Cookie Synchronization: Everything You Always Wanted to Know But Were Afraid to Ask

Panagiotis Papadopoulos
Brave Software
panpap@brave.com

Nicolas Kourtellis
Telefonica Research, Spain
nicolas.kourtellis@telefonica.com

Evangelos P. Markatos
FORTH-ICS, Greece
markatos@ics.forth.gr

ABSTRACT

User data is the primary input of digital advertising, fueling the free Internet as we know it. As a result, web companies invest a lot in elaborate tracking mechanisms to acquire user data that can sell to data markets and advertisers. However, with same-origin policy and cookies as a primary identification mechanism on the web, each tracker knows the same user with a different ID. To mitigate this, Cookie Synchronization (CSync) came to the rescue, facilitating an information sharing channel between 3rd-parties that may or not have direct access to the website the user visits. In the background, with CSync, they merge user data they own, but also reconstruct a user's browsing history, bypassing the same origin policy.

In this paper, we perform a first to our knowledge in-depth study of CSync in the wild, using a year-long weblog from 850 real mobile users. Through our study, we aim to understand the characteristics of the CSync protocol and the impact it has on web users' privacy. For this, we design and implement CONRAD, a holistic mechanism to detect CSync events at real time, and the privacy loss on the user side, even when the synced IDs are obfuscated. Using CONRAD, we find that 97% of the regular web users are exposed to CSync: most of them within the first week of their browsing, and the median userID gets leaked, on average, to 3.5 different domains. Finally, we see that CSync increases the number of domains that track the user by a factor of 6.75.

CCS CONCEPTS

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

KEYWORDS

Cookie Synchronization, Cross-domain tracking, HTTP Cookies

ACM Reference Format:
Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. 2019. Cookie Synchronization: Everything You Always Wanted to Know But Were Afraid to Ask. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313542

1 INTRODUCTION

In the online era, where behavioural advertising fuels the majority of Internet, user privacy has become a commodity that is being bought and sold in a complex, and often ad-hoc, data ecosystem [17, 24, 35, 38]. Users' personal data collected by IT companies constitute a valuable asset, whose quality and quantity significantly affect each company's overall market value [40]. As a consequence, it is of no doubt that in order to gain advantage over their competitors, web companies such as advertisers and trackers participate in a user data collecting spree, aiming to retrieve as much information as possible and form user profiles. These detailed profiles contain personal data [12] such as interests, preferences, personal identifying information, geolocations, etc. [5, 31, 44], and could be sold to 3rd-parties for advertising or other purposes beyond the control of the user [30, 36, 37].

Highlighting the importance of this data collection, web companies have invested a lot in elaborate user tracking mechanisms. The most traditional one includes the use of cookies: they have been commonly used in the Web to save and maintain some kind of state on the web client's side. This state has been used as an identifier to authenticate users across different sessions and domains. Initially, *1st-party* cookies were used to track users when they repeatedly visited the same site, and later, *3rd-party* cookies were invented to track users when they move from one website to another. The same-origin policy (SOP) was invented a few years later [45] to restrict the potential amount of information trackers can collect about a user and share with other 3rd-party platforms.

To overcome this restriction, and create unified identifiers for each user, the ad-industry invented the Cookie Synchronization (CSync) process [19]: a mechanism that can practically "circumvent" the same-origin policy, and allow web companies to share (synchronize) cookies, and match the different IDs they assign for the same user while they browse the web. Sadly, recent results show that most of the 3rd-parties are involved in CSync: 157 of top 200 websites (i.e. 78%) have 3rd-parties which synchronize cookies with at least one other party, and they can reconstruct 62-73% of a user's browsing history [11]. Furthermore, 95% of pages visited contain 3rd-party requests to potential trackers and 78% attempt to transfer unsafe data [46]. Finally, a mechanism for respawing cookies has been identified, with consequences in the reconstruction of users' browsing history, even if they delete their cookies [1].

Although past works highlight the use of CSync across the most popular sites worldwide, they avoid diving into the details and fail to thoroughly explore this increasingly popular technique. In fact, the majority of related works perform their analysis using crawled data by fetching the top Alexa websites. Consequently, little is known on how CSync works in the wild, how many of real users' cookies are getting synced during their everyday browsing, and to what extend it affects the end users' privacy. Importantly, existing studies focus solely on desktop web; however, today more than 52.2% of all web traffic is generated through mobile phones (up from 50.3% in the previous year [42]). This traffic points to a whole new, unexplored ecosystem of mobile Web, where users browse through their very personal devices, while employing a variety of sensors to experience a highly personalized content, but always at the expense of privacy. Thus, what is the impact of CSync in this new mobile ecosystem? *Which are the basic characteristics of CSync, and the mechanics used? How does CSync impact the user's privacy and anonymity on the mobile web?*

In this paper, we aim to answer these questions, by studying Cookie Synchronization using a large, year-long dataset of 850 real mobile users, and exploring in depth its use and growth through time, dominant companies, along with its side-effects on user privacy. The contributions of this work are as follows:

- We design and implement CONRAD: (COokie syNchRonizAtion Detector) a holistic mechanism to detect at real time CSync events and the privacy loss on the user side, even when synced IDs are concealed or obfuscated by participating companies aiming to reduce identifiability from traditional, heuristic-based detection algorithms. In such cases, when non-ID related features are used, our approach achieves high accuracy (84%-90%) and AUC (0.89-0.97).
- We perform the first of its kind, large-scale, longitudinal study of Cookie Synchronization on mobile users in the wild. We perform a passive data collection of the activity of 850 volunteering mobile users, lasting an entire year. This means that the data collected are not crawled, like in past studies, and therefore do not capture a distorted or biased picture of CSync on the Web. Instead, these data provide a rare glimpse of CSync in the mobile space, and an opportunity to study CSync and its impact on real mobile users' privacy and anonymity.
- Using the proposed detection mechanism, we conduct an in-depth privacy analysis of CSync. Our results show that 97% of regular web users are exposed to CSync. In addition, the average user receives ~1 synchronization per 68 HTTP requests, and the median userID gets leaked, on average, to 3.5 different domains. Furthermore, CSync increases the number of domains that track the user by a factor of 6.75. Finally, we detect CSync involved in different scenarios such as breaking-off an SSL session, and exposing userIDs and other personal data in cleartext.

2 COOKIE SYNCHRONIZATION

2.1 How does Cookie Synchronization work?

Figure 1 presents a simple example to understand in practice what is CSync and how it works. Let us assume a user browsing several domains like `website1.com` and `website2.com`, in which there are 3rd-parties like `tracker.com` and `advertiser.com`, respectively. Consequently, these two 3rd-parties have the chance to set their own cookies on the user's browser, in order to re-identify the user in the future. Hence, tracker.com knows the user with the ID `user123`, and advertiser.com knows the same user with the ID `userABC`. Now let us assume that the user lands on a website (say `website3.com`), which includes some JavaScript code from tracker.com but not from advertiser.com. Thus, advertiser.com **does not (and cannot) know which users visit website3.com**. However, as soon as the code of tracker.com is called, a GET request is issued by the browser to tracker.com (step 1), and it responds back with a REDIRECT request (step 2), instructing the user's browser to issue another GET request to its collaborator advertiser.com this time, using a specifically crafted URL (step 3):

GET *advertiser.com?syncID=**user123**&publisher=**website3.com***
Cookie: {cookie_ID=**userABC**}

When advertiser.com receives the above request along with the cookie ID `userABC`, it finds out that `userABC` visited website3.com. To make matters worse, advertiser.com also learns that the user whom tracker.com knows as `user123`, and the user `userABC` is basically one and the same user. Effectively, CSync enabled advertiser.com to collaborate with tracker.com, in order to: (i) find out which users visit website3.com, and (ii) synchronize (i.e., join) two different identities (cookies) of the same user on the web.

