a profile of that user.

Previously, third-party trackers could build user profiles across websites because information stored by the tracker was accessible to that tracker across all websites that include it. Trackers commonly used third-party cookies for this purpose, although any type of browser storage could be used. Trackers could use this shared storage to build shared state for each user across every website that included the tracker. However, several browsers are now employing an anti-tracking defense called "partitioned storage," which removes this sharing ability. By partitioning all browser storage by the domain of the top-level website, browsers intended to prevent trackers from linking user information across sites.

However, trackers have responded by implementing a new class of tracking technique that we call *UID smuggling*. UID smuggling allows trackers to share a user's information across websites by modifying the user's navigation requests. The tracker accomplishes this style of tracking by decorating users' navigation requests with identifying information, which will then be shared across first-party boundaries. The tracker may also choose to momentarily redirect the user to its own domain, where it can record this smuggled information as a first party itself. In each case, trackers use UID smuggling to regain the ability to link user identifiers across sites, circumventing the browser's attempt to partition such information.

This work presents the first systematic measurement of UID smuggling in the wild. We make the following contributions:

(1) We perform the **first systematic measurement** of UID smuggling in the wild.
(2) We construct a multi-stage **analysis pipeline**, nicknamed CrumbCruncher, to crawl the Web and measure how frequently UID smuggling occurs.
(3) We improve on prior techniques for **differentiating user identifiers** from other values and **synchronizing multiple crawlers**.
(4) We categorize the **behaviors** of trackers, including which categories of sites are more likely to include UID smuggling.
(5) We contribute to countermeasures against UID smuggling, both by sharing our hand-edited **dataset**, and by publishing our tool for finding new instances, CrumbCruncher.

IMC '22, October 25–27, 2022, Nice, France

ACM ISBN 978-1-4503-9259-4/22/10.
https://doi.org/10.1145/3517745.3561415

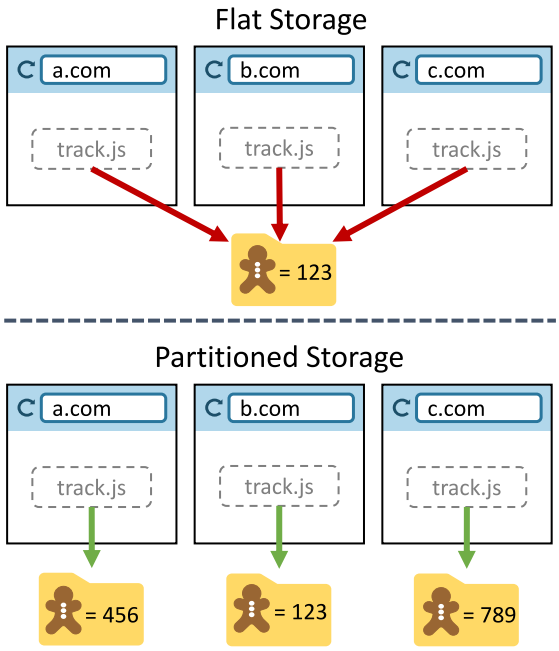

Figure 1: Flat storage versus partitioned storage.

The remainder of our work is organized as follows. Section 2 covers the background of navigational tracking and related work. Section 3 describes the design of our crawler, CrumbCruncher, and its capabilities and limitations. Section 4 discusses ethical ramifications. Section 5 presents our findings, including the most common participants in navigational tracking and a summary of their behaviors and categories. Section 6 describes the limitations of our work. Section 7 details our contribution to the countermeasures that various entities have taken against navigational tracking. Section 8 discusses related work, and Section 9 concludes.

2 BACKGROUND

Advertisers want to track user activity across sites for a variety of purposes, including performing identity resolution and supporting affiliate marketing, but such capabilities represent a significant threat to user privacy.

For over a decade, browsers allowed advertisers to perform cross-site tracking functions with third-party cookies. But because this capability presents a threat to privacy, several popular browsers have implemented *partitioned storage* to isolate third-party cookies so they cannot be used for cross-site tracking. At the time of writing, three browsers — Firefox, Safari, and Brave [19, 25, 50] — all use partitioned storage by default. Partitioned storage uses a hierarchical namespace, where the hierarchy is based on the domain of the frame that contains the cookie-storing element. Figure 1 shows the difference between flat and partitioned storage from a tracker's perspective. When flat storage is in use, the tracker can read from or write to the same storage area regardless of which website it is on, but when partitioned storage is implemented, the tracker accesses a different storage area on each website that loads the tracker. This prevents trackers from assigning the same user-identifying cookie (represented by the gingerbread man icon) to users across sites. A similar system is used for other browser storage locations, such as local storage.

To circumvent the protections that partitioned storage provides, advertisers are increasingly using *UID smuggling*. UID smuggling modifies a user's navigation requests by adding information to the navigation
may also redirect the user to one or more third-party trackers before redirecting to the intended destination.

Figure 2 shows this process in detail. In UID smuggling, the user is sent through a **navigation path**. This path begins at the **originator** website, where the user clicks a link (step ①). When the link is clicked, the page itself or a tracker on the page decorates the URL by adding the originator's user identifier (UID) as a query parameter. The link may be either a first party or a third party link, because any script on a webpage may modify any link within its frame. Third party scripts are often loaded within the top level frame of the page, so they may modify any link within the top level frame. After the user clicks the link, the navigation path passes through zero or more **redirectors**, which are invisible to the user but are permitted to store first party cookies (step ②). Each of these redirectors has the ability to store the UID from the query parameter as a cookie or local storage value under the redirector's domain. Finally, the user is sent to the **destination**, the website the link originally pointed to (step ③). The destination may also store the UID under its own domain. If there are no redirectors in the navigation path, the threat to users is simply that the UID gets passed between the originator and the destination. This may occur when the tracker that performs the UID smuggling is confident that its scripts will be present on both the originator and destination page, to decorate the link and collect the UID respectively. Thus, trackers using UID smuggling regain the ability to share UIDs across websites with different domains, in defiance of the browser's partitioned storage protections.

UID smuggling is related to, but more powerful than, two previously studied tracking techniques: bounce tracking and cookie syncing. Bounce tracking also modifies a user's navigation path by redirecting them through tracking sites that can store first-party cookies. Bounce tracking allows a tracker to record which originator and destination websites a user has visited, but not to aggregate any information about a user's behavior on those websites (the links the user clicks, purchases the user makes, etc.). This is the case because no link decoration is used to insert UIDs into the navigation path: if such link decoration is used, the technique becomes UID smuggling. A bounce tracker thus cannot link together the different UIDs it has assigned to a user across different websites. Both UID smuggling and bounce tracking are part of a class of tracking techniques known as "navigational tracking."

Cookie syncing allows multiple third parties on a *single* first-party site to share UIDs with each other. However, if partitioned storage is in place, third parties cannot share information *across* first-party websites using cookie syncing. When partitioned storage is in use, the storage available to trackers on the destination site is partitioned away from their storage on the originator. Thus, all trackers on the originator can share their UIDs with each other,

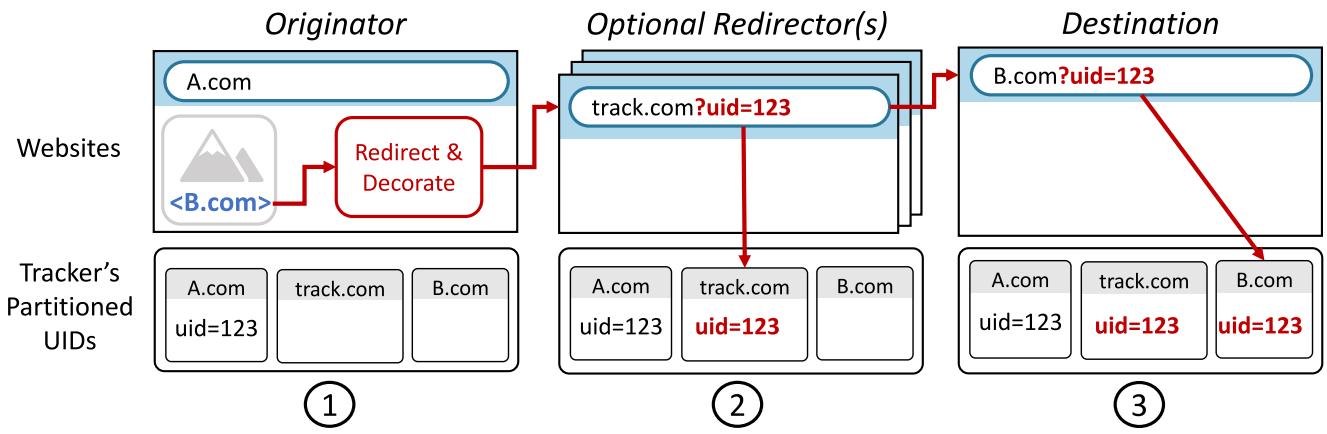

Figure 2: How UID smuggling allows trackers to circumvent partitioned storage.

and all trackers on the destination can do likewise, but trackers on the originator cannot share UIDs with trackers on the destination.

