Review of Cookie Synchronization Detection Methods

Journal:	*ACM Journal on Responsible Computing*
Manuscript ID	JRC-2022-0007
Manuscript Type:	Literature Synthesis
Computing Classification Systems:	Security and privacy~Network security~Web protocol security, Networks~Network properties~Network privacy and anonymity, Social and professional topics~Computing / technology policy~Surveillance

Review of Cookie Synchronization Detection Methods

The research community has deemed cookie synchronization detection an inherently challenging task [12, 20, 34]. Studies aiming to identify cookie synchronizations often share high-level design choices, but deviate amongst low-level implementations. For example, the majority of studies label a cookie synchronization iff a user identifier is shared with a third party; however, there is a lack of consistency among implementations, such as party relations or identifier value definitions, or whether such definitions are even included. This review intends to provide a record of established methods and promote standardization of methods choice in future work.

CCS Concepts: • **Security and privacy** → **Web protocol security**; • **Networks** → *Network privacy and anonymity*; • **Social and professional topics** → Surveillance.

Additional Key Words and Phrases: cookie synchronization, cookie matching, tracking, cookies, methods review

ACM Reference Format:
. NA. Review of Cookie Synchronization Detection Methods. 1, 1 (September NA), 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The sharing of user browsing information is necessary for the Internet advertising and tracking industries to serve targeted ads [14, 23, 28, 29], perform cross-device tracking [18], and sell user information [28–30]. Browser cookies are a standard container for user browsing data, and the sharing of first party cookies with third parties is restricted by the Same-Origin policy [6] to protect user privacy [24, 30, 32]. Cookie synchronization is used to bypass the Same-Origin policy and share first party cookies with third parties to support the advertising and tracking ecosystem [30–32]. Cookie synchronization is defined by a variety of terms in the research community, such as cookie matching, cookie linking, cookie leaking, and ID syncing.

2 BACKGROUND

2.1 Browser Cookies and User Identifiers

Cookies are `key=value` pairs set on a user's browser to bring state to the HTTP protocol and provide session management, user personalization, and tracking functionality.

Browser cookies can be set by the `Set-Cookie` header of HTTP responses [10, 21, 24] or the `document.cookie` operation of JavaScript embedded in a visited website [7].

Cookie synchronization involves the sharing of cookie values that can uniquely identify a user (i.e. the cookie value is unique to one user). This review defines such cookie values as *identifiers*. Methods used to define and label identifiers are discussed in *Section 6.2*.

Author's address:

Manuscript submitted to ACM

2.2 Party Relations

First party cookies are set by a user requested domain, and third party cookies are set by an entity (i.e. domain or parent organization) other than the domain requested.

2.3 Cookie Synchronization

Cookie synchronization is defined as the sharing of a first or third party identifier with another third party, which can be initiated by an embedded third party resource, third party redirect, or the first party itself [22, 30, 31].

2.4 How is Cookie Synchronization Performed?

Assume a user is browsing `website1.com` and `website2.com`, and there exists tracking entities `tracker1.com` and `tracker2.com` who both set identifiers on the user's browser, `ABC` and `123`, respectively. The user later visits `website3.com`, which has an embedded resource from `tracker1.com` that initiates a GET request to `tracker1.com`. `tracker1.com` responds with a 3XX redirect instructing the user's browser to issue another request to `tracker2.com`, with the identifier for `tracker1.com` (`ABC`), placed in the parameters of the requested URL[1]. `tracker2.com` is now able to link its identifier (`123`) with `tracker1.com`'s identifier (`ABC`) [30–32].

2.5 User Privacy Erosion

Cookie synchronization allows a third party to reconstruct portions of a user's browsing history by receiving the visited first party site in the `Referer` field of a GET request header [8, 31]. Websites visited over TLS are not exempt from this history leakage, as plaintext HTTP requests to third parties share URLs requested using HTTPS [31–33]. As a tracker learns more third party identifiers for a single user, it can reconstruct a larger portion of her browsing history [31].

Cookie respawning methods such as `evercookie` [26] can enable third parties to re-identify users after clearing browser cookies. A respawned identifier can be re-synced with a tracker, effectively eliminating a user's ability to delete browser cookies. This enables third parties to track users and join browsing histories across browser refreshes [12, 31].

Server-to-server user data merges are facilitated by cookie synchronization. Separate tracker data-sets of known user information can be combined by linking respective identifiers for each tracker [12, 31].