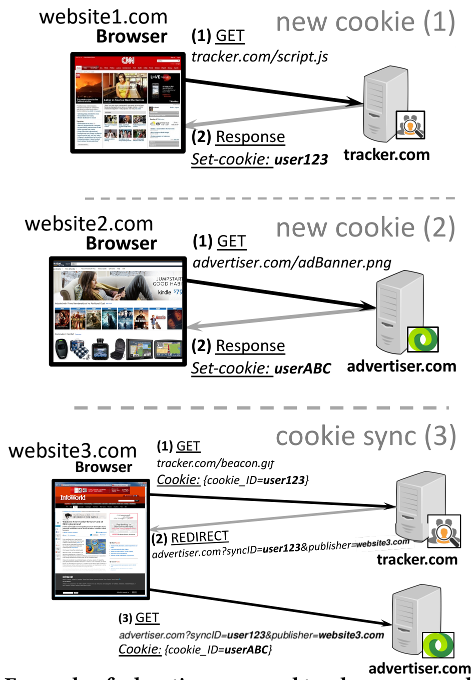

Figure 1: Example of advertiser.com and tracker.com synchronizing their cookieIDs. Interestingly, and without having any code in website3, advertiser.com learns that: (i) cookieIDs userABC==user123 and (ii) userABC has just visited the given website. Finally, both domains can conduct server-to-server user data merges.

2.2 Cookie Synchronization, Personalized Advertising & Privacy Implications

Digital advertising has moved towards a more personalized model, where ad-slots are purchased programmatically (e.g., Real Time Bidding (RTB) based auctions) in auctions, based on how well the profile of the visitor matches the advertised product. Consequently, advertisers need to obtain user data (e.g., interests, behavioral patterns) to use as input in their sophisticated decision engines. The core component for this data purchasing/sharing includes CSync [16].

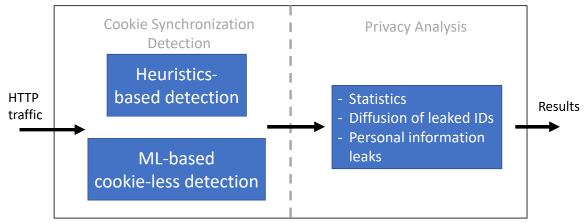
Cookie Synchronization Detection
Heuristics-based detection
ML-based cookie-less detection
HTTP traffic
Privacy Analysis
- Statistics
- Diffusion of leaked IDs
- Personal information leaks
Results

Figure 2: High-level overview of CONRAD's internal components.

It allows trackers and advertisers to perform common user identification and by participating in data markets enrich their knowledge base with user information from several data sources.

There are several privacy implications for the online users who access websites planted with such sophisticated tracking technologies. Using CSync, in practice, advertiser.com learns that: (i) what it knew as `userABC` is also `user123`, and (ii) this user has just visited website3.com. This enables advertiser.com to track a user to a much larger number of websites than was initially thought. Indeed, by collaborating with several trackers, advertiser.com is able to track users across a wide spectrum of websites, even if those websites do not have any collaboration with advertiser.com.

To make matters worse, the ill effects of CSync may reach way back in the past - even up to the time before the invention of CSync. Assume, for example, that someone manages to get access to all data collected by tracker.com, and all the data collected by advertiser.com (e.g., by acquisition [30, 36], merging or hacking of companies [22]). In the absence of CSync, in those two datasets, our user has two different names: `user123` and `userABC`. However, after one single CSync, those two different names can be joined into a single user profile, effectively merging all data in the two datasets. Nowadays, such cases of *server-to-server user data merges* are taking place at a massive scale [11], with the different web companies conducting mutual agreements for data exchanges or purchases, in order to enrich the quality and quantity of their user data warehouses [6, 30].

As if these threats to user privacy were not enough, CSync can rob users of the right to erase their cookies. Indeed, when coupled with other tracking technologies (i.e. evercookie [39], or user fingerprinting [9]), CSync may re-identify web users even after they delete their cookies. Specifically, when a user erases her browser state and restarts browsing, trackers usually place and sync a new set of userIDs, and eventually reconstruct a new browsing history. But if one of them manages to respawn [1] its cookie (e.g. through evercookie [39]), then through CSync, all of them can link the user's browsing histories from before and after her state erasure. Consequently: (i) users are not able to abolish their assigned userIDs even after carefully erasing their set cookies, and (ii) trackers are enabled to link user's history across state resets.

3 COOKIE SYNCHRONIZATION DETECTION

In this paper, we design CONRAD, a holistic methodology to detect CSync events in real time, on the user side. CONRAD monitors the HTTP(S) traffic of the user on the browser level and detects userIDs when shared from one domain to the other. To achieve that, it uses a (i) Heuristics-based (stateful) detection mechanism (Section 3.1), where the IDs from cookies are tainted and alert is raised when they are exfiltrated to a domain other than the owner of the cookie. However, as also presented in past studies [2], more and more companies include encryption (or cryptographic hashing) in the CSync-related APIs, thus concealing the synced IDs. To deal with these cases, CONRAD uses a (ii) ML-based (stateless) detection mechanism (Section 3.2) capable of classifying with high accuracy such possible concealed synchronizations, without relying on any previously stored cookie IDs, but only using characteristics from the connections themselves. Figure 2, provides a high level overview of CONRAD's internal components.

Table 1: Examples of userIDs synchronized among various domains.

URLs of Cookie Synchronization HTTP Requests
1. a.atemda.com/id/csync?s=**L2zaWQvMS9lkLzMxOUwOTUw**
2. bidtheater.com/UserMatch.ashx?bidderid=23&bidderuid=**L2zaWQvMS9lkLzMxOUwOTUw**&expiration=1426598931
3. d.turn.com/r/id/**L2zaWQvMS9lkLzMxOUwOTUw**/mpid/

For each CSync detection method, CONRAD extracts its information flow: the chain of domains that share the synced ID, the domain that triggered the CSync event, the domains that (without having access to the website) used this sync request to set and sync their own userIDs (see Section 5.1). This way, our tool is able to measure the diffusion of anonymity loss for the given user by analysing what number of their overall userIDs budget got synced, and to how many 3rd-party domains. In addition, by using a simple pattern matching technique, CONRAD extracts possible personal information leaks tailored with the synced ID (see Section 5.5).

Although there are several existing techniques for detecting ID-sharing events even when cookies are encrypted, accurate CSync detection in real time is a hard task. The main advantages of our approach, contrary to existing detection mechanisms [1, 2, 11] are as follows: (i) It offers the ability to detect synchronizations when the userID is embedded not only in the URL's parameter, but also in its path (either in case of request/response URL or Location URL of the referrer). (ii) By filtering-out domains of the same provider, our approach can discriminate between intentional CSync and unequivocally legitimate cases of internal ID sharing, thus avoiding false positives. (iii) It is capable of detecting Cookie Synchronization *at real time*, even when shared IDs are encrypted.

3.1 Heuristics-based detection

Technically, as we see in Table 1, CSync is nothing more than a request from the user's browser to a 3rd-party domain carrying (at least one) parameter that constitutes a unique ID set by the calling domain. However, what CSync typically enables is a multiple, back-to-back operation with several 3rd-party domains getting updated with one particular ID. This multiple synchronization happens by utilizing URLs of HTTP requests (i.e., the Location HTTP header), in which the cookie ID (i.e., the userID) of the triggering domain is embedded. The userID may be embedded in the: (i) parameters of the URL, (ii) URL path, or (iii) referrer field. In some cases, detection may be straightforward: one can simply look for specific parameter names (e.g., syncid, user_id, uuid). However, different companies use different APIs and parameter naming; relying only on string matching for Cookie Synchronization detection will lead to a large number of false negatives in case of newcomer syncing domains. To remedy this, by extending previous work [29], we design a stateful heuristics-based detection algorithm, which relies on the previously set cookies to taint userIDs that may get synced with

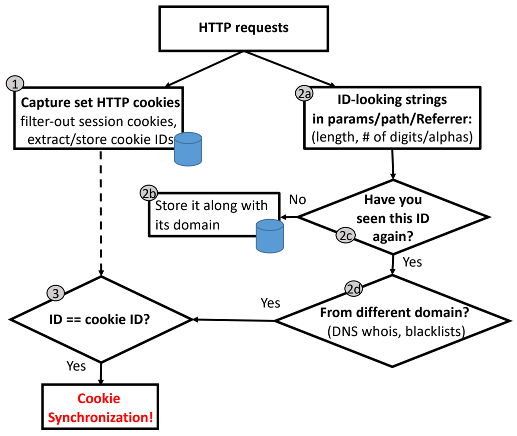

Figure 3: Heuristics-based CSync detection mechanism.