3 METHODOLOGY

In this section, we describe CrumbCruncher, a web crawling system based on Puppeteer that measures the prevalence of UID smuggling in the wild. CrumbCruncher's goal is to collect as many potential cases of UID smuggling as possible, and then distinguish the benign cases from true UID smuggling by determining which smuggled values are truly UIDs. To collect potential UID smuggling, CrumbCruncher employs multiple synchronized crawlers that simulate a set of users, with an additional, trailing crawler that simulates a user returning to each site. CrumbCruncher must then identify which potential UIDs are truly UIDs by comparing them across each crawl: values that vary across the set of different users but remain static for the repeat visitor are likely to be UIDs.

The canonical approach for identifying UIDs in prior work is to use only two crawlers to compare potential UID values across users. Unfortunately, these studies have been forced to discard a large number of potential UIDs from their analyses under two circumstances: first, when the potential UID only appeared on one crawler instead of both, and second, when the potential UID might be a session ID instead. Because we expect UID smuggling to be rare and difficult to find in the wild, we require CrumbCruncher to discard as few UIDs as possible. CrumbCruncher achieves this goal in three ways. First, it distinguishes UIDs from session IDs more accurately than prior studies, which allows it to retain UIDs that would have been discarded by previous common strategies [1, 12, 13, 26]. Second, when potential UID smuggling does not appear on all crawlers, CrumbCruncher applies programmatic and manual heuristics to identify UIDs, rather than discarding the cases entirely as prior work does [1, 12, 13, 15, 26]. Finally, CrumbCruncher introduces a novel method for synchronizing web crawlers that click iframes, which allows it to collect data from the elements that are most likely to contain UID smuggling. CrumbCruncher also uses four synchronized crawlers, rather than two, giving it multiple chances to observe each potential UID across two crawlers. Each of these improvements allows CrumbCruncher to collect or retain more data than previous systems.[1]

3.1 Crawling the Web

CrumbCruncher collects a sample of websites that contain UID smuggling by performing ten-step random walks. Each random walk begins at a "seeder domain" taken from the Tranco list of the globally most-popular 10,000 domains [27].[2] Each of CrumbCruncher's multiple crawlers follows the same walk.

At each step of a walk, CrumbCruncher records all first-party cookies, local storage values, and web requests on the originator page. Next, it chooses either a frame (`<iframe>`) or anchor (`<a>`) element to click on, in an attempt to trigger navigation. CrumbCruncher selects iframes because we expect them to contain advertisements which might use UID smuggling. CrumbCruncher also follows anchors because many webpages do not contain iframe elements. Regardless of element type, CrumbCruncher preferentially chooses elements that navigate to a URL with a different registered domain than the current page. If such an element does not exist, CrumbCruncher selects one at random. For each click that triggers a navigation, CrumbCruncher's browser extension collects all navigation web requests by implementing a handler for the `chrome.webRequest.onBeforeRequest` event. Upon arriving at the destination page, CrumbCruncher again records all first-party cookies, local storage values, and web requests for ten seconds.

CrumbCruncher repeats this navigation process, starting at each new page loaded by the click in the previous step, nine times. It then selects a new seeder domain to start the next random walk. CrumbCruncher retains browser state (including cookies and storage values) for the duration of each walk and discards it when beginning a new walk. CrumbCruncher proceeds in this depth-first manner to maximize the number of distinct pages visited, rather than maximizing the elements visited per page, to distribute the sites it performs clicks on as widely as possible. This strategy minimizes CrumbCruncher's potential impact on advertisers (see Section 4 for

[1] For more details on how prior work identified UIDs, please see Section 8.
[2] We choose 10,000 seeder domains because several prior studies also used that number [10, 15, 35].

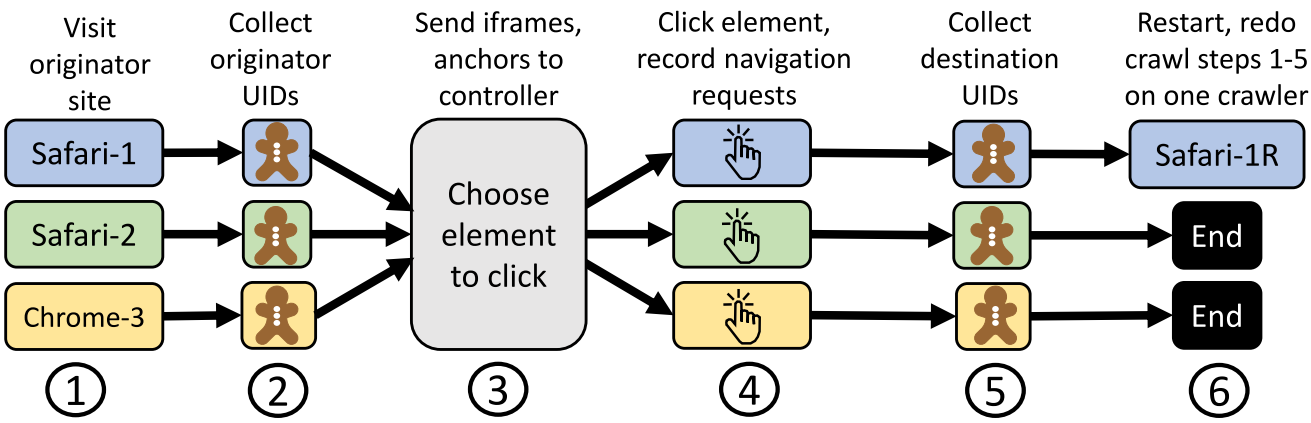

Figure 3: A single step of the ten-step random walk that CrumbCruncher performs for each seeder domain.

more details). It also allows CrumbCruncher to observe websites that range in popularity, rather than staying within the ecosystem of popular websites.

3.2 Detecting potential UID Smuggling

The goal of CrumbCruncher's is to identify cases where a UID has been smuggled — i.e., passed across sites in defiance of browser protections — which requires differentiating UIDs from non-tracking tokens. We use the term "token" to refer to any potential UID found inside the value of a name-value pair, whether that pair is a first-party cookie, a local storage object, or a query parameter. CrumbCruncher builds on prior work that identifies UIDs by comparing the tokens that are passed by two different users access a particular website [1, 12, 13, 15, 26, 46]. However, instead of using two crawlers, CrumbCruncher uses four.

Three of the four crawlers — named Safari-1, Safari-2, and Chrome-3 — each simulate a different user on a Safari or Chrome browser. These three crawlers, which run in parallel, allow CrumbCruncher to discard tokens that are the same across users and are thus unlikely to be UIDs. We explain how CrumbCruncher spoofs browsers and why we simulate both Safari and Chrome in Section 3.4 and how we impersonate different users in Section 3.5. The fourth crawler, Safari-1R, simulates the same user as Safari-1. Safari-1R checks whether the same token is observed when a webpage is accessed twice by the same user: specifically, Safari-1R repeats each crawl step immediately after Safari-1 finishes it. Safari-1R allows CrumbCruncher to discard tokens that differ when observed repeatedly by the same user, and thus are probably session IDs, not UIDs; Section 3.7 provides more details on this process.

3.3 Synchronizing multiple crawlers

One underlying assumption behind the multi-crawler methodology is that all browsers are accessing the same version of a particular webpage: the four crawlers must visit the same URL and click the same elements on each page. Figure 3 illustrates this process. However, we find that keeping the crawlers synchronized presents a significant challenge due to the dynamic nature of the web. Determining which elements are the same on different instances of the same webpage is not straightforward. Even when accessed in parallel, websites often load dynamic content: elements that appear on one crawler's page might not appear on the others'. We also find that even when elements are the same (e.g., iframes that load the same content), they might not appear in the same locations or with the same size. Additionally, CrumbCruncher clicks iframe elements, which often do not have any attribute that identifies where a user will navigate when they click the iframe. Determining which iframe elements are equivalent across different instances of a webpage is more challenging than comparing anchor elements, which almost always have an easily comparable `<href>` attribute.

To mitigate this issue, CrumbCruncher uses a central controller (a local HTTP server) to choose the element that Safari-1, Safari-2 and Chrome-3 click in parallel. Upon loading a page, each crawler sends a list of all anchor and iframe elements on that page to the central controller. These lists contain the elements' properties, location, bounding boxes, and x-paths. The controller compares the three lists to find elements that are the same across all three instances of the page. We consider elements to be the same if any of three heuristics are met:

(1) They are anchors and their `href` values are the same (not including query parameters).
(2) They have the same HTML attribute names (the values may differ) and similar bounding boxes (the y-coordinate may differ, to allow for elements that render at different heights on the page).
(3) They have the same HTML attribute names and x-path.