2.6 Cookie Synchronization and the Advertising Industry

Advertising companies are motivated to collect as much user information as possible in order to serve the most targeted ads; Bashir et. al. [14] report Demand-Side Platforms (DSPs) place higher bids to serve users whom they have more information about. Cookie synchronization enables this information acquisition by sharing user browsing data and linking tracker databases, which enables ad targeting based on web history [31]. Papadopoulos et. al. [31] report ad related domains are the most prevalent entities involved in cookie synchronization, participating in 75% of all synchronizations and acquiring as much as 90% of all identifiers synced.

3 RELATED WORK

As early as 2014, Olejnik et. al. [28] showed how advertisers use cookie synchronization in real-time bidding (RTB) to reconstruct and share browsing history. HTTP traffic and browser cookies were collected from 100 real users browsing

[1] Additional locations to share identifiers are discussed in *Section 6.2.*

more than 70 sites each. After 70 site visits, a user experienced on average 100 cookie synchronizations with 30 domains involved.

Acar et. al. [12] investigated the effect Firefox's privacy settings {`Allow Third Party Cookies`, `Allow All Cookies but Do Not Track`, `Block Third-Party Cookies`} have on the number of cookie synchronization a user encounters. Multiple crawls of the Alexa top 3,000 sites were performed with browser cookies logged. When third party cookies were allowed, 596 identifiers were synced over 407 unique first parties, with 323 third parties involved. Selecting `Do Not Track` only decreased the number of domains involved in cookie synchronization by 2.9% and identifiers shared by 2.6%. When third party cookies were blocked, this decreased the number of identifiers synced to 353 over 321 first parties, with 129 third parties involved. They report 3 instances of respawned cookies being synced over two 3,000 site crawls.

Papadopoulos et. al. [31] investigated the prevalence of cookie synchronization events in mobile web traffic. The study collected 850 mobile users HTTP traffic for 12 months. 263,635 cookie synchronizations were detected over 179M total requests, with 22,329 identifiers shared; 91.996% of the shared identifiers were located in URL parameters, 3.705% in the `Referer` URL, and 3.771% in the URL path. The study reports 5% of identifiers set in TLS sessions being leaked over plain HTTP, as well as the websites visited in the `Referer` field.

Brookman et. al. [18] examined the extent of cross-device tracking visible to an end-user, including cookie synchronizations. The study crawled the Alexa top 100 websites four times each. They report 106 unique third parties syncing identifiers with 210 other third parties.

Englehardt et. al. [20] performed an extensive analysis of online tracking using their open source crawler, OpenWPM. They collected web traffic and browser cookies from two crawls of the top 10K Alexa websites. They report the majority of common third parties embedded in websites participating in cookie synchronization: 45 of the top 50, 85 of the top 100, and 157 of the top 200.

Papadopoulos et. al. [32] investigated TLS privacy breaches facilitated by cookie synchronization, specifically the sharing of websites visited and identifiers set over HTTPS. The top 12K Alexa websites were crawled, with 440K HTTP(S) requests logged. They report 89,479 HTTP(S) syncing requests (i.e. HTTP redirects sharing an identifier) occurring from 32% of the crawled domains; 17,171 unique identifiers were shared with 733 unique domains. Of the 8,398 websites visited over TLS, 2,317 websites were involved in cookie synchronization. Most critically, these TLS websites conducted 2,879 cookie synchronizations with non-TLS websites and leaked 174 HTTPS visits over plaintext. They report 1 in 13 TLS-supported websites performing cookie synchronization over HTTP.

Urban et. al. [35] performed a longitudinal study documenting the effects of the General Data Protection Regulation (GDPR) on cookie synchronization rates in the European Union (EU). 12 measurements were performed, with one occurring a month before the GDPR going into effect (May 2018), and the rest performed each month after. Each measurement instrumented 400 individual browsing profiles (i.e. unique browsing instances). The measurements each crawled an average of 8.5K domains, totalling over 2.5M requests over the year. After the legislation's passing in May 2018, they report an immediate drop in the number of cookie synchronizations per month (~510) in relation to the pre-GDPR measurement (898); a year later, this number decreased to ~480 cookie synchronization per month. The number of third parties conducting cookie synchronizations per month also decreased from ~12K to ~10.2K. The number of involved third parties per month gradually recovered over the year to ~12K. The study claims "cookie synchronization is still used in practice, but its extent is significantly reduced and still declining" in the EU [35]. This claim is not supported by the results of later studies conducted in the EU by Fouad et. al. [22] and Papadogiannakis et. al [30].