domains different than the cookie setter[1]. In particular, our Cookie Synchronization detection methodology includes the following steps, which are also illustrated in Figure 3:

(1) We extract all cookies set on the user's browser. To accomplish that, we parse all HTTP requests in our dataset and extract all *Set-Cookie* requests.
  (a) We filter out all session cookies. These are cookies without expiration date, that get deleted after the end of a session.
  (b) We parse the cookie value strings using common delimiters (i.e. ":", "&"). By extracting potentially identifying strings (cookie IDs), we create a list with the cookie IDs that could uniquely identify the user in the future.
(2) We detect possible ID-sharing events in the HTTP requests:
  (a) We identify ID-looking strings that are unique per user, carried either (i) as parameters in the URL, (ii) in URL path, or (iii) in the referrer field. As ID-looking strings, we define strings with specific length ($>$ 10 characters) – false positives do not matter at this point.
  (b) If this ID is seen for the first time, it is stored in a hashtable along with the URL's domain (receiver of the ID).
  (c) If this ID has already been seen, we consider it as a shared ID and the requests carrying it as *ID-sharing requests*.
  (d) To check if the above ID-sharing requests regard different domains, we use several external sources (DNS whois, blacklists etc.) to filter-out cases where the IDs are shared among domains owned by the same provider (e.g., amazon.com and amazonaws.com) [25]. This way, our approach can discriminate between intentional ID leaking and legitimate cases of internal ID-sharing, thus avoiding false positives.
(3) Finally, to verify if the detected shared ID is a userID able to uniquely identify a user, we search this ID within the list of cookie IDs extracted in the first step. If there is a match, then we consider this request as CSync.

3.2 Cookie-less detection

It is apparent that in order for the above methodology to be a viable CSync detection method, cookie IDs need to be shared in plaintext.

[1]A known limitation of this approach is its inability of capturing a small portion of 1st-party cookies set by javascript[7]

Table 2: Examples of Cookie Synchronization between 3rd-parties with plaintext and encrypted cookie IDs.

ID syncs beyond the 1-1 of source domain (plaintext):
Domain syncs ID with Tracker1 ID1; $\longrightarrow$ ID1=ID;
Domain syncs ID with Tracker2 ID2; $\longrightarrow$ ID2=ID;
Tracker1 syncs ID1 with Tracker2 ID2; $\longrightarrow$ ID2=ID1=ID;
ID syncs beyond the 1-1 of source domain (encrypted):
Domain syncs h(ID,A) with Tracker1 ID1; $\longrightarrow$ ID1=h(ID,A);
Domain syncs h(ID,B) with Tracker2 ID2; $\longrightarrow$ ID2=h(ID,B);
Tracker1 syncs ID1 with Tracker2 ID2; $\longrightarrow$ h(ID,A)!=h(ID,B), i.e., ID1!=ID2;

However, major web companies such as DoubleClick [19] have started encrypting the cookie ID in an attempt to protect the actual cookie from being revealed to unwanted parties that may snoop the user's traffic (plugins or even ISPs).

While the former case is obvious why companies would want to block such snooping, the latter case may not be clear why and, thus, we discuss this case further. In particular, under the traditional, plaintext case of cookie ID syncing, the same source company can sync independently with multiple 3rd-parties for the same user cookie ID. Thus, no-one forbids these other 3rd-parties from syncing their IDs with each other, and find out that they have information about the same user, something that goes beyond what the source company intended to do (top example of Table 2). With hashing or encryption of the cookie ID, these 3rd-parties are unable to do this syncing (bottom example of Table 2). As a consequence of this encryption, CSync events can proceed undetected, if the previously used detection method is employed.

To address this scenario, in this paper, we propose a novel method for identifying CSync which is oblivious to the IDs shared. This mechanism is able to identify with high accuracy CSync events in web traffic, even when the leaked IDs are protected and cannot be matched. To build this mechanism, we employ machine learning methods, which we train on the ground truth datasets created with the previous, heuristic-based technique. In particular, we analyze various features extracted from the web traffic due to CSync, and train a machine learning classifier to automatically classify a new HTTP connection as being a CSync event or not. Here, we make the assumption that the various features used to characterize, and eventually detect, CSync with plaintext IDs, are equally used, and have the same distributions and variability as in the CSync with encrypted IDs. We believe this is a reasonable assumption, since the companies employing encrypted IDs are not expected to change the rest of their mechanism which delivers these IDs and triggers CSync with their partners; these companies only want to obfuscate the IDs to avoid further, and unwanted, CSync.

For training the classifier, we extract various relevant features from the network traces. As ground truth, we use confirmed CSync events which were detected with the heuristics-based method described earlier. Beyond these confirmed events, there are *id-sharing* events which were first selected by the method as potential CSync events, but eventually were rejected as *non-CSync*, as they did not match cookie IDs already seen by the method (step 1 in Figure 3).

The features available for these network events can be several; we constrain the learning algorithm to use only features available at run time, and during the user's browsing to various websites:

- EntityName: {domain of recipient company}
- TypeOfEntity: {Content, Social, Advertising, Analytics, Other}
- ParamName: {aid, u, guidm, subuid, tuid, etc.}

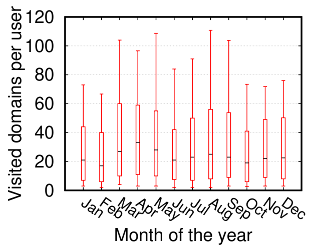

Figure 4: Distribution of number of unique domains visited per user, per month. The median user in our dataset visits 20 - 30 different domains per month.

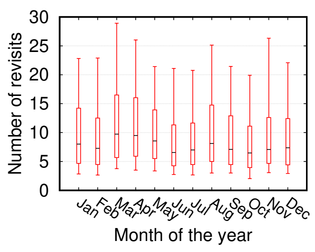

Figure 5: Distribution of number of times a user revisits the same domain per month. The median user revisits a domain around 7-10 times per month.

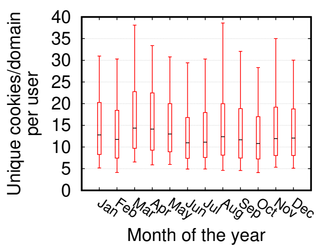

Figure 6: Number of (first and 3rd-party) cookies per domain per user. We see that the median user receives 12.25 cookies, on average, per visited website.

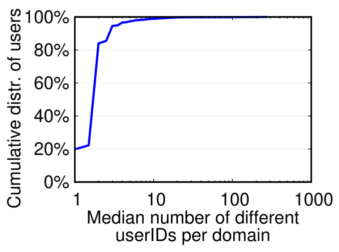

Figure 7: Unique userIDs set per domain, across the year. 80% of users are known to a single domain with only ~2 aliases, on average.

Table 3: Summary of contents in our dataset.

Description	#	Description	#
Total mobile users	850	Unique shared IDs (S)	68215
Requests captured	179M	Unique userIDs synced ($C \cap S$)	22329
Unique Cookies (C)	8.97M	CSync requests	263635
ID sharing requests	412805		

- WhereFound: {parameter in URL, parameter in Referrer, in the URL path}
- StatusCode: {200, 201, 202, 204, etc.}
- Browser: {Firefox, Chrome, Internet Explorer, etc.}
- NoOfParams: {0, 1, 2, ..., etc.}

Various machine learning algorithms can be applied: Random Forest, Support Vector Machines, Naive Bayes, and even more advanced methods such as Neural Networks. However, a balance must be found between training the given algorithm to reach a good accuracy, the computation cost for this training, as well as the capability of the algorithm to be used at real time on the user device.

This method aims to address two possible scenarios that can arise while IDs are being shared: First, we consider the realistic scenario that an already identified set of id-sharings are candidate CSync events (as found by the heuristics-based method), but cannot be validated as CSyncs because of the cookie ID being encrypted or unavailable, and therefore not matching the repository of IDs. Second, we consider the scenario where various HTTP connections are ingested by the method, and it needs to decide at run-time which are CSync events and which are not. This case is a more generalized version of the previous scenario, and attempts to detect CSync events, as an alternative method to the heuristic-based approach. In either of the two cases, we follow a generally accepted methodology (e.g., [34]) that separates training such a machine learning model, from applying it at real-time. The training can be performed offline, on an existing dataset (e.g., the one we collected, or from anonymous user network traffic donations), and the model trained can be distributed accordingly with the tool at hand. Then, the tool can apply the classifier on each network connection under question, for real-time classification.