These heuristics are imperfect: they may incorrectly label elements as the same when they are not, or incorrectly discard elements. To mitigate these possibilities, CrumbCruncher compares the fully qualified domain name (FQDN) of the site each crawler has landed on at the end of every crawl step. If all three FQDNs are not the same, CrumbCruncher terminates the walk. We still include data from this unsynchronized step in our analyses, because this situation often occurs when CrumbCruncher has clicked on

different advertisements that each exhibit a separate instance of UID smuggling.

We evaluate the effectiveness of these heuristics and find 7.6% of all crawl steps fail because CrumbCruncher is unable to find an element that is the same across all three synchronized crawls: this type of failure occurs at step ③ in Figure 3. A further 1.8% of crawl steps fail at step ⑥ because the clicked elements were not actually the same, and led to different destination websites. The only other significant reason why a crawl step might fail is if CrumbCruncher fails to connect to the website because of a network error (`ECONNREFUSED`, `ECONNRESET`, etc.) at step ①, which occurred on 3.3% of the sites it attempted to visit. We expect the probability of any of these failures occurring to be independent of the step of the random walk CrumbCruncher was on.

3.4 Impersonating different browsers

All four crawlers use Chrome (version 95 or 92) because our chosen crawling framework, Puppeteer, is designed for that browser. However, CrumbCruncher impersonates Safari on three of our four crawlers by spoofing the `User-Agent` string.[3] We chose to test Safari and Chrome because at the time of writing, Safari implemented partitioned storage by default, while Chrome's own defenses against third-party cookies were optional (we enabled them for our study). Our hypothesis was that trackers might use UID smuggling more frequently on Safari to evade its ubiquitous partitioned storage protections. Our Chrome-3 crawler was originally intended to test this hypothesis, but we were unable to use it for this purpose: UID smuggling cases quite often appeared on only one crawler, regardless of whether that crawler was one of the three Safari crawlers or the Chrome crawler (see Section 3.7). Differentiating cases where content that performed UID smuggling was loaded dynamically from cases where UID smuggling occurred deliberately on Safari and not Chrome proved to be impossible, so we simply use Chrome-3 as another distinct user to identify UIDs.

We note that while spoofing the `User-Agent` string does change the value of `window.navigator`, which is commonly used as a proxy for identifying the browser, it is not a foolproof method of impersonating a browser. Websites may use more sophisticated methods to identify a browser, such as comparing the codecs it supports [48]. However, we do not believe this is a significant problem for our study, because relatively few websites go to such lengths: Vastel et al. crawled the Alexa top 10,000 websites and found that only 93 appeared to use sophisticated fingerprinting techniques to identify the browser that was loading them [48]. We therefore consider the risk of sites misidentifying our browser to be small, given how few websites appear to use fingerprinting to identify browsers.

3.5 Impersonating different users

The Chrome browser differentiates users by storing profiles in a folder called the "user data directory" [36]. To simulate a new user at the start of each random walk, each crawler starts with a new user data directory. This folder is modified from the default in two ways: first, third-party cookies are disabled, and second, a Chrome extension is installed that records web requests.

One potential limitation of our user simulation method is that websites may generate UIDs using fingerprinting, i.e., by examining factors like `User-Agent` string, supported fonts, hardware, and more.[4] Many of these inputs are identical across all four crawlers since they are run on one machine. If a tracker generated its UIDs using fingerprinting, assigned the same UID across multiple crawlers, and then performed UID smuggling, CrumbCruncher would erroneously discard those cases. However, we find that this rarely occurs by performing the following experiment.

We observe that CrumbCruncher will not discard potential instances of UID smuggling that only appear on a single crawler: only instances that appear on multiple crawlers and have identical UIDs will be discarded. If CrumbCruncher is erroneously discarding instances of UID smuggling, we would expect to see very few cases that both occur on multiple crawlers *and* originate on sites that perform fingerprinting.

To test this hypothesis, we separate cases of UID smuggling into two groups: the cases that originate on sites that are known to employ fingerprinting, and cases that originate on other sites. To determine which sites use fingerprinting, we use the list of fingerprinters found by Iqbal et al. [24]. Only 13% of UID smuggling in our data originates on pages hosted by one of Iqbal et al.'s fingerprinters. We then divide both groups again, into the instances that occur on a single crawler and the instances that occur on multiple crawlers. Next, we compare the proportion of single-crawler to multiple-crawler instances in the fingerprinting group to the non-fingerprinting group. In the fingerprinting group, 44% of UID smuggling cases occur on multiple crawlers, whereas in the non-fingerprinting group, 52% of cases occur on multiple crawlers. While the two-proportion Z test suggests that this difference is statistically significant — and, therefore, that CrumbCruncher likely missed some cases of UID smuggling due to fingerprinting — the difference is small. The relative difference between populations suggests CrumbCruncher may have missed on the order of 13 cases of UID smuggling on sites that employ fingerprinting.

3.6 Identifying Potential UID Smuggling

Once CrumbCruncher has finished collecting data, we search for potential UID tokens that were transferred across first-party contexts. We define "different first-party contexts" as the case when the site the token was originally found on has a different registered domain than any of the sites that eventually received the token, whether those sites are redirectors or the ultimate destination.

We extract potential UID tokens from cookies, local storage, and query parameters by recursively attempting to parse the value of each name-value pair[5] as JSON or URL-encoded values. For example, if a query parameter contains a JSON string that itself contains several URL-encoded tokens, we extract each URL-encoded token individually.

We then discard all of the tokens that were not passed across at least one first-party context as a query parameter. For example,

[3] We use the Safari `User-Agent` string `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.1.2 Safari/605.1.15`.

[4] IP address is generally too variable to be used as an input by fingerprinters [11].

[5] We do not look for tokens in the names of name-value pairs because Fouad et al. found that storing UIDs as names rather than values was a very uncommon practice [15].

if the same token was found on both the originator site and the destination, but was not passed from the originator to the destination as a query parameter, we discard it. We find that the vast majority of these particular tokens are not used in UID smuggling, but rather false positives that happen to appear on both websites, such as location or language specifiers.

However, we keep tokens that only get passed across part of a navigation path. For example, if a token appears as a query parameter in the URL of a redirector, then gets passed to the destination, we keep it even if it did not appear on the original URL of that navigation path. We note that while we cannot tell whether a site that received a UID did anything harmful with it, the fact that the site received the UID at all represents a privacy risk. Tokens are also not required to appear as cookies or local storage values: they can appear on the originator and destination as query parameters in third-party web requests.

3.7 Identifying UIDs

After collecting all potential cases of UID smuggling, we identify and discard all of the cases that transfer harmless values rather than UIDs. Examples of harmless values include timestamps, language specifiers, session IDs, and so on. While performing this analysis, we discovered that cases of potential UID smuggling fell into two categories: we labeled these categories "static" and "dynamic." Static UID smuggling occurs on elements that are always the same on every visit to the page. Consequently, cases of static UID smuggling appear on all four crawlers. Dynamic UID smuggling occurs on elements that load different content on different page visits. Cases of dynamic UID smuggling appear on fewer than all four crawlers, despite our efforts to keep the crawlers synchronized. Identifying UIDs in static UID smuggling is simpler than in dynamic UID smuggling, and we describe our procedure for the static case first.

3.7.1 Identifying UIDs in the static case. To track an individual user, a UID must be the same across all website visits by the same user and different across visits to the same website by different users. Consequently, we discard any token that is the same across our crawlers that simulate different users, since these cannot be UIDs. However, it is also necessary to discard tokens that differ across a single user, since these tokens are likely to be session IDs that are not used for user tracking. Prior work discarded session IDs by discarding all tokens whose lifetime was less than a specific time, such as 90 days [12, 13, 26] or a month [1]. CrumbCruncher improves on prior work by comparing potential session IDs across Safari-1 and Safari-1R, which simulate the same user visiting the same website twice, and discarding the tokens that differ across these crawlers. A sampling of data collected from one of our crawler machines indicates that 16% of the UIDs we identify have a lifetime of less than 90 days, and 9% have a lifetime shorter than a month. These UIDs would have been missed by prior work that uses lifetime to determine whether a token is a session ID.

3.7.2 Identifying UIDs in the dynamic case. Unfortunately, we found that the majority of potential UID smuggling instances were dynamic and thus did not occur on all four crawlers: in fact, many instances occurred on only a single crawler. For example, we encountered many cases where each crawler loaded the same originator website and clicked the same iframe element, but the iframes contained different advertisements, so each advertisement presented a different navigation path and arrived at a different destination. We classify tokens that appear on fewer than four crawlers in the following manner:

(1) If a token is present in any two crawls with different user profiles, and its value is the *same* across those crawls, we discard it.
(2) If a token's name is present in Safari-1 and Safari-1R, which have the same user profile, and its value *differs*, we discard the token.

We are left with two classes of tokens: tokens that are present in only a single crawl, and tokens that only appear in crawls with different profiles (and have different values across each crawler). For these tokens, we employ both the programmatic heuristics used by previous work and manual sorting.