Fouad et. al. [22] investigated the role of 1x1 pixel images and other embedded content types in initiating cookie synchronization. They conducted two crawls of the Alexa top 10k domains, and successfully crawled 8,744 domains. They report 34.36% of tracking was initiated by scripts, 23.34% by pixels, 20.01% by text/html, 8.54% by large images, and 4.32% by application or JSON. Of the 8,744 websites crawled, 67.96% were involved in cookie synchronization, with 17,425 third parties involved. Third party identifiers were shared with other third parties in 22.73% of websites with 1,263 unique partners.

Sanchez-Rola et. al. [33] conducted a large scale crawl of the Tranco top 1M most accessed domains list to reconstruct the cookie ecosystem, clarifying known roles and defining novel ones involved in the creation and sharing of cookies. They define the ghost cookie, which is created by an embedded third party script on a first party website that sets a first party cookie. The study claims the existence of a ghosted cookie decreases a first party's control over the cookies their web-page sets on a browser. They report 8.97M cookie synchronization across 387K websites, with the most common sender and receiver relationship (48%) being the own sender to own receiver (i.e. a first party ghost cookies shared with the third party who embedded the script). 52.4% of domains experience at least one cookie synchronization or cookie value overwriting event. Reflecting the results of Papadopoulos et. al. [31, 32], 37.71% of cookies synchronized over HTTP were created in a TLS session.

Papadogiannakis et. al. [30] investigated whether third party trackers respect cookie consent banner choices {`No Action`, `Reject All Cookies`, `Accept All Cookies`}. Their data-set was derived from the Tranco top 850K sites and successfully crawled 27,953 domains containing a Consent Management Platform (CMP). They specify two types of cookie synchronization relationships. They define a `First-Party ID Leak` if a first party identifier is shared with a third party, and a `Third-Party ID Synchronization` if a third party identifier is shared with a third party. When the user takes `No Action`, 52.88% and 24.03% of websites conduct `First-Party ID Leaking` and `Third-Party ID Synchronization`, respectively. When `Rejecting All Cookies`, 56.41% and 26.20% of websites conduct `First-Party ID Leaking` and `Third-Party ID Synchronization`, respectively.

4 PURPOSE

This review intends to document the variety of methods employed to detect cookie synchronization. All studies under review must log HTTP data and label cookie synchronizations from the collected network traffic.

5 DATA-SET COLLECTION METHODS

Crawled Data-set: Web crawlers instrumented include OpenWPM [18, 20, 22, 24, 34, 35], Chromium-based crawlers [14–16, 33], Selenium-based crawlers [1, 12, 32], or custom crawlers [30].

User Data Collection: To collect the HTTP traffic of real users, study-specific browser plugins are installed on a user's browser [21, 28, 29, 31].

Henceforth, the term *user* will refer to the browser instance instrumented, regardless of whether the study collected crawled or real user data.

6 LABELING COOKIE SYNCHRONIZATIONS BY SHARED IDENTIFIERS

6.1 Shared Identifier Heuristic

The majority of cookie synchronization detection methods draw inspiration from the shared identifier heuristic proposed by Olejnik et. al. [28]. This method labels a cookie synchronization iff an identifier is shared in a HTTP request's URL

parameters to an entity other than the entity who set the cookie (i.e. a third party) [21, 22, 28, 29]. An entity can be defined as either a domain or the parent organization of a domain.

Related methods build on this heuristic by additionally extracting identifiers shared with third parties from the URL path of requests [13, 30, 31], `Referer` URL of requests[2] [12, 13, 20, 30–32], redirect `Location` URL [12], nonstandard request and redirect headers [24], or POST request bodies [30].

6.2 Extracting Identifiers from Browser Cookies

6.2.1 What Defines an Identifier? A cookie set on a user's browser is an identifier iff the cookie's value can identify a specific user (i.e. the value is mapped to only one user). These identifying cookie values and the entities who set them are stored to later detect instances of identifiers shared in HTTP traffic. This method confirms that a cookie value shared with a third party can uniquely identify the user who initiated the third party request [12, 18, 20–22, 24, 29–33, 35].

6.2.2 Extracting Browser Cookies. To create the set of all cookies set on a user's browser, cookie values are extracted from the `Set-Cookie` header of HTTP responses [10, 21, 24, 31] or `Cookie` header of HTTP requests [11, 24].

Solomos et. al. [34] use OpenWPM's `javascript_instrument` [20] to log cookie values set by JavaScript embedded in visited web pages.