4 DATASET

In this section, we describe the data collection process and our year long dataset. In order to collect data from real users, we set a group of proxies fronted by a load balancer, and gathered 850 volunteering users residing in the same country. These users agreed to strip their browsers from any previous state (i.e., cookies, cache, webStorage) and have their devices to continuously redirect their network traffic through our proxies for 12 consecutive months (2015-2016). They signed a consent form allowing us to collect and analyze their data during this period, and publish any anonymized results. They were well-aware of the purposes of the data collection, and were compensated with free data plan, as long as they were using the proxies. Before the analysis was performed, all data were anonymized and never shared with any other domains.

Given the long duration of the experiment, and in order not to jeopardize the confidentiality of the volunteering users' secure sessions, we capture only their HTTP mobile traffic (same method can be applied for HTTPS). On the server-side and based on the user agent of each request, we filtered out any possible app-related traffic. Overall, we collected a dataset containing a total of 179M requests, spanning an entire year. Table 3 summarizes the contents and CSync findings in our dataset.

Users: To analyse our dataset, we create a simple weblog parser. As noted earlier, this dataset consists of web browser traffic from the mobile devices of 850 users. After separating the flows of each one of them, we produced their timelines, and in Figure 4, we present the number of different domains each user visits per month. As we see from the distribution (Percentiles: 10th, 25th, 50th, 75th, 90th), the median user in our dataset visits 20-30 different domains depending on the month (we observe a seasonal phenomenon with increases during spring break and summer holidays).

Similarly, in Figure 5, we present the number of times each of these domains gets revisited by the median user in our dataset. In every revisit, there is a new request that asks the user's browser if there is a previously set cookie. If this *Get-Cookie* request regards a previously set cookie, this means that the domain already knows the user and we consider it as a revisit. We observe that the median user revisits around 7-10 times the same domain from their mobile browser. Again, we observe the seasonal phenomenon as earlier, when users tend to have more time to browse the web. Also, the 75th percentile of the users may revisit the same website more than 15 times (March, August).

Cookies: To have a good view of the cookie activity of users, we extract all (1st & 3rd-party) cookies set in users' browsers across the year, and in Figure 6, we plot the distribution of number of cookies per visited website. The median user receives a fairly constant number of 12.25 cookies per visited website per month, on average.

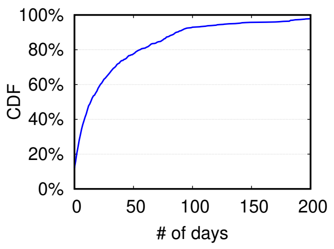

Figure 8: Distribution of time taken for first CSync to appear per user. 20% of users get their first userID synced in 1 day or less.

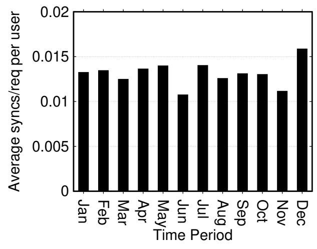

Figure 9: CSyncs per request for the average user across the year. The average user receives 1 synchronization every 68 requests.

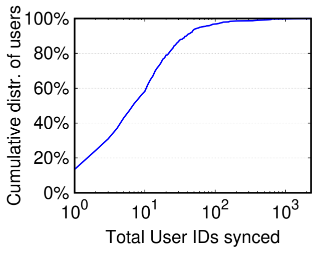

Figure 10: Distribution of the synced userIDs per user. The median user has 7 userIDs synced, when 3% of users has up to 100.

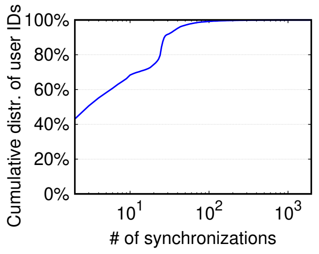

Figure 11: Distribution of synchronizations per userID. The median userID gets synced with 3.5 different domains.

Table 4: Breakdown of the CSync triggering factors.

	Initiator	Portion
(i)	Publisher syncs its userID	2.692%
(ii)	Embedded 3rd-party triggers syncing of its own set userID	49.668%
(iii)	3rd-party uses sync request to share its own set userID	45.697%
(iv)	3rd-party uses sync request to share with other domains the publisher's set userID	0.2658%

Next, we extract the unique identifiers (i.e., cookie ID) set in these cookies. A cookie ID constitutes a unique string of characters that websites and servers associate with the browser and, thus, the user who stored the cookie. Thereby, here, we consider a cookie ID as a unique user identifier called userID. As it can be seen in Table 3, in our dataset, there are almost 9 million such unique IDs.

In Figure 7, we plot the distribution of the number of unique userIDs assigned to the users per domain. The vast majority of the users (80%) receive, on average, only 2.2 userIDs per domain, across the year. This means that users tend not to erase their cookies frequently, thus, allowing web domains to accurately identify them through time and during the users' browsing. Only 1.13% of users erase their cookies (either manually or by browsing with Private/Incognito Browsing), receiving more than 9.5 different userIDs per domain, on average. This means that it is very rare for a domain to meet a previously known user with a different alias.

5 PRIVACY ANALYSIS

By applying CONRAD in our dataset, we find several IDs passed from one domain to another. We detect 68215 such unique shared IDs. From these IDs, 22329 were actually cookie IDs from previously set cookies that were synced among different domains. In total, these cookie IDs were found in 263635 synchronization events (see Table 3 for a summary). From the CSyncs detected in our dataset, userIDs were found in 91.996% of the cases inside the URL parameters, 3.705% in the Referrer URL, and in 3.771% of the cases in the URL path. Thus, CONRAD was able to detect 3.771% more cases of CSyncs than existing detection methods [1, 29, 32].

5.1 Initiation of Cookie Synchronization

First, we correlate the cookie domain (the setter), the synchronizing request's Referrer field, and the publisher that the user visited, in order to extract the domain that triggered the CSync on the user's browser. As seen in Table 4, there are 4 distinct cases: (i) the CSync was initiated by the publisher who syncs the userID he assigned for the user: we find 2.692% of these cases in our dataset, (ii) the synchronization was initiated from a publisher's iframe, by the guest 3rd-party which syncs its own userID (49.668%), (iii) a 3rd-party which participated in a previous synchronization (case (i) or (ii) above) and uses the sync request to share its own userID (45.697%). Lastly, there is a rare case (iv), where a 3rd-party participating in a previous synchronization of the publisher's userID (case (i)) initiates a new round of syncs while it continues to share the publisher's userID (0.2658%). Obviously, in case (iv), the initiating 3rd-party shares with its 3rd-party affiliates, a userID assigned by a domain (the publisher) beyond its control, and possibly awareness.[2]

5.2 How are users exposed to CSync?

CSync impacts users' privacy by leaking assigned userIDs, and sharing them with 3rd-parties. In our dataset, we see that for users with regular activity on the web ($>$ 10 requests per day), **97% were exposed to CSync at least once**. This means that CSync constitutes a phenomenon affecting the totality of online users. Next, we study how long it takes for the first synchronization to happen, or in effect, how quickly a user gets exposed to CSync after she starts browsing. Recall that as mentioned in Section 4, all participating users, during bootstrapping phase, had all state from their browsers erased. This means that our proxy was able to capture the very first cookie that was set during the user's monitoring period. Of course, the time depends on the browsing patterns of each user, however, as we see in Figure 8, a median user experiences at least one CSync within the first week of browsing. In fact, a significant **20% of users get their first userIDs synced in 1 day or less**. It is worth noting that users tend to browse the same top websites repeatedly (e.g., facebook.com, twitter.com, cnn.com), so the set cookies are already shared and no sync is fired.

Next, we investigate if the synchronizations the users are exposed to, change over time. Hence, we extract CSyncs per user, and normalize with the user's total number of requests. In Figure 9, we plot the average synchronizations per request across the year. As shown, CSync is persistent through the duration of an entire year, with the user being exposed to a steady number of synchronizations across time. Specifically, we see that **the average user receives around 1 synchronization per 68 requests**.

[2]We reported all such cases and notified the respected publishers.