We base our programmatic heuristics on those of prior studies [12, 13, 26]. We remove tokens that appear to be dates or timestamps, tokens that appear to be URLs, and tokens that are less than eight characters long. We do not impose any restrictions based on cookie expirations. However, even after we applied these filters, manual inspection of the remaining tokens revealed a high number of obvious false positives. These included natural language strings separated by delimiters (such as "Dental_internal_whitepaper_topic," "share_button"), concatenated words with no delimiter ("sweetmagnolias," "trustpilot"), semi-abbreviated words ("navimail"), acronyms ("en-US"), and more. Filtering most of these out programmatically presented a significant challenge.

We therefore concluded that programmatic heuristics would be insufficient to distinguish UIDs from other tokens, and resorted to removing obvious false positives by hand. Our final, conservative strategy is to remove tokens that are composed of any combination of natural language words, coordinates, domains, or obvious acronyms like "en-US." Table 1 shows how many of the final set of UIDs were present on various combinations of crawlers.

User Profiles	# Tokens
2 identical plus 1 or more different profiles	325
2 or more different profiles only	171
2 identical profiles only	20
1 profile only	445

Table 1: Crawler combinations where UIDs appeared.

In the end, we manually removed 577 out of 1,581 tokens because our programmatic filters failed to recognize them as non-UIDs. This number is significantly higher than we expected and underscores the value of attempting to observe UIDs across as many crawlers as possible.

3.8 Implementation

We implemented CrumbCruncher using both Puppeteer, to automate site visits and record cookies and local storage, and a custom Chrome extension, to record web requests. We use Puppeteer in "headful" mode, using the monitor emulator XVFB [53], to reduce the chance that CrumbCruncher will be identified as a bot. While

Puppeteer is capable of recording most web requests, it cannot guarantee that it can attach request handlers before any requests on a page have been sent [3, 4]. We found during initial testing that this led to a significant number of missed requests; hence, CrumbCruncher records requests using a browser extension instead. CrumbCruncher runs on twelve Amazon EC2 `t2.large` instances. Each EC2 instance has a different set of 834 seeder domains. The full crawl of 10,000 seeder domains takes approximately three days to complete.

Unique URL Paths	10,814
Unique URL Paths w/ UID Smuggling	850
Unique Domain Paths w/ UID smuggling	321
Unique Redirectors	214
Dedicated Smugglers	27
Multi-Purpose Smugglers	187
Unique Originators	265
Unique Destinations	224

Table 2: Summary of the navigation paths and their participants measured by CrumbCruncher.

4 ETHICS

CrumbCruncher's mechanical measurements do not reflect the interests or intentions of individual consumers. Because CrumbCruncher cannot be influenced to make a purchase after clicking an ad, there exists a view that our measurements represent potential economic harms against the profits of the advertising industry and its clients, and that such economic harms may represent ethical considerations.

Unfortunately, it is difficult to precisely quantify this potential economic harm for a variety of reasons. First, different ad placements can have different payment triggers, such as cost-per-mille (CPM), cost-per-click (CPC), or cost-per-action (CPA). Second, different advertisements are priced differently for each publisher. For example, CrumbCruncher predominantly engages with display ads, which are commonly placed via real-time auctions. The prices of these ads fluctuates continuously based on a variety of factors, including conversion rates. Finally, advertising platforms commonly refund expenditures on ad clicks that they deem to be "bots." To establish an *upper bound* on the economic impact of our study, we estimate that our study involved fewer than 50,000 ad clicks. If we assume that all of these ads were placed on a top-tier network (e.g., Google Display Ads, with average CPC of $0.67 and average CPM of $3.12 [45]), and that none of our clicks were identified as bots, the total cost would be somewhere between $152 (all CPM ads) and $33,000 (all CPC ads). This expenditure in turn would be spread across the range of advertising networks and advertisers found in our random walks (a number that is also hard to estimate, but likely represents an average cost of a dollar or less for each).

Based on this assessment, we argue that the actual costs borne by the advertising ecosystem due to our experiments are modest. However, even if they were not, we would still argue that measurements such as those described in this paper are ethical and should continue to reflect a norm of research practice (as they have for over a decade). The digital advertising market today stands at close to $400B [43] in annual revenue and increasingly profits based on its ability to target ads based on detailed profiles of each user. The incentives to provide ever better targeting are enormous and there is scant evidence that the advertising industry, on its own, is likely to limit such targeting for the benefit of those users who prefer to maintain greater privacy. Thus, one of the only mechanisms for monitoring the evolution in advertising targeting technology — including techniques such as UID smuggling which are designed to bypass privacy protections — is to have researchers engage directly with the advertising ecosystem and measure it. Such efforts are critical to inform consumers, regulators and those technology developers providing improved privacy protections. We believe that such benefits vastly outweigh any modest losses to advertisers.

5 RESULTS

We consider two forms of navigation paths in our evaluation. "URL paths" consist of the full URLs of the originator, any redirectors, and the destination (e.g., `a.com/x/y?UID=0` → `b.com/x/y?UID=0`). Domain paths consist only of the domains at each step of the path (e.g., `a.com` → `b.com`).

In total, we observed 10,814 unique URL paths in the data set we gathered using CrumbCruncher. We consider unique URL paths, rather than total URL paths including duplicate paths, because this metric gives a better estimate of how many websites participate in UID smuggling.

Using our method for identifying UIDs, we found UID smuggling on 8.11% of the unique URL paths taken by CrumbCruncher. It is interesting that such a non-trivial percentage of advertisers have implemented UID smuggling, especially given that Chrome — the most widely used web browser — still permits tracking with third-party cookies by default.

We speculate that the affiliate advertising market may be driving the adoption of UID smuggling. An affiliate advertising model is one where a company that wishes to publish advertisements, such as a retailer, hires "affiliates" to distribute the advertisements on their behalf. The retailer runs an "affiliate program" which creates the advertisements and gives them to affiliates to distribute. The affiliates earn a commission on every clicked ad that leads to a purchase, known as a "conversion" [28]. Affiliate programs have reportedly been failing to attribute conversions because of browsers' third party cookie blocking [18], and link decoration allows conversions to be attributed correctly.

In the rest of this section we examine the UID smuggling we discovered in detail to understand who is implementing it, how they implement it, and why they implement it.

5.1 Redirectors

We start by identifying the trackers involved as redirectors in the navigation paths that include UID smuggling. Again, a redirector is an entity that lies in the middle of a navigation path between the originator and the destination. A smuggler can be any entity along the path that sends or receives a UID, including a redirector, an originator, or a destination. We use unique domain paths instead of

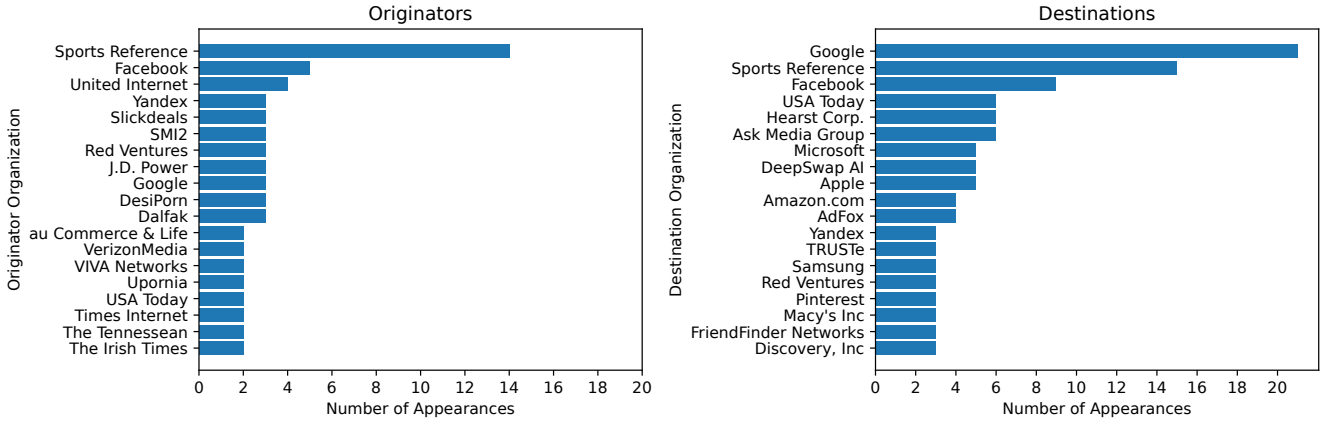

Figure 4: Most common entities involved in UID smuggling as originators or destinations.

URL paths for this analysis, because this metric better captures how widely a redirector is spread across the web, without over-counting repeated instances of UID smuggling by the same entity.

We classify redirectors into two groups: "dedicated smugglers" and "multi-purpose smugglers." We use a conservative heuristic to identify dedicated smugglers that appear to have no purpose in the navigation path besides UID smuggling.[6] Dedicated smugglers exist because they have the right to set first-party cookies, since they are visited in a first-party context, even when partitioned storage is in place. They provide an easy way for trackers to aggregate all the UIDs they collect from different sites into a first-party storage bucket. We consider a redirector a dedicated smuggler if it meets three requirements:

- The redirector appears in navigation paths whose originators have multiple different registered domains,
- The redirector appears in navigation paths that end in destinations with multiple registered domain names,
- The redirector's FQDN is never observed as an originator or destination.