6.2.3 User Identifier Filtering. The following restrictions are used to filter identifier cookie values from the original browser cookie set.

Value Length Restrictions: Identifiers often have minimum length requirements: cookie values > 10 characters [13, 28, 29, 31], > 8 characters [24], > 7 characters [20, 21, 35], and > 5 characters [30]. Of studies that provide identifier length restrictions, only one provides an upper bound: ≤ 100 characters [20].

Value Character Quality Restrictions: Identifiers can be extracted based on character values. Studies that set character value restrictions only extract cookie values consisting of alphanumeric characters and other common characters [20, 22, 24]. Common character values include [`-`, `_`, `=`], with = indicating a `key=value` pair [20]. Fouad et. al. [22] also consider the comma and period and exclude the equals sign.

Delimiter Parsing: To extract consecutive identifier strings bounded by known characters, cookie values can be parsed (i.e. split) at these common delimiters. All studies that split consecutively shared identifiers consider [`&`, `;`] to be delimiters, except Ghosh et. al. [23] who consider the colon rather than semicolon [12, 13, 20–22, 24, 30, 31, 35].

Similarity Measurement: Identifiers can be extracted by uniqueness. All studies extracting identifiers based on string entropy use the Ratcliff/Obershelp Algorithm [17] with a provided maximum similarity score: eliminate cookie values > 66% similar to another cookie value [20], > 33% similar [12, 13, 18], or not provided [35].

Multiple Values Set for a `Key=Value` Pair: Falahraster et. al. [21] and Urban et. al. [35] exclude any cookie value extracted from a `key=value` pair containing more than one value.

`Key=Value` Pairs with Dynamic Values: Cookie values can be eliminated if the key's value changes over the course of a crawl or user browsing session [12, 13, 20].

Keyword Filtering: Papadogiannakis et. al. [30] use a manually curated list of keywords to eliminate cookie values containing dates, timestamps, regions, locale, URLs, prevalent keywords, consent information (e.g. values of the keys `euconsent`, `eupubconsent`, `__cmpconsnent`, `__cmpiab`), or end in common file extensions.

Filtering Non-Unique Strings: Studies with access to multiple cookie data-sets from multiple crawls or user browsing

[2] As of November 2020, the HTTP `Referrer-Policy` default directive has been updated to `strict-origin-when-cross-origin` to only share the origin of a request. This prevents identifiers from being shared in the path and querystring [9].

sessions can eliminate cookie values present for multiple crawls or users [21, 22, 31, 35].

Session Cookie Values: Session cookies are deleted at the end of a browsing session and their values can be eliminated [11]. Studies that eliminate session cookies examine the `Expires` and `Max-Age` attributes [11] and eliminate values associated with cookies lacking an expiration date [31, 32] or expire earlier than a specified future date: earlier than 90 days [20] or 30 days [12].

6.3 Detecting Identifiers Shared in HTTP Traffic

6.3.1 Labeling Requests to First or Third Parties. Studies that label the party relation of (referrer, request) pairs only label identifiers shared in requests to third parties [13, 18, 20–22, 24, 28–35].

Parent Organization Mapping: Domain names can be mapped to parent organizations using DNS `whois` records and blacklists [13, 21, 31, 32] or the `WhoTracks.me` database [2, 33]. To resolve domain names obfuscated by CNAME cloaking [19], Sanchez et. al. [33] use the NextDNS blocklist [4] to resolve these cloaked domains to known trackers; `tldExtract` [25] is then used to determine the private suffix of each domain; private suffixes are mapped to parent organizations using the `Disconnect` [3], `WhoTracks.me` [2], and `webxray` [27] lists.

String Matching: Domain name string matching is also common, with matches indicating a first party and mismatches indicating a third party [18, 28–30].

Englehardt et. al. Case Study: Englehardt et. al. [20] label request party relations using the Mozilla Public Suffix list [5]; iff the landing page's domain name and public suffix (not including subdomains) do not match a request's domain name and public suffix, the request is labeled as to a third party.

6.3.2 HTTP Identifier Sharing Locations. The research community has examined the following HTTP elements for instances of shared identifiers using exact string matching, with matches indicating a cookie synchronization.

HTTP GET Requests: URL query parameters [12, 13, 18, 20, 22, 24, 28, 30–32], URL path[3] [12, 13, 18, 20, 24, 30–32], `Referer` URL[4] [12, 13, 20, 30–32], and non-standard headers [24].