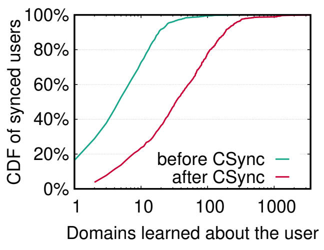

Figure 12: Distribution of domains learning at least a userID of a user (with/without the effect of CSync). After syncing, the domains that learned about the median user grew by 6.75×.

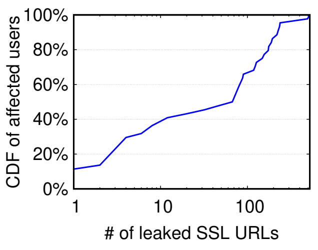

Figure 13: Distribution of the leaked TLS URLs per affected user. The median user has 70 TLS URLs leaked through Cookie Synchronization, when the 90th percentile has up to 226 TLS URLs leaked.

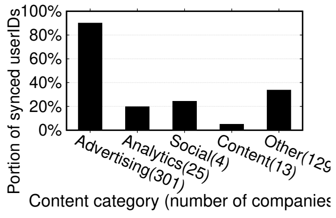

Figure 14: Portion of synced userIDs learned per content category. As expected, ad-related companies learned the vast majority (90%) of the total synced userIDs in our dataset.

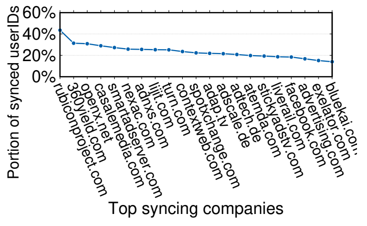

Figure 15: Portion of synced userIDs learned per tracker: 3 trackers learn more than 30% of the total userIDs in our dataset; 14 trackers learn more than 20% of userIDs each.

Considering the different userIDs that trackers may assign to a user, in Figure 10, we measure the number of unique userIDs that got synced per user. Evidently, **a median user gets up to 6.5 userIDs synced**, and 3% of users has up to 100 userIDs synced. It becomes apparent that the IDs of a user may leak to multiple 3rd-party domains through CSync. To measure the userID leak diffusion, in Figure 11, we plot the distribution of synchronizing requests per userID. As we see, **the median userID gets leaked, on average, to 3.5 different domains**. There is also a significant 14% that gets leaked to up to 28 different 3rd-parties.

To better understand the effect of CSync on the diffusion of the overall user privacy, we measure for each user the number of domains that learned about them (i.e., learned at least one of their userID) before and after CSyncs. From the distribution in Figure 12, **the domains that learned about the median user after CSyncs grew by a factor of 6.75**, and for 22% of users this factor becomes > 10. This means that before the rise of CSync, when the user visited a website, the domains that could track them were only the publisher and the included 3rd-parties, but in an independent fashion. However, with the introduction of CSync, the number of domains that can track the user drastically increased (6.75x for median user), severely decreasing their online anonymity.

5.3 Buy 1 - Get 4 for free: ID bundling and Universal IDs

We find **63 cases of domains which set on the users' browsers cookies with userIDs previously set by other domains**. For example, we see the popular `baidu.com`, the world's eighth-largest Internet company by revenue, storing a cookie with an ID $baiduid = \{idA\}$, and more than 5 different domains after this incident setting their own cookie using the same ID $baiduid = \{idA\}$. This by-product of CSync, enables trackers to use *universal IDs*, thus, bluntly violating the same-origin policy and merging directly (without background matching) the data they own about particular users.

Table 5: Example of an *ID Summary* stored on the user's browser. It includes userIDs and expiration dates used for the particular user by 4 different domains.

ID Summary stored in cookie by adap.tv
"key=**valueclickinc**:value=708b532c-5128-4b00-a4f2-2b1fac03de81:expiresat=wed apr 01 15:03:42 pdt 2015,key=**mediamathinc**:value=60e05435-9357-4b00-8135-273a46820ef2:expiresat=thu mar 19 01:09:47 pst 2015,key=**turn**:value=2684830505759170345:expiresat=fri mar 06 16:43:34 pst 2015,key=**rocketfuelinc**:value=639511 149771413484:expiresat=sun mar 29 15:43:36 pst 2015"

In addition, we find **131 cases of domains storing in cookies the results of their CSyncs, thus composing *ID Summaries***. In these summaries, we see the userIDs that other domains use for the particular user previously obtained by CSyncs. An example of such summaries in JSON is shown in Table 5. As one can see, the cookie set by `adap.tv` includes the userIDs and cookie expiration dates of `valueclick.com`, `mediamath.com`, `turn.com` and `rocketfuel.com`. In our dataset, we find at least 3 such companies providing *ID Summaries* to other collaborating domains. This user-side info allows (i) the synchronizing domains to learn more userIDs through a single synchronization request, and (ii) `adap.tv` to re-spawn any deleted or expired cookies of the participating domains at any time, just by launching another CSync.

5.4 Spilling userIDs out of TLS

It is well-known that mixing encrypted and non encrypted sessions in TLS is a bad tactic [10]. Using TLS, everything except IPs and ports is encrypted [15] in a HTTP connection. As a result, although there are sophisticated estimation techniques [18], no observer can monitor what the user is actually browsing over TLS, or the IDs assigned to her. Given that our proxy is monitoring only HTTP traffic (see Section 4), one would expect that no information from secure TLS sessions would be captured. To our surprise, and as already found in [33], in our dataset we see userIDs that originated from TLS sessions getting leaked over plain HTTP to plain 3rd parties. As a result, *any curious, in-path observer (e.g., a snooping ISP) can eavesdrop the leaked userIDs*[3]. To make matters worse, we see that the same ID leaking requests have referrer fields[4], which leak the particular webpage the user visited over TLS (e.g., the particular article of the news site she read), thus leaking her interests.

To reproduce this leak, we manually visit the TLS protected websites where ID-spilling was found and we see it is caused by

[3]As soon as we verified this leak, we notified publishers and also our volunteering users who updated their signed consent.

[4]There are specific directives [23] for Referrer field hiding when referring over TLS visited domains.

Table 6: Example of ID-spill from SSL in our dataset.

Role	Domain
Visited website:	https://financialexpress.com
Cookie setter: SetCookie:	https://tapad.com **D0821FA0-8A80-4D9E-BC85-C40EAC4E4FF5**
Cookie syncer:	http://delivery.swid.switchadhub.com/adserver/user_sync.php?SWID=cf43265166a9ccf5f6fd0472f23776fa&sKey=PM2&sVal=**D0821FA0-8A80-4D9E-BC85-C40EAC4E4FF5** referrer: **financialexpress.com** Get-cookie: {cf43265166a9ccf5f6fd0472f23776fa}
Cookie syncer:	http://tags.bluekai.com/site/3096?id=**D0821FA0-8A80-4D9E-BC85-C40EAC4E4FF5** referrer: **financialexpress.com** Get-cookie: {c57b29d1-f8e2-11e7-ac1b-0242ac110005}

CSync events that sync a userID from a TLS cookie with non-TLS 3rd parties. In Table 6, we present one such real case we observe. As shown, while visiting over TLS the page *https://financialexpress.com*, two CSyncs are performed: *https://tapad.com* advertiser shares with *http://switchadhub.com* and *http://bluekai.com* the ID it assigned to the user. This way, the latter two tracking domains sync their set-cookies with the one of *https://tapad.com*. However, by doing that over plain HTTP, the visited webpage gets leaked through the referrer field to a monitoring entity, even when users browse through proxies or VPN, or even Tor. In addition, this entity from now on can re-identify the user in the web, just by monitoring the userIDs of cookies in requests destined to *http://switchadhub.com* and *http://bluekai.com*, even if the user's IP address is frequently changed. Obviously, the more 3rd-parties were participating in the Cookie Synchronization, the easier it would be for the monitoring entity to capture HTTP requests loaded with these synced cookies and, thus, re-identify the user using a secure VPN.

We measured such cases in our dataset and found **44 users (5%) affected by the ID-spilling of CSync**. The majority of leaked domains regard popular content providers, where an eavesdropper from the referrer field, apart from the domain, can also see sensitive information. From Figure 13, the median (90th percentile) user has 70 (226) TLS URLs leaked through CSync.