If a redirector does not meet this criteria, we classify it as a multi-purpose smuggler. We separate out dedicated smugglers because we are confident that these domains have no purpose besides UID smuggling, and their sole intent is therefore likely to be enabling trackers to aggregate users' information across websites. We also predicted that dedicated smugglers might be particularly underrepresented in filter lists that block trackers, because UID smuggling is such a recent technique. Indeed, when we compared the dedicated smugglers that we found to the Disconnect list of trackers [23], 41% of them (11 out of 27) were not yet present in the list.

However, our heuristic is conservative. The less often CrumbCruncher sees a redirector, the less likely it is to observe multiple originators and destinations for that redirector, in which case the redirector would not be classified as a dedicated smuggler. Consequently, some dedicated smugglers might appear in the "multi-purpose smugglers" category.

[6]For example, site-specific redirection services (e.g., Twitter's `t.co`) are not considered dedicated smugglers using this classification.

Table 3 shows the 30 most commonly-occurring redirectors in the navigation paths we measured. From this list, 16 of the 30 most common redirectors are dedicated smugglers and 14 are multi-purpose (the multi-purpose smugglers are marked with an asterisk). Of the 16 dedicated smugglers, 14 are owned by advertisers, while the other two (`btds.zog.link` and `secure.jbs.elsevierhealth.com`) have unclear owners or purposes. The most commonly used dedicated smuggler is DoubleClick, which appears in more than 20% of all cases of UID smuggling.

The multi-purpose trackers appear to fill a variety of roles: while all of them perform UID smuggling, some have a separate purpose as well. Some redirect to sign-in pages (e.g., `signin.lexisnexis.com`), host user-facing websites (e.g., `www.facebook.com`), redirect to the English-language version of a site by appending "/en/" (e.g., `www.getfeedback.com`), or upgrade or downgrade HTTP/HTTPS connections (e.g., `kuwosm.world.tmall.com`). Some multi-purpose smugglers are owned by advertising companies, just as the dedicated smugglers are. Two redirectors, `swallowcrockerybless.com` and `d.agkn.com`, appear to be associated with Potentially Unwanted Programs (PUPs) such as adware.

5.2 Originators and Destinations

Next, we identify the organizations that acted as originators or destinations during UID smuggling. We began with the Disconnect entity list [22], which recorded an owning organization for 45 out of the 436 unique registered domains of the originators and destinations. We then identified the owners of a further 235 registered domains manually (all of the domains that appeared multiple times, plus as many of the long tail as we could). To manually attribute a hostname to an organization, we use a combination of WHOIS records, copyright ownership information published by the company, and visiting the hostname in a browser. We found that WHOIS was not a reliable method for finding this information as many websites use WHOIS privacy services, so we relied more frequently on copyright information and other publicly available information found via searching the Web. An organization is counted once per unique domain path: if multiple domains owned

Redirector	Count	% Domain Paths
adclick.g.doubleclick.net	36	11.2
googleads.g.doubleclick.net	20	6.2
advance.lexis.com*	10	3.1
d.agkn.com	9	2.8
btds.zog.link	9	2.8
ad.doubleclick.net	8	2.5
gm.demdex.net	8	2.5
www.kinopoisk.ru*	7	2.2
secure.jbs.elsevierhealth.com	6	1.9
t.myvisualiq.net	6	1.9
11173410.searchiqnet.com	6	1.9
optout.hearstmags.com*	6	1.9
signin.lexisnexis.com*	6	1.9
trc.taboola.com	5	1.6
l.instagram.com*	5	1.6
ads.adfox.ru*	5	1.6
www.facebook.com*	5	1.6
reseau.umontreal.ca*	5	1.6
l.facebook.com	4	1.2
rtb-use.mfadsrvr.com	4	1.2
www.campaignmonitor.com*	4	1.2
6102.xg4ken.com*	4	1.2
swallowcrockerybless.com*	4	1.2
montreal.imodules.com*	4	1.2
www.getfeedback.com*	4	1.2
kuwosm.world.tmall.com*	4	1.2
www.awin1.com	3	0.9
www.zenaps.com	3	0.9
pr.ybp.yahoo.com	3	0.9
go.dgdp.net	3	0.9
other redirectors	45	14.0

Table 3: The most common redirectors observed in unique domain paths. Here, "count" refers to the number of unique navigation paths the domain appeared in. *Multi-purpose smuggler.

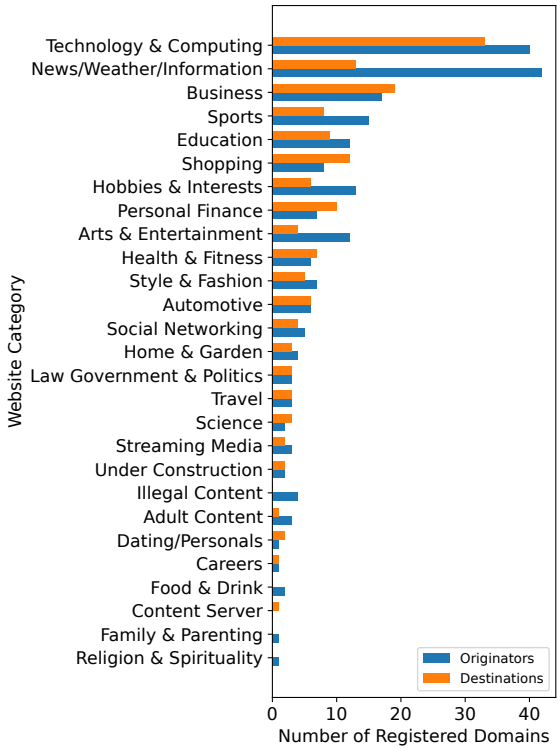

Figure 5: Categories of websites that participate in UID smuggling as originators or destinations.

by a single organization appear more than once in a domain path, the owning organization is only counted once for that path.

Figure 4 shows the originators and destinations observed most frequently in our measurements. We present the entities as organizations rather than hostnames because some organizations own multiple hostnames that appeared in our results. We note that many originators might be expected to publish affiliate advertisements, such as sports websites, news organizations, and adult websites, while many destinations might have affiliate advertising programs, such as retailers or technology companies. While we cannot guarantee that these entities participate in UID smuggling as part of affiliate advertising campaigns, many of these organizations are the types of organizations that usually participate in affiliate advertising programs as affiliates and advertisers.

Figure 4 also illustrates one particular case of UID smuggling between unexpected organizations. One of the most common cases of UID smuggling in our measurements was a navigation path that led from the originator `instagram.com`, owned by Facebook, to the Google Play Store. This path existed because the button on `instagram.com` advertising the Instagram mobile app always appended `instagram.com`'s UID cookie to the navigation request for `play.google.com`. We were surprised to see that two large advertising companies, that might be expected to be competitors, were apparently sharing UIDs with each other.

Figure 4 also contains an example of UID smuggling that was not initiated by an advertiser, but rather used to synchronize information between multiple domains owned by a single company. The most common originator in Figure 4 is Sports Reference, an organization that maintains several websites with statistics for popular American sports. This company owns several sports-themed domains whose websites link frequently to each other, such as `hockey-reference.com`, `stathead.com`, `baseball-reference.com`, and others [29]. CrumbCruncher spent several random walks in this ecosystem of websites. We hypothesize that rather than using UID smuggling for advertising, Sports Reference uses it to share information between its own affiliated sites.

5.2.1 Content categories. We further break down the originators and destinations by categorizing them by the topic of their site content. We use the categorization defined by the IAB Tech Lab Content Taxonomy [20] as provided by Webshrinker [52], whose

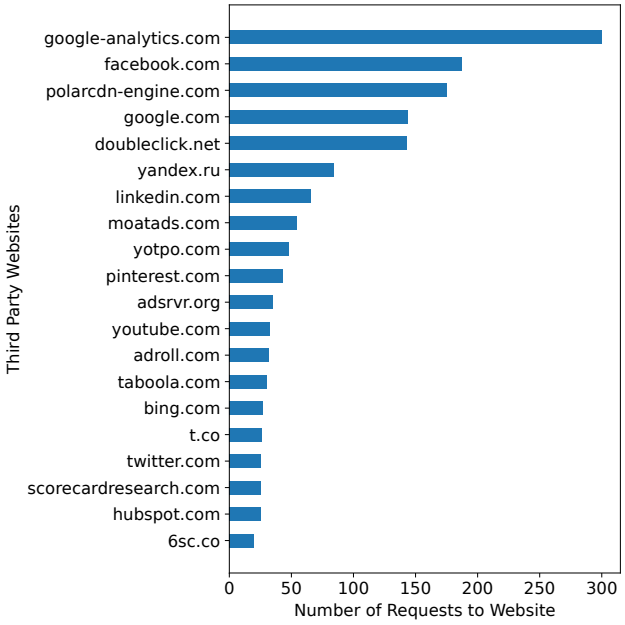

Figure 6: Most common domains of third party web requests sent from the destination site.