HTTP Redirects: `Location` URL [12, 13, 20] and non-standard headers [24].

HTTP POST Requests: Request bodies [30].

6.3.3 Papadopoulos et. al. Shared Identifier Labeling Case Study. Papadopoulos et. al. [31, 32] implemented a distinct method of detecting instances of shared identifiers over two cookie synchronization studies.

Rather than using string matching to label instances of shared identifiers, they first extract all ID-looking strings from GET request URL paths, query parameters, and `Referer` headers. An ID-looking string is defined by the same qualities used for filtering identifiers from browser cookies {*Section 6.2*}.

The study stores detected ID-looking strings in a hashtable with the receiving domain. If an ID-looking string is seen for the first time in an HTTP element, the string is added to the hashtable with the requested domain. If an ID-looking string is seen for at least the second time, all requests carrying it are labeled as an ID-sharing event.

Cookie synchronizations are labeled from the ID-sharing event set; iff an ID-looking string present in an ID-sharing event matches a known identifier, the ID-sharing event is labeled a cookie synchronization.

[3]Studies who report examining URLs–without specifying which elements–are assumed to check both the path and querystring.
[4]As of November 2020, the HTTP `Referrer-Policy` default directive has been updated to `strict-origin-when-cross-origin` to only share the origin of a request. This prevents identifiers from being shared in the path and querystring [9].

7 ALTERNATIVE COOKIE SYNCHRONIZATION DETECTION METHODS

7.1 Decision Tree Classifier of Encrypted Identifier Synchronization

Papadopoulos et. al. [31] trained a decision tree model to detect cookie synchronizations of encrypted identifiers. The model does not consider the presence of a shared, known identifier when classifying cookie synchronizations.

The study assumes an equal distribution of HTTP traffic feature variability between cookie synchronization of non-encrypted and encrypted identifiers. The training and testing sets were labeled by non-encrypted cookie synchronizations detected using the study's shared identifier heuristic.

The features selected include requested entity name, type of entity {`Content, Social, Advertising, Analytics, Other`}, URL parameter names, location of hashed identifier {URL parameter, URL path, `Referer` URL parameter}, HTTP status code, browser type, and number of parameters.

7.2 Labeling Cookie Synchronizations in Retargeted Ad Serving Information Flows

Bashir et. al. [14] collect the resource inclusion chain for all websites crawled. At a high level, a cookie synchronization is labeled iff an auction is held by the `publisher-side` and requests between the Supply-Side Platforms (SSP) of the chain directly include a resource.

The study defines the following terminology. `Personas` are individually created to represent 90 unique categories of shoppers by browsing specific products on e-commerce sites. These categories are used to later compare with the qualities of retargeted ads for each `persona`. A `publisher-side` resource chain serves a retargeted ad to a user's browser. `pub` is the root node's publisher domain. `d` is the last entity in a chain and serves the ad. `s` denotes a SSP. `shop` is the e-commerce site domain of the retargeted ad.

Cookie synchronizations are labeled iff `s` and `d` are adjacent at the end of a chain, `d` observes the `persona` at `shop`, and a request from `s` to `d` (or `d` to `s`) is present in a chain prior to the retargeted ad being served [14].

7.3 Labeling Tracker to Tracker Cookie Synchronizations with Pre-Existing Data-sets

Bashir et. al. [15] and Solomos et. al. [34] label any (tracker, tracker) referrer-request pair as a cookie synchronization iff the pair is present on a list of known cookie synchronizing third parties [14, 31].

REFERENCES

[1] [n. d.]. http://docs.seleniumhq.org/
[2] [n. d.]. Bringing transparency to online tracking. https://whotracks.me/
[3] [n. d.]. Disconnectme - Overview. https://github.com/disconnectme
[4] [n. d.]. NextDNS. https://github.com/nextdns
[5] [n. d.]. Public suffix list. https://publicsuffix.org/
[6] 2010. Same origin policy. https://www.w3.org/Security/wiki/Same_Origin_Policy
[7] 2022. Document.cookie - web apis: MDN. https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie
[8] 2022. Referer - http: MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer
[9] 2022. Referrer-policy - http: MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy
[10] 2022. Set-cookie - http: MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie
[11] 2022. Using HTTP cookies - http: MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
[12] Gunes Acar et al. 2014. The Web Never Forgets. *Computer and Communications Security, ACM* (2014), 674–689.
[13] Pushkal Agarwal, Sagar Joglekar, Panagiotis Papadopoulos, Nishanth Sastry, and Nicolas Kourtellis. 2020. Stop tracking me bro! differential tracking of user demographics on hyper-partisan websites. In *Proceedings of The Web Conference 2020*. 1479–1490.
[14] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. 2016. Tracing information flows between ad exchanges using retargeted ads. In *25th USENIX Security Symposium (USENIX Security 16)*. 481–496.