5.5 Sensitive information leaked with userIDs

The websites a user browses can easily leak through the referrer field during a CSync. Moving beyond this type of basic leak, in our dataset we find several cases of privacy-sensitive information passed to the syncing domain regarding the particular user with the particular synced ID. By deploying a simple string matching script, we look for keywords (e.g., gender, age, name, etc.) and find:

- 13 syncs leaking the user's city level location
- 2 syncs leaking the user's registered phone number
- 10 syncs leaking the user's gender
- 9 syncs leaking the exact user's age
- 3 syncs leaking the user's full birth date
- 2 syncs leaking the user's first and last name
- 16 syncs leaking the user's email address
- 4 syncs leaking user login credentials: username/password

The above information constitutes not only a severe privacy threat for the user, but can also enables potential impersonation attacks.

5.6 Who are the dominant CSync players?

In order to assess the content that CSync parties provide, we extract all domains involved in CSync, and using EasyList, EasyPrivacy [14] and the blacklist of the Disconnect browser extension [8] (enriched

Table 7: Performance of decision tree model trained on different subsets of features available at runtime for classification, given already identified id-sharing entries, and 10 cross-fold validation.

Feature subset	F	TPR	FPR	PR	RC	FM	AUC
NoOfParams*	1	0.639	0.639	0.408	0.639	0.498	0.500
WhereFound+	1	0.643	0.610	0.612	0.643	0.535	0.602
StatusCode*+	1	0.648	0.619	0.723	0.648	0.523	0.633
TypeOfEntity*+	1	0.735	0.432	0.752	0.735	0.701	0.661
Browser*+	1	0.700	0.492	0.710	0.700	0.651	0.628
ParamName+	1	0.815	0.295	0.828	0.815	0.803	0.834
EntityName*+	1	0.803	0.295	0.806	0.803	0.793	0.854
{id-less}*	5	0.840	0.242	0.845	0.840	0.834	0.887
{high imp.}+	6	0.870	0.206	0.877	0.870	0.865	0.919
ALL	9	0.900	0.144	0.901	0.900	0.898	0.946

with our additions after manual inspection), we categorize them according to the content delivered. This way, we create five categories of domains related with: (i) Advertising, (ii) Analytics, (iii) Social, (iv) 3rd-party content (e.g., CDNs, widgets, etc.), and (v) Other.

As we saw in Figure 11, the median userID gets shared with more than one domains. We find that **ad-related domains participate in more than 75% of the overall CSyncs through the year**. Consequently, as we can also observe in Figure 14, **ad-related domains have learned as much as 90% of all userIDs that got synced**, with Social and Analytics -related domains following with 24% and 20% respectively. In Figure 15, we plot the top 20 companies[5] that learned the biggest portion of the total userIDs through CSyncs in our dataset. Interestingly, the top 3 companies (i.e., rubiconproject.com, 360yield.com and openx.net) learned more than 30% of all userIDs in our dataset each. There is also a significant number (14) of companies that learn more than 20% of userIDs.

6 COOKIE-LESS DETECTION

In this section, we explore two different scenarios outlined in Section 3.2 regarding the detection of CSync via ID sharing, while such IDs may be obfuscated to remove the possibility of matching them with past IDs shared between entities. First, we explore the scenario where IDs have been shared, detected by the heuristic-based approach, but have not yet been confirmed as CSync events. That is, we consider an already identified set of id-sharings, which are candidate CSync events, but cannot be validated as CSyncs because of the cookie ID being encrypted or unavailable (Section 6.1). Second, we take a step back and consider the more general case where various HTTP connections are ingested by the method, and it needs to decide at run-time which are CSync events and which are not based on given features (Section 6.2).

Towards this end, we train and test the classifier in these two experiments. We remind the reader of the assumption made earlier: the distributions of the features describing the CSync events with unencrypted IDs, have the same variability in the cases of encrypted IDs, and therefore can be used for the detection of such cases. This assumption allows us to handle the problem as an out-of-sample estimation, leaving as future work the final validation with a set of ground-truth data of encrypted IDs that we also know their unencrypted versions.

Data and Features: Based on the ground truth data presented earlier with the heuristic-based technique, we have $412.8k$ *id-sharing* events, from which $263.6k$ are confirmed CSync, and $149.2k$ are

[5]In a dataset with both HTTP and HTTPS traffic, market shares may differ since there are companies operating over SSL only (e.g., DoubleClick)

Table 8: Performance of decision tree model trained on different subsets of features available at runtime for classification, given a pre-filter for ID-looking strings. All results besides the last row are with balanced dataset across the three classes, and 10-cross fold validation. The last row's results are computed given an unseen, and unbalanced test set, maintaining the original ratio of classes.

Feature subset	F	TPR	FPR	PR	RC	FM	AUC
NoOfParams*	1	0.541	0.314	0.584	0.541	0.495	0.706
StatusCode*+	1	0.666	0.229	0.673	0.666	0.598	0.764
TypeOfEntity*+	1	0.760	0.162	0.724	0.760	0.695	0.834
EntityName*+	1	0.865	0.075	0.863	0.865	0.860	0.962
ParamName+	1	0.870	0.083	0.878	0.870	0.859	0.953
{id-less}*	4	0.904	0.057	0.904	0.904	0.898	0.973
{high imp.}+	4	0.919	0.051	0.923	0.919	0.914	0.978
ALL	5	0.920	0.051	0.925	0.920	0.916	0.978
Unbalanced	5	0.981	0.004	0.989	0.981	0.984	0.999

identified as *non-CSync*. The features available for these events can be various, as already explained in 3.2. The ones we use are features available at run time, and during the user's browsing to websites.

Algorithms: The final machine learning classifier used is decision tree-based. Others like Random Forest, Support Vector Machines and Naive Bayes were tested, but the decision tree algorithm outperformed them, with significantly less computation and memory overhead. Indeed, more advanced methods can be used, such as Neural Networks, but since the decision tree-based algorithm works very well, we leave this exploration for the future.

Metrics: To evaluate the performance of the classifier on the different classes and available features (F), standard machine learning metrics were used such as Precision (PR), Recall (RC), F-measure (FM), True Positive rate (TPR), False Positive rate (FPR), and area under the receiver operating curve (AUC).

6.1 CSync in ID-sharing HTTP

In this experiment, we assume there is already in place an existing technique for analysis of the HTTP traffic of the user, similar to the method outlined in Figure 3. However, there are candidate CSync events that cannot be confirmed, as the IDs cannot be matched with SET cookie IDs, either because these actions are not available to the method, or because the IDs are encrypted.

In this case, a machine learning classifier can be trained to detect if an id-sharing HTTP request is a true CSync event, by matching its pattern to past verified CSync events. For this experiment, we use two classes: the CSync events and the *id-sharing but non-CSync* events, to train and test a decision tree classifier under different subsets of features. The training and testing was performed using 10 cross-fold validation process. The results are shown in Table 7. We observe that independently, each of the features considered have some predictive power, except from the *NoOfParams* feature. When the most important features (using information gain as metric) are combined, a weighted AUC of 0.919 is achieved. When we select non-ID related features, a weighted AUC=0.887 is reached, and with all features the classifier can reach a weighted AUC=0.946.

6.2 CSync in HTTP with ID looking strings

In this setup, we assume there is a simple HTTP pre-filter, keeping connections with ID looking strings for further investigation. This is a necessary step to reduce the run-time workload of the classifier, as connections relevant to the task are only ∼20% of the overall HTTP workload. Then, the classifier has to decide which of the 3

Table 9: Detailed performance of decision tree model trained on different subsets of features in a balanced dataset, and tested on an unseen, and unbalanced test set, which maintains the original ratio of classes (last row of Table 8). C: CSync, ICS: *id-sharing but non-CSync*, O: Other, WA: weighted average.

Class	TPR	FPR	PR	RC	FM	AUC
CS	0.988	0.014	0.534	0.988	0.693	0.998
ICS	0.603	0.005	0.458	0.603	0.521	0.990
O	0.984	0.004	1.000	0.984	0.992	0.999
WA	0.981	0.004	0.989	0.981	0.984	0.999

classes match for each of the selected HTTP requests: 1) CSync, 2) *id-sharing but non-CSync*, 3) other. In this case, *other* refers to HTTP entries containing an ID-looking string, but are *not id-sharing*.