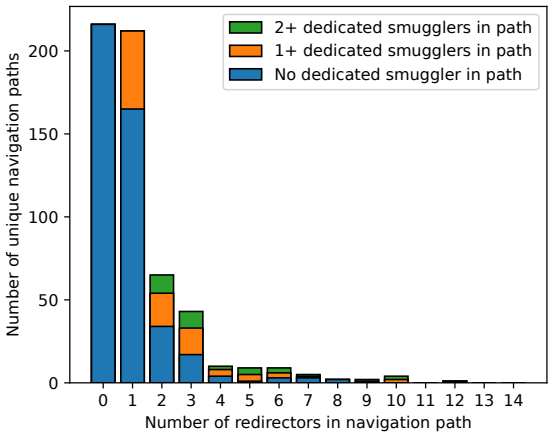

Figure 7: Distribution of types of redirectors in URL paths.

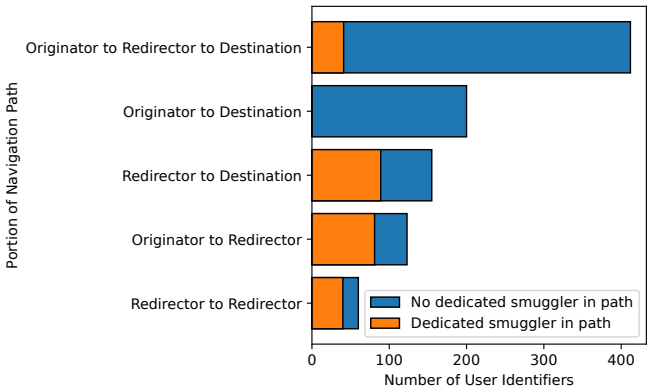

Figure 8: Counts of UIDs that traversed each portion of a URL path.

data set contains 404 domain categories [51]. Out of 339 unique registered domains, 307 had a useful category and 32 were categorized as unknown.

Figure 5 shows the most common categories of websites that participate in UID smuggling in our dataset. The counts of websites per category reflect the number of unique registered domains in that category, so that each registered domain is represented only once even if CrumbCruncher encountered it multiple times. For example, even though Facebook's domains are common originators as seen in Figure 4, they only appear twice as originators in Figure 5: once for `facebook.com` and once for `instagram.com`, both in the "Social Networking" category.

Notably, "News/Weather/Information" is the most common category for originators, and the second most common category overall. This result is consistent with previous studies that found news websites to have an above-average amount of more traditional tracking mechanisms, such as fingerprinting and tracking pixels [12, 24]. Our impression, based on manual inspection of a few of these originators, is that news websites have an above-average number of advertisements in iframes that perform UID smuggling when they are clicked.

5.2.2 Third parties. After a UID has been transferred through the entire navigation path, it may not have finished its journey: third parties on the destination site may also send the UID back to their own servers. Figure 6 shows the 20 most common registered domains of the targets of web requests sent from destination sites that included UIDs.

The third-party trackers listed in this figure include trackers that did not appear to use UID smuggling. We note that many requests to third party trackers passed the UID only because the request included the entire URL of the destination site, suggesting that the UID may have been "leaked" to these entities accidentally. This unintended consequence of UID smuggling may present a further privacy harm, in that trackers that do not participate in UID smuggling are nevertheless gaining access to UIDs that they would otherwise be unable to observe.

5.3 Navigation Paths

In this section, we examine the characteristics of navigation paths used for UID smuggling, including the features that differentiate them from benign navigation paths.

Figure 7 shows the number of redirectors in the middle of each URL path that was used for UID smuggling. The first bar, with zero redirectors, shows the cases where a UID was transferred directly from the originator to the destination without passing through any redirectors in between.

The higher the number of redirectors in a path, the greater the proportion of those paths that contain dedicated smugglers, and the greater the number of dedicated smugglers in each path. We

conclude that shorter navigation paths are more likely to have a benign purpose, whereas longer navigation paths are more likely to be used for UID smuggling.

Long navigation paths give multiple trackers the ability to share UIDs with each other. For example, one navigation path started at a coupon-collecting website (`couponfollow.com`), passed through a partner site owned by the same entity, then passed through four different trackers before arriving at the final destination (a retailer). Each of these trackers had the ability to record information about the ad the user had clicked and their apparent interest in the retailer's products.

Long navigation paths can also allow a single tracker to coordinate multiple domains that it controls. If those domains are connected to separate infrastructure (as might be the case if one advertising company acquired another and inherited the acquisition's infrastructure), the company might wish to synchronize the UIDs stored as first party cookies by redirectors. For example, the most common pair of redirectors we observed (where the first domain in the pair immediately redirects to the second domain) is `awin1.com` → `zenaps.com`. Both domains are owned by the advertiser AWIN.

UIDs do not always begin at the originator and pass through each redirector before arriving at the destination: they may appear at any step of the path and cease their journey at any number of hops further along. Each navigation path can also contain multiple UIDs. Figure 8 shows how many UIDs traverse each portion of the navigation path. We divide the UIDs that traverse each partial path into two groups: the UIDs that passed through a dedicated smuggler, and the UIDs that passed through either multi-purpose smugglers only or no redirectors at all. For example, the second bar, "Originator to Destination," shows the number of UIDs that were passed through navigation paths with no redirectors.

We observe that the majority of UIDs are transferred across the entire path from the originator, through any redirectors if they exist, to the destination. A tracker might wish to do this when it is reasonably confident that the destination will include one of its scripts, which is capable of storing the UID under the destination's domain. If a tracker is present on the originator and capable of initiating UID smuggling, but is not confident that the destination will contain one of its scripts, it might choose to transfer the UID through only part of the navigation path. These "partial transfer" cases involve a higher proportion of dedicated smugglers, which is further confirmation that the redirectors we label "dedicated" have no other purpose in the navigation path than UID smuggling. We hypothesize that trackers who only send a UID through a part of the navigation path might be less widely used, since they are apparently not confident that the destination will contain one of their scripts.

6 LIMITATIONS

CrumbCruncher has several limitations. First, we only look for UIDs that are transferred in the query parameters of URLs, and not by other methods. For example, trackers reportedly sometimes decorate the link in the document.referrer header with the UID, instead of the link to the destination page [54]. Our initial reasoning was that there are a wide variety of ways to transfer UIDs, so we could simply check once a crawl was complete for UIDs that had mysteriously appeared on different websites without being passed through a URL. In practice, this turned out to be difficult: dynamic instances of UID smuggling had to be detected using heuristics, which gave large numbers of false positives when used without the additional information provided by multiple crawlers. It turned out that when the same value appeared on two different websites, the most common reason was that the value was not a UID and had simply happened to be generated on both sites. To reduce our false positive rate and therefore the number of identifiers we had to remove by hand, we chose to consider only values that we had observed get transferred across at least two first party contexts.

Second, if a website uses browser fingerprinting to generate UIDs, our methodology may not fool the site into believing that our crawlers represent different users. As detailed in Section 3.5, the effect of browser fingerprinting on our results is very small.

Third, if a tracker uses fingerprinting to identify the browser loading the site, as opposed to the user, it may be able to tell that CrumbCruncher uses Chrome, not Safari. If a tracker only uses UID smuggling on Safari, it may choose not to perform it and CrumbCruncher may miss cases. We expect this to be very uncommon (see Section 3.4).

Fourth, our proposed solution to UID smuggling is to strip out the query parameters that contain UIDs. This may cause pages to break, especially in cases where the UID in the URL is used for a benign purpose, such as in a login page. Some login pages send UIDs to the server to determine if a user is already signed in. To test this limitation, we selected ten login pages from our dataset that CrumbCruncher had classified as performing UID smuggling. We manually removed the query parameter that contained the UID from the URL, reloaded the page, and evaluated whether the page changed or broke. We found that seven of the ten sites showed no change. One showed minor visual changes: the `<body>` element of the page moved down by 20 pixels. The final two pages showed more significant changes: one failed to auto-fill a field in a form and the other took the user to a homepage rather than to a specific subpage. These breakages are a limitation of our proposed mitigation.

Another minor limitation is that the Tranco list of websites includes non-user-facing websites [38]; however, we note that we only failed to connect to websites on the Tranco list in 3.3% of cases. Additionally, our manual heuristic for identifying UIDs may miss UIDs that are generated by concatenating natural language words; we expect this case to be so rare as to be almost nonexistent.

7 COUNTERMEASURES

7.1 Existing Mitigations

Defending against UID smuggling is not straightforward. Given the difficulty of designing defenses that do not degrade user experience, most defenders (whether browsers or browser extensions) have so far opted for either heuristic-based or blocklist-based approaches.

Safari. Safari uses heuristics: the browser will delete cookies and website data set by a redirector unless the user also interacts with the redirector as a first-party website [21]. Safari labels an originator as performing UID smuggling if 1) it automatically redirects the user to another site, and 2) it did not receive a user activation [41, 55].