[15] Muhammad Ahmad Bashir and Christo Wilson. 2018. Diffusion of User Tracking Data in the Online Advertising Ecosystem. *Proc. Priv. Enhancing Technol.* 2018, 4 (2018), 85–103.
[16] Eric Bidelman. 2018. Getting started with headless chrome. https://developer.chrome.com/blog/headless-chrome/
[17] Paul Black. 2021. Ratcliff/Obershelp pattern recognition. https://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html
[18] Justin Brookman, Phoebe Rouge, Aaron Alva, and Christina Yeung. 2017. Cross-Device Tracking: Measurement and Disclosures. *Proc. Priv. Enhancing Technol.* 2017, 2 (2017), 133–148.
[19] Romain Cointepas. 2019. CNAME Cloaking, the dangerous disguise of third-party trackers.
[20] Steven Englehardt and Arvind Narayanan. 2016. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 1388–1401.
[21] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. 2016. Tracking personal identifiers across the web. In *International conference on passive and active network measurement*. Springer, 30–41.
[22] Imane Fouad, Nataliia Bielova, Arnaud Legout, and Natasa Sarafijanovic-Djukic. 2018. Missed by filter lists: Detecting unknown third-party trackers with invisible pixels. *arXiv preprint arXiv:1812.01514* (2018).
[23] Arpita Ghosh, Mohammad Mahdian, R Preston McAfee, and Sergei Vassilvitskii. 2015. To match or not to match: Economics of cookie matching in online advertising. *ACM Transactions on Economics and Computation (TEAC)* 3, 2 (2015), 1–18.
[24] Umar Iqbal, Charlie Wolfe, Charles Nguyen, Steven Englehardt, and Zubair Shafiq. 2022. Khaleesi: Breaker of Advertising and Tracking Request Chains. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 2911–2928. https://www.usenix.org/conference/usenixsecurity22/presentation/iqbal
[25] John-Kurkowski. [n. d.]. John-Kurkowski/tldextract: Accurately separates a URL's subdomain, domain, and public suffix, using the public suffix list (PSL). https://github.com/john-kurkowski/tldextract
[26] Samy Kamkar. 2010. Evercookie. https://samy.pl/evercookie/
[27] Tim Libert. [n. d.]. Timlib/webxray: WebXray is a tool for analyzing webpage traffic and content, extracting legal policies, and identifying the companies which collect user data. https://github.com/timlib/webXray
[28] Lukasz Olejnik, Minh-Dung Tran, and Claude Castelluccia. 2014. Selling off privacy at auction. *Proceedings 2014 Network and Distributed System Security Symposium* (2014). https://doi.org/10.14722/ndss.2014.23270
[29] Michalis Pachilakis, Panagiotis Papadopoulos, Nikolaos Laoutaris, Evangelos P Markatos, and Nicolas Kourtellis. 2021. YourAdvalue: Measuring advertising price dynamics without bankrupting user privacy. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3 (2021), 1–26.
[30] Emmanouil Papadogiannakis, Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P Markatos. 2021. User tracking in the post-cookie era: How websites bypass gdpr consent to track users. In *Proceedings of the Web Conference 2021*. 2130–2141.
[31] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. 2019. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *The World Wide Web Conference*. 1432–1442.
[32] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P Markatos. 2018. Exclusive: How the (synced) cookie monster breached my encrypted vpn session. In *Proceedings of the 11th European Workshop on Systems Security*. 1–6.
[33] Iskander Sanchez-Rola, Matteo Dell'Amico, Davide Balzarotti, Pierre-Antoine Vervier, and Leyla Bilge. 2021. Journey to the center of the cookie ecosystem: Unraveling actors' roles and relationships. In *IEEE Symposium on Security and Privacy*.
[34] Konstantinos Solomos, Panagiotis Ilia, Sotiris Ioannidis, and Nicolas Kourtellis. 2019. Clash of the trackers: Measuring the evolution of the online tracking ecosystem. *arXiv preprint arXiv:1907.12860* (2019).
[35] 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 15th ACM Asia Conference on Computer and Communications Security*. 222–235.

Received 27 September 2022; revised NA; accepted NA