We perform two rounds of tests on one month's data: 1) train and test the algorithm using balanced data from the three classes, in a 10 cross-fold validation process. 2) train the algorithm on balanced data from the three classes (as in (1)), but test it on an unseen and unbalanced dataset which maintains the ratio of the three classes: CSync: 1.6%, *id-sharing but non-CSync*: 0.73% other: 97.67%.

As seen from the classification results (Table 8), the company name and parameter used are among the most important features; number of parameters is the worst. Non-ID related features allow the classifier to reach weighted AUC = 0.973, with a high weighted Precision and Recall across all classes. When all features are used, a weighted AUC = 0.978 is reached, similarly to the high importance feature set that disregards the number of parameters. Interestingly, when the classifier is trained on the balanced dataset, and tested on the unbalanced test set (last row of Table 8), the classifier can distinguish very well the three classes, with low error rates across all three classes, even though there is high imbalance in the classes. These results are further validated by the breakdown of performance per class, demonstrated in Table 9, which show high TP rate and low FP rate for all three classes independently.

Overall, the results show that it is possible to understand and model the patterns of CSync, as they are driven by particular types of companies, using specific parameters, etc. Therefore, an online classifier could be trained to provide insights as to what each HTTP connection is and how likely it is to be performing CSync, without the need to match the IDs to SET cookie actions.

7 RELATED WORK

CSync has become a commonplace on the Web. One of the first works to discuss this mechanism [29] studies programmatic auctions from a privacy perspective and presents CSync as an integral part of communication between the participating entities. The study identified over 100 CSync events while crawling the top 100 sites. In our study, we extend their detection mechanism to detect CSync when cookieID is piggybacked in either URL parameters or path.

In [1], authors conduct a CSync privacy analysis by studying a small dataset of 3000 crawled sites, in conjunction with re-spawning cookies and how, together, they affect the reconstruction of user's browsing history by trackers. In [32] they measure the advertising ecosystem cost to users. Focusing on user privacy and targeted advertising, they use CSync as a metric for anonymity loss, showing that users receive 3.4 CSyncs per ad-impression.

Papadopoulos et al. [33] present how CSync can wreck a secure browsing session. They show cases where 3rd-parties may leak a user's cookie IDs and browsing history, thus increasing the identifiability of the user to a snooping ISP. By probing the top 12k Alexa sites they find 1 out of 13 websites exposing their users to these privacy leaks even when they use TLS and secure VPN services. In a recent census by Englehardt et al. [11], authors measure CSync and its adoption in a small subset of 100,000 crawled sites, before highlighting the need of further investigation given its increased privacy implications. Their results show that 157 of top 200 (i.e. 78%) 3rd parties synchronize cookies with at least one other party.

In [16] they study the economics and the revenue implications of CSync from the point of view of an informed seller of advertising space, uncovering a trade-off between targeting and information leakage. Similarly, in [4], authors explore the role of data providers on the price and allocation of consumer-level information and develop a simple model of data pricing that captures the key trade-offs involved in selling information encoded in 3rd-party cookies. In [13] they investigate tracking groups that share user-specific identifiers in a dataset collected after recording the browsing history of 100 users for two weeks. In this dataset, they detect 660 ID-sharing groups and found domains with sensitive content (such as health-related) that shared IDs with domains related to ad-trackers.

In [2] they aim to enhance the transparency in ad ecosystem with regards to information sharing, developing a content-agnostic methodology to detect client- and server-side flows of information between ad exchanges by leveraging retargeted ads. By using crawled data, authors collected 35448 ad impressions and identified 4 different kinds of information sharing behaviour between ad exchanges. In [3], they study the diffusion of user tracking caused by RTB-based programmatic ad-auctions, considering CSync as the core component of such auctions and the primary factor of the diffusion of privacy leaks. Results of their study show that under specific assumptions, no less than 52 tracking companies can observe at least 91% of an average user's browsing history.

Contrary to these works, we conduct a first of its kind, full-scale study of CSync by analysing a year-long dataset of 850 real users, thus avoiding any biases from crawling websites using artificial personas. Additionally, since nowadays mobile drives the majority of the network traffic, our work is the first to study CSync in the growing ecosystem of mobile devices.

8 SUMMARY AND DISCUSSION

One of the most popular techniques for trackers to share the IDs they assign to users today is Cookie Synchronization, with which different domains can merge their databases in the background. However, syncing userIDs of a given user increases the user identifiability while browsing, thus reducing their overall anonymity on the Web. In this paper, we build CONRAD: a holistic system to detect CSync events, either when the synced IDs are available in plaintext, or even when they are obfuscated (i.e., hashed, encrypted). Using our detection mechanism, we are the first to explore CSync in the mobile ecosystem and the first to analyze it in depth, using a year-long dataset of real mobile users. CONRAD is able to capture 3.771% more CSync cases than related work.

Results: Our analysis of $263k$ CSync requests, syncing $22.3k$ unique userIDs, led to the following findings:

- 97% of users are exposed to CSync at least once. The median user is synced at least once within the first week of browsing.
- Ad-related domains participate in more than 75% of all CSync through the year, learning as much as 90% of all synced userIDs.
- Three companies learn more than 30% of all userIDs, each.
- The median userID gets leaked to 3.5 domains, on average.
- The average user receives around 1 synchronization per 68 HTTP requests, and gets up to 6.5 of their userIDs synced.
- The number of domains that learn about the median user after CSyncs grows by a factor of 6.75.
- We find 63 cases, where domains set cookies on the users' browsers with userIDs previously set by other domains. This universal identification model enables collaborating domains to share data without background database merges.
- We find 131 cases of domains storing in cookies their CSyncs results forming ID Summaries.
- 5% of users suffer from ID-spilling in their secure TLS traffic.
- several sensitive information (e.g., gender, birth dates) is passed to the syncing domain along with the userID.

In addition to the classic, heuristics-based method applied to collect the above findings, in this work, we proposed a novel, CSync detection mechanism able to detect at real time CSync events, even if the synced IDs are obfuscated. In particular, this online, machine learning classifier can be trained to provide insights as to what each HTTP connection is, and how likely it is to be performing CSync, without the need to match the IDs to SET cookie actions. We use the set of detected CSyncs from the heuristics-based method as ground truth, to train our machine learning, cookie-less detection algorithm. We were able to achieve high accuracy (84%-90%) and high AUC (0.89-0.97), when non-ID related features were used.

Countermeasures: Nowadays, the most popular defence mechanism of CSync is the use of the traditional ad-blockers. Indeed, since the vast majority (75%) of CSync takes place among ad-related domains (see Figure 14), it is easy to anticipate that by blocking ad-related requests, one can eliminate a large portion of the privacy leak that CSync causes. However, the all-out approach of ad-blockers causes significant harm on publishers' content monetization models, forcing some of them to deploy anti-adblocking mechanisms [21, 27, 28] and deny serving ad-blocking users [26, 41, 43].

Mitigating mechanisms against CSync require a more targeted blocking strategy, that would not blindly harm the current ad-ecosystem. Instead, by applying detection techniques such as CONRAD, and blocking the specific traffic which has been found to facilitate CSync, we believe that the harmful privacy leakage and loss of anonymity of users due to CSync can be avoided, without the dire consequences on publishers' business models.

Impact: User data collection and sharing activities done without users' explicit consent can be illegal with hefty penalties imposed to companies involved, as described in the new EU regulations for protecting user personal data and online privacy (GDPR and e-Privacy). Thus, it is important to design practical web transparency tools such as CONRAD, readily available to privacy researchers, regulators and end-users to investigate personal data leakage and anonymity on the Web, due to 3rd parties' activities such as CSync [20].

ACKNOWLEDGEMENTS: The research leading to these results has received funding from EU's Marie Sklodowska-Curie and Horizon 2020 Research & Innovation Programme under grant agreements 690972, 786669 and 830927. The paper reflects only the authors' view and the Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. 2014. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA.

[2] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. 2016. Tracing information flows between ad exchanges using retargeted ads. In *Proceedings of the 25th USENIX Security Symposium*.

[3] Muhammad Ahmad Bashir and Christo Wilson. 2018. Diffusion of User Tracking Data in the Online Advertising Ecosystem. *Proceedings on Privacy Enhancing Technologies* 4 (2018), 85–103.