Safari also classifies a site as a UID smuggler if it participates in a navigation path that contains another known UID smuggler.

Firefox. In contrast, Firefox defends against UID smugglers using the Disconnect Tracker Protection blocklist [23, 30]. Firefox clears all storage from sites on the Disconnect tracking list after 24 hours, unless the user has loaded the site as a first party in the previous 45 days [41]. Unfortunately, we found that many UID smugglers are not yet present on the Disconnect list.

Brave. The Brave browser has multiple approaches for preventing UID smuggling. First, if the browser is navigating to a link with a query parameter for another destination URL, Brave will simply redirect to the URL in the query parameter [40]. If the browser cannot detect the final destination of the navigation, it allows the navigation to proceed, but inserts an interstitial that warns users they will be tracked if they continue. Brave also maintains a list of UID smuggling URLs created from crowd-sourced and open-source information, as well as a blocklist of query parameter names that are commonly used for UID smuggling [42, 44]. Finally, Brave clears the storage areas associated with any sites it classifies as UID smugglers as soon as the user closes the tab that loaded them.

Chrome. While Chrome is in the process of deprecating third-party cookies [39], it does not appear to implement any features to defend against UID smuggling yet.

Extensions. Some browser extensions have begun to implement protections against UID smuggling as well. For example, Privacy Badger [16] — a browser extension by the Electronic Frontier Foundation that blocks cross-site tracking — identifies when a tracker inserts a redirector into a navigation path, and extracts the destination link from the query parameter in the redirector's URL [7]. Another extension, uBlock Origin, implements an interstitial-based approach similar to Brave's [31]. Many browser extensions, such as Adblock, Adblock Plus, and uBlock Origin, use the EasyList and EasyPrivacy filter lists [14]. We tested the URLs that CrumbCruncher found to participate in UID smuggling against the EasyList and EasyPrivacy lists, and unfortunately only 6% of the unique URLs we found would have been blocked. This result is likely because UID smuggling is such a new technique that filter lists have not yet caught up and begun blocking the URLs that participate. Additionally, EasyList and EasyPrivacy do not yet implement filters for specific query parameters. Stripping query parameters rather than blocking entire URLs is likely to result in fewer broken pages and therefore less inconvenience to users.

7.2 Proposed Mitigations

CrumbCruncher's data can help augment the blocklists used by privacy tools and browsers to defend against UID smuggling. We provide two contributions: first, we publish our list of token names and trackers. This list contains the query parameter names that were used to transfer UIDs across websites, as well as the list of entities that participate in UID smuggling as redirectors. Our second contribution is the code for CrumbCruncher itself, which can be run as an almost entirely automated pipeline to continuously update blocklists of navigational trackers. A major challenge of blocklist-based defenses lies in keeping those blocklists up to date: CrumbCruncher can help perform that task with much less human intervention than systems that rely on user reports of UID smuggling. The code and list of token names and trackers is available at `https://github.com/ucsdsysnet/crumbcruncher`. We also observe that while CrumbCruncher requires far less human effort than a manually created blocklist would, it still requires some manual intervention. We suggest that an approach based on machine learning for distinguishing UIDs would be a good avenue of future work, and would allow CrumbCruncher to perform its tasks in an entirely automated manner.

8 RELATED WORK

The work that is most closely related to our own is Koop et al.'s study of bounce tracking [26]. Bounce tracking is similar to UID smuggling in that users' navigation paths are modified to insert redirectors that can store values as first parties, but differs in that no UIDs are transferred across contexts. Koop et al. study bounce tracking only, and do not measure whether UIDs are transferred across contexts. CrumbCruncher also clicks both iframes and anchors, whereas Koop et al.'s crawler clicks only anchors. As a result, CrumbCruncher can detect UID smuggling used by advertisements in iframes.

To verify that CrumbCruncher crawled a reasonable sample of the Web and successfully detected modified navigation paths, we measured the instances of bounce tracking that CrumbCruncher observed while it searched for UID smuggling, and compared our findings to the instances found by Koop et al. We found that bounce tracking that did not also involve UID smuggling was present on 2.7% of the navigation paths we studied (UID smuggling was present on 8.1%). Because Koop et al. did not measure whether UIDs were transferred across contexts, their study labeled all UID smuggling that involved one or more redirectors as bounce tracking. Koop et al. found that "11.6% of the websites in the Alexa top 50,000 had at least one link leading to one of the top 100 redirectors" [26]. This finding seems consistent with our measurement that either UID smuggling or bounce tracking is present on a total of 10.8% (8.1% UID smuggling and 2.7% bounce tracking) of the unique navigation paths we followed.

8.1 Prior work on differentiating UIDs

Multiple groups have attempted to differentiate between identifiers that are capable of tracking users (UIDs) and identifiers that are not. To be a UID, a value must differ across different users, remain the same for the same user (i.e., it must not be a session ID), and contain sufficient entropy. Techniques for making these three determinations vary.

Prior work, which focused on cookies that might be UIDs, determined whether a cookie varied across users by directly or indirectly simulating different users across different crawls. Some work used two crawlers that visited the same sites simultaneously [12, 13, 15], while others simulated multiple users sequentially using a single crawler [26] or multiple crawlers [46]. Simulating multiple users sequentially enables a crawler to simulate more different users, because keeping multiple crawlers synchronized becomes more difficult as the number of crawlers increases, and a single crawler can evade this problem entirely. The disadvantage of sequential

user simulation in prior work is that the crawlers did not guarantee that they visited each website more than once and thus observed each cookie more than once. Consequently, some of the cookies measured by the single sequential crawlers could not be compared across multiple users. In contrast, CrumbCruncher makes a concerted effort to visit every website in each crawl with four crawlers that represent three different users, which increases the chance that we can compare cookies and local storage values across users.

Determining whether a token is a UID also requires discarding session IDs. Most past studies labeled cookies as session IDs if their lifetime was less than a specific time, such as 90 days [12, 13, 26] or a month [1]. These works also required that the token not vary during the crawl. In contrast, Fouad et al. did not put a lifetime limit on cookies, arguing that trackers can easily link short-lived cookies on their servers [15]. We improve on prior work for discarding session IDs by immediately repeating every crawl step using a crawler that mimics one previous user. We only assume a token is a session ID if it differs across these two crawls. As detailed in Section 3.7.1, This technique allowed us to include the 16% of UID smuggling instances that we would have discarded if CrumbCruncher had used a 90-day minimum lifetime.

A further difference between CrumbCruncher and prior work is in how we determine if tokens are "the same" across users. Some previous work used the Ratcliff/Obershelp algorithm [6] to compare cookie values and allowed those values to differ by 33% [1, 12, 26], 45% [13], or by an unspecified amount [46], while still treating the cookies as "the same." We chose to discard tokens as non-UIDs only when they are entirely identical across different users, because we wished to be unambiguous about why we had discarded a particular potential UID. Some previous work also required cookie lengths to remain the same across crawls [1, 13, 46] or to only differ by 25% [26], as well as requiring cookie lengths to be at least eight characters. We require token lengths to be greater than or equal to eight characters, but we do not place any restrictions on the similarity of token lengths across users.

8.2 Related work on cookie syncing

A related technique to UID smuggling is cookie syncing, which has been investigated by multiple groups [1, 12, 33, 34, 46, 47]. Cookie syncing is not the same as UID smuggling, because it does not allow third parties to share a UID across top level sites when partitioned storage is in use. Instead, cookie syncing allows third parties on the same site to share a UID with each other.

8.3 Other related work

Trackers may circumvent partitioned storage protections using techniques that do not rely on UID smuggling, such as CNAME cloaking [9, 10] or browser fingerprinting [11, 24, 32].

CNAME cloaking is the procedure of mapping a website subdomain to a third party domain using a DNS CNAME record. This technique allows trackers to share their first party cookies, because the browser is tricked into attaching cookies from the original website's subdomain rather than the third party domain the subdomain redirects to [10]. Trackers can access session cookies, even those belonging to financial institutions, using this technique [2, 35].

Browser fingerprinting is another technique used by trackers to circumvent partitioned storage and track users across websites. Browser fingerprinting allows a tracker to use features of a user's browser such as window size, installed fonts, supported codecs, and more to create a unique "fingerprint" of that user that can function as (or generate) a UID [11]. A 2013 study crawled 20 pages for each of the Alexa top 10,000 sites and found that 40 performed browser fingerprinting [32]. A more recent study improved detection of fingerprinting code by using machine learning [24]. They then measured the Alexa top 100,000 sites and found that 10 percent of them perform fingerprinting. They find fingerprinting is more common with popular sites, as almost 25% of the Alexa top 10,000 sites perform fingerprinting.

9 CONCLUSION

In this work, we present the first systematic study of UID smuggling, a technique that allows trackers to evade browsers' protections against cross-website tracking. We find that UID smuggling is present across 8.1% of the navigations paths we observed. We publish a list of the entities that participate in UID smuggling, and classify these entities according to their behavior and purposes. Our findings can be used by browsers to improve protections against UID smuggling.