[4] Dirk Bergemann and Alessandro Bonatti. 2015. Selling cookies. *American Economic Journal: Microeconomics* 7, 3 (2015), 259–294.

[5] Juan Miguel Carrascosa, Jakub Mikians, Ruben Cuevas, Vijay Erramilli, and Nikolaos Laoutaris. 2015. I always feel like somebody's watching me: measuring online behavioural advertising. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM.

[6] Tom Chavez. 2010. Data: Deja Vu All Over Again? https://adexchanger.com/considering-digital/tom-chavez/.

[7] Mozilla Developer. 2018. Document.cookie - Web APIs. https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie.

[8] Disconnect. 2019. A faster, safer Internet is one click away. https://disconnect.me/.

[9] Peter Eckersley. 2010. How Unique is Your Web Browser?. In *Proceedings of the 10th International Conference on PETS' 10*.

[10] Jo el van Bergen. 2017. Mixed content weakens HTTPS. https://developers.google.com/web/fundamentals/ security/prevent-mixed-content/what-is-mixed-content.

[11] 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 (CCS '16)*.

[12] European Commission. 2018. What is personal data? https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en.

[13] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. 2016. *Tracking Personal Identifiers Across the Web.*

[14] Famlam Fanboy, MonztA and Khrin. 2018. EasyList - Overview. https://easylist.to/.

[15] Gemfury Dev Center. 2019. HTTPS: Is your URL string secure over SSL? https://gemfury.com/help/url-string-over-https/.

[16] 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 Trans. Econ. Comput. 2015* (2015).

[17] Phillipa Gill, Vijay Erramilli, Augustin Chaintreau, Balachander Krishnamurthy, Konstantina Papagiannaki, and Pablo Rodriguez. 2013. Follow the Money: Understanding Economics of Online Aggregation and Advertising. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference (IMC '13)*.

[18] Roberto Gonzalez, Claudio Soriente, and Nikolaos Laoutaris. 2016. User Profiling in the Time of HTTPS. In *Proceedings of the 2016 ACM SIGCOMM Conference on Internet Measurement Conference (IMC '16)*.

[19] Google Developers. 2018. Cookie Matching. https://developers.google.com/ad-exchange/rtb/cookie-guide.

[20] Costas Iordanou, Georgios Smaragdakis, Ingmar Poese, and Nikolaos Laoutaris. 2018. Tracing Cross Border Web Tracking. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*.

[21] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. 2017. The ad wars: retrospective measurement and analysis of anti-adblock filter lists. In *Proceedings of the 2017 ACM SIGCOMM Conference on Internet Measurement Conference (IMC'17)*.

[22] Jacob Kleinman. 2018. Stop Using WhatsApp If You Care About Your Privacy. https://lifehacker.com/stop-using-whatsapp-if-you-care-about-your-privacy-1825719172.

[23] Paul J Leach, Tim Berners-Lee, Jeffrey C Mogul, Larry Masinter, Roy T Fielding, and James Gettys. 1999. Encoding Sensitive Information in URI's. https://tools.ietf.org/html/rfc2616#section-15.1.3.

[24] Bernard Marr. 2017. Where Can You Buy Big Data? Here Are The Biggest Consumer Data Brokers. https://www.forbes.com/sites/bernardmarr/2017/09/07/where-can-you-buy-big-data-here-are-the-biggest-consumer-data-brokers/.

[25] Jonathan Mayer. 2011. Tracking the Trackers: Microsoft Advertising. *The Center for Internet and Society* (2011).

[26] Brian Morrissey. 2015. Forbes starts blocking ad-block users. https://digiday.com/media/forbes-ad-blocking/.

[27] Muhammad Haris Mughees, Zhiyun Qian, and Zubair Shafiq. 2017. Detecting anti ad-blockers in the wild. *Proceedings on Privacy Enhancing Technologies* 2017, 3 (2017), 130–146.

[28] Rishab Nithyanand, Sheharbano Khattak, Mobin Javed, Narseo Vallina-Rodriguez, Marjan Falahrastegar, Julia E. Powles, Emiliano De Cristofaro, Hamed Haddadi, and Steven J. Murdoch. 2016. Adblocking and Counter Blocking: A Slice of the Arms Race. In *6th USENIX Workshop on Free and Open Communications on the Internet (FOCI 16)*.

[29] Lukasz Olejnik, Minh-Dung Tran, and Claude Castelluccia. 2014. Selling off User Privacy at Auction. In *21st Annual Symposium Network and Distributed System Security (NDSS'14)*.

[30] Kurt Opsahl and Rainey Reitman. 2013. The Disconcerting Details: How Facebook Teams Up With Data Brokers to Show You Targeted Ads. https://www.eff.org/deeplinks/2013/04/disconcerting-details-how-facebook-teams-data-brokers-show-you-targeted-ads.

[31] Elias P. Papadopoulos, Michalis Diamantaris, Panagiotis Papadopoulos, Thanasis Petsas, Sotiris Ioannidis, and Evangelos P. Markatos. 2017. The Long-Standing Privacy Debate: Mobile Websites vs Mobile Apps. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*.

[32] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. 2018. The Cost of Digital Advertisement: Comparing User and Advertiser Views. In *Proceedings of the 27th International Conference on World Wide Web (WWW'18)*.

[33] 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 (EuroSec'18)*.

[34] Panagiotis Papadopoulos, Nicolas Kourtellis, Pablo Rodriguez Rodriguez, and Nikolaos Laoutaris. 2017. If You Are Not Paying for It, You Are the Product: How Much Do Advertisers Pay to Reach You?. In *Proceedings of the 2017 ACM SIGCOMM Conference on Internet Measurement Conference (IMC '17)*.

[35] Andrea Peterson. 2015. Bankrupt RadioShack wants to sell off user data. But the bigger risk is if a Facebook or Google goes bust. https://www.washingtonpost.com/news/the-switch/wp/2015/03/26/bankrupt-radioshack-wants-to-sell-off-user-data-but-the-bigger-risk-is-if-a-facebook-or-google-goes-bust/.

[36] Andrea Peterson. 2015. Bankrupt RadioShack wants to sell off user data. But the bigger risk is if a Facebook or Google goes bust. https://www.washingtonpost.com/news/the-switch/wp/2015/03/26/bankrupt-radioshack-wants-to-sell-off-user-data-but-the-bigger-risk-is-if-a-facebook-or-google-goes-bust/.

[37] Rainey Reitman. 2013. How To Opt Out of Receiving Facebook Ads Based on Your Real-Life Shopping Activity. https://www.eff.org/deeplinks/2013/02/howto-opt-out-databrokers-showing-your-targeted-advertisements-facebook.

[38] Matt Richtel. 2000. F.T.C. Moves to Halt Sale Of Database at Toysmart. http://www.nytimes.com/2000/07/11/business/ftc-moves-to-halt-sale-of-database-at-toysmart.html.

[39] samy.pl. 2014. Evercookie - virtually irrevocable persistent cookies. https://samy.pl/evercookie/.

[40] Judy Selby. 2016. The Impact of Big Data Decisions on Business Valuations. https://datafloq.com/read/impact-big-data-decisions-business-valuation.

[41] Nicola Smith. 2016. How publishers are turning up the heat in the ad-blocking war. https://www.theguardian.com/media-network/2016/sep/02/publishers-ad-block-users-hide-content.

[42] Statista Inc. 2018. Percentage of all global web pages served to mobile phones from 2009 to 2018. https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share.

[43] The Editors of Wired. 2016. How WIRED Is Going to Handle Ad Blocking. https://www.wired.com/how-wired-is-going-to-handle-ad-blocking/.

[44] Narseo Vallina-Rodriguez, Srikanth Sundaresan, Abbas Razaghpanah, Rishab Nithyanand, Mark Allman, Christian Kreibich, and Phillipa Gill. 2016. Tracking the trackers: Towards understanding the mobile advertising and tracking ecosystem. *arXiv preprint arXiv:1609.07190* (2016).

[45] World Wide Web Consortium (W3C). 2010. Same Origin Policy. https://www.w3.org/Security/wiki/Same_Origin_Policy.

[46] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M. Pujol. 2016. Tracking the Trackers. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*.