Understanding the scope of UID smuggling, and the techniques by which it is conducted, is important to continue improving privacy on the Web. Browsers are increasingly (though not yet universally) trying to protect their users from being tracked. Understanding how trackers are circumventing new browser privacy protections is important, to make sure privacy improvements are not lost as quickly as they are gained.

ACKNOWLEDGMENTS

The authors would like to thank Umar Iqbal for shepherding this work and the anonymous reviewers for their excellent advice and feedback. Funding for this work was provided in part by National Science Foundation grant CNS-1705050, the Irwin Mark and Joan Klein Jacobs Chair in Information and Computer Science, and operational support from the UCSD Center for Networked Systems.

REFERENCES

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. 2014. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 674–689.
[2] Assel Aliyeva and Manuel Egele. 2021. Oversharing Is Not Caring: How CNAME Cloaking Can Expose Your Session Cookies. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS)*. 123–134.
[3] aslushnikov. 2018. Intercept target creation: Issue #3667. https://github.com/puppeteer/puppeteer/issues/3667
[4] berstend. 2018. Target creation event listeners are sometimes not executed early enough: Issue #2669. https://github.com/puppeteer/puppeteer/issues/2669
[5] Chetna Bindra. 2021. Building a privacy-first future for web advertising. https://blog.google/products/ads-commerce/2021-01-privacy-sandbox/
[6] Paul E. Black. 2021. Ratcliff/Obershelp pattern recognition. https://www.nist.gov/dads/HTML/ratcliffObershelp.html
[7] Bennett Cyphers. 2018. Privacy Badger Rolls Out New Ways to Fight Facebook Tracking. https://www.eff.org/deeplinks/2018/05/privacy-badger-rolls-out-new-ways-fight-facebook-tracking
[8] Bennett Cyphers. 2021. Google's FLoC Is a Terrible Idea. https://www.eff.org/deeplinks/2021/03/googles-floc-terrible-idea

[9] Ha Dao, Johan Mazel, and Kensuke Fukuda. 2020. Characterizing CNAME cloaking-based tracking on the web. In *Proceedings of the IFIP/IEEE Traffic Measurement Analysis Conference (TMA)*.

[10] Yana Dimova, Gunes Acar, Lukasz Olejnik, Wouter Joosen, and Tom Van Goethem. 2021. The CNAME of the Game: Large-scale Analysis of DNS-based Tracking Evasion. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*. 394–412.

[11] Peter Eckersley. 2010. How Unique is Your Web Browser?. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*. 1–18.

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

[13] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten. 2015. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In *Proceedings of the International Conference on World Wide Web (WWW)*. 289–299.

[14] Fanboy, MonztA, Famlam, and Khrin. 2022. EasyList - Overview. https://easylist.to/

[15] Imane Fouad, Nataliia Bielova, Arnaud Legout, and Natasa Sarafijanovic-Djukic. 2020. Missed by Filter Lists: Detecting Unknown Third-Party Trackers with Invisible Pixels. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*. 499–518.

[16] Electronic Frontier Foundation. 2022. Privacy Badger. https://privacybadger.org/

[17] Vinay Goel. 2022. Get to know the new Topics API for Privacy Sandbox. https://blog.google/products/chrome/get-know-new-topics-api-privacy-sandbox/

[18] Peter Hamilton. 2012. Server-to-Server Tracking Basics (Web-Based Affiliate Marketing). https://www.tune.com/blog/server-side-tracking-basics/

[19] Tim Huang, Johann Hofmann, and Arthur Edelstein. 2022. Firefox 86 Introduces Total Cookie Protection. https://blog.mozilla.org/security/2021/02/23/total-cookie-protection

[20] IAB. 2022. IAB Tech Lab Content Taxonomy. https://www.iab.com/guidelines/iab-tech-lab-content-taxonomy/

[21] Apple Inc. 2022. Prevent cross-site tracking in Safari on Mac. https://support.apple.com/guide/safari/prevent-cross-site-tracking-sfri40732/mac

[22] Disconnect Inc. 2022. Entity List. https://github.com/mozilla-services/shavar-prod-lists/blob/master/disconnect-entitylist.json

[23] Disconnect Inc. 2022. Tracker Protection Lists. https://github.com/disconnectme/disconnect-tracking-protection

[24] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. 2021. Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. 1143–1161.

[25] Brian Johnson, Ivan Efremov, and Peter Snyder. 2021. Ephemeral Third-party Site Storage. https://brave.com/privacy-updates/7-ephemeral-storage/

[26] Martin Koop, Erik Tews, and Stefan Katzenbeisser. 2020. In-Depth Evaluation of Redirect Tracking and Link Usage.. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*. 394–413.

[27] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2021. A research-oriented top sites ranking hardened against manipulation - Tranco. https://tranco-list.eu/

[28] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Mark Felegyhazi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. 2011. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. 431–446.

[29] Sports Reference LLC. 2022. Sports Reference | Sports Stats, fast, easy, and up-to-date. https://www.sports-reference.com/

[30] Mozilla. 2022. Enhanced Tracking Protection in Firefox for desktop. https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop

[31] Jared Newman. 2021. The incredibly sneaky way websites sidestep privacy tools to spy on you. https://www.fastcompany.com/90663878/bounce-tracking-privacy-browsers-brave-firefox-safari-edge

[32] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2013. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. 541–555.

[33] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. 2019. Cookie Synchronization: Everything You Always Wanted to Know But Were Afraid to Ask. In *Proceedings of the World Wide Web Conference (WWW)*. 1432–1442.

[34] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. 2018. Exclusive: How the (synced) Cookie Monster breached my encrypted VPN session. In *Proceedings of the European Workshop on Systems Security (EuroSec)*. 1–6.

[35] Tongwei Ren, Alexander Wittman, Lorenzo De Carli, and Drew Davidson. 2021. An Analysis of First-Party Cookie Exfiltration due to CNAME Redirections. In *Proceedings of the Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*.

[36] Chromium Git repository. 2022. User Data Directory. https://chromium.googlesource.com/chromium/src.git/+/HEAD/docs/user_data_dir.md

[37] Sam Schechner, Patience Haggin, and Tripp Mickle. 2022. Google Overhauls Cookie Replacement Plan After Privacy Critiques - WSJ. https://www.wsj.com/articles/google-overhauls-cookie-replacement-plan-after-privacy-critiques-11643115603

[38] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D Strowes, and Narseo Vallina-Rodriguez. 2018. A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists. In *Proceedings of the ACM Internet Measurement Conference (IMC)*. 478–493.

[39] Justin Schuh. 2020. 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

[40] Peter Snyder. 2021. Debouncing. https://brave.com/privacy-updates/11-debouncing/

[41] Peter Snyder and Jeffrey Yasskin. 2022. Navigational-Tracking Mitigations. https://privacycg.github.io/nav-tracking-mitigations/

[42] Brave Software. 2022. adblock-lists/brave-lists/debounce.json. https://github.com/brave/adblock-lists/blob/1453e599881854f970ab9164a104104ea9ec139f/brave-lists/debounce.json

[43] Statista. 2022. Digital advertising spending worldwide from 2021 to 2026. https://www.statista.com/statistics/237974/online-advertising-spending-worldwide/

[44] Brave Privacy Team. 2022. "Unlinkable Bouncing" for More Protection Against Bounce Tracking. https://brave.com/privacy-updates/16-unlinkable-bouncing/

[45] Top Draw Team. 2021. Online Advertising Costs In 2021 | Top Draw. https://www.topdraw.com/insights/is-online-advertising-expensive/

[46] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2018. The Unwanted Sharing Economy: An Analysis of Cookie Syncing and User Transparency under GDPR. *arXiv preprint arXiv:1811.08660*.

[47] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2020. Measuring the Impact of the GDPR on Data Sharing in Ad Networks. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS)*. 222–235.

[48] Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Xavier Blanc. 2020. FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. In *Proceedings of the Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*. San Diego, CA.

[49] Jane Wakefield. 2022. Google slammed over ad-cookie replacement flip-flop. *BBC News* (26 Jan. 2022). https://www.bbc.com/news/technology-60138876

[50] WebKit. 2019. Tracking Prevention Policy. https://webkit.org/tracking-prevention-policy/

[51] Webshrinker. 2022. IAB Categories. https://docs.webshrinker.com/v3/iab-website-categories.html#iab-categories

[52] Webshrinker. 2022. Webshrinker Website. https://www.webshrinker.com/

[53] David P. Wiggins. 2022. Xvfb—virtual framebuffer X server for X Version 11. https://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml

[54] John Wilander. 2019. Intelligent Tracking Prevention 2.3. https://webkit.org/blog/9521/intelligent-tracking-prevention-2-3/

[55] John Wilander. 2020. Bounce Tracking Protection · Issue #6 · privacycg/proposals. https://github.com/privacycg/proposals/issues/6