Computer Science Department

Analyzing Data Gathering and Sharing Online

Report for Qualifying Exam

by
Maaz Bin Musa
maazbin-musa@uiowa.edu

Advisor
Rishab Nithyanand
rishab-nithyanand@uiowa.edu

1 Abstract

Online tracking has resulted in user data becoming the currency of the internet. Research has shown user data propagates from users to trackers via two types of data flows. When user data is collected by tracking entities directly on the website, it is called **first degree** data flow. When this data is shared from the website to external entities and beyond, it is called **second degree** data flow. Research on first degree data flows has identified tracker penetration, persistent cookies, and browser fingerprinting as popular techniques used by trackers to collect user data. Unlike first degree data flows, second degree data flows occur outside the website, making them comparatively difficult to analyze. Transparency of these data flows is important for regulatory compliance. GDPR in Europe and CCPA in the USA are two such regulatory compliances, that regulate the usage of user data. In the case of data misuse, these regulatory compliances hold unscrupulous entities accountable. Hence, understanding data gathering and sharing flows is an important problem which the following papers work towards solving:

- Steven Englehardt and Arvind Narayanan. Online Tracking: A 1-million-site Measurement and Analysis. ACM CCS. 2016.
- Shehroze Farooqi , Maaz Musa, Zubair Shafiq and Fareed Zaffar. CanaryTrap: Detecting Data Misuse by Third-Party Apps on Online Social Networks. Proceedings on Privacy Enhancing Technologies. 2020.
- John Cook, Rishab Nithyanand, and Zubair Shafiq. Inferring Tracker-Advertiser Relationships in the Online Advertising Ecosystem using Header Bidding. Proceedings on Privacy Enhancing Technologies. 2020.

2 Introduction

Monetization of user data via tracking has become the driving force of the internet. Research work has been able to study how trackers use various techniques to collect and share user data. These data flows exist on the websites as well as outside the website. Data flows that occur directly on the website are called **first degree** data flows. Whereas data flows taking place between the website and external entities, or between two external entities are called **second degree** data flows. Browser limitations and server-side data sharing make analyzing these data flow a challenging task. Not being able to analyze these data flows exempts the entities involved from complying with regulatory authorities such as GDPR [6] or CCPA [3]. These regulatory authorities are in place to monitor how user data is consumed.

Considering the importance of understanding user data flows, this report summarizes the research that deals with analyzing these flows. The methodologies highlighted by this report help identify entities that are involved in these data flows and it highlights the tools and techniques required to carry out these measurements effectively on a large scale.

How is data gathered? To analyze data flows, first, we must look at how user data is gathered. Fig. 1 shows how trackers can gather user data through various techniques on a website (facilitator). As per earlier definition, gathering data directly on the website is categorized as first degree data flow. Research work on first degree data flows [22, 23] has

highlighted the pervasiveness of trackers and associated privacy implications caused by online tracking. In addition to trackers, research has also identified new and evolved tracking techniques involved in first degree data flows, such as browser fingerprinting [2, 29, 26, 24, 11] and persistent cookies [7]. Further research [25, 13, 11, 29] shows the increasing popularity of these techniques in the community. Research has also been able to produce tools such as WebXray [25] and FPDetective [13] to carry out measurements about these first degree data flows.

Websites use user data to study website analytics to improve their performance (Fig. 1). More importantly, user data is gathered to identify individual users and their activity. This helps consumers offer more personalized services e.g targeted advertisements, shopping suggestions, or movie recommendations.

How is data shared? As Fig. 1 shows, user data can be shared with consumers outside the website. These data flows are defined as **second degree** data flows. Research [31, 33, 16] has shown the value of an online user depends on the amount of data it shares with consumers. This instigates the consumers to gather as much data as they can. Acar et al. and others [12, 14, 31] show prevalence of client-side data sharing via cookie syncing [9]. In addition to client-side data sharing, consumers can share data on the server-side, with the user having no knowledge of it. These server-side data sharing flows can exist in the form of a simple purchase of data, data leakage, or even exchange between two entities belonging to the same organization. This motivates the analysis of these second degree data flows. Bashir et al. [15] introduces a technique the uncovers several cases of client and server-side data sharing via cookie syncing and DeBlasio et al. [17] analyzes password reuse attacks originating from server-side data sharing.

Technical challenges. With the changing online ecosystem, analyzing these data flows has proven to be a challenge. First, browsers by nature weren't made for automation. In addition to this, there aren't many stable libraries that automate the browser. Combined, these things make it difficult to measure data flows of both first and second degree on a large scale. Second, to show results that represent the real world, we must be able to replicate the browsing experience in detail. Most of the tools do not allow detailed browser automation and instrumentation. Finally, as server-side second degree data flows take place at the backend, they are not directly visible for measurement and analysis.

The three papers we discuss in this report help overcome these limitations. All together they produce a robust tool that can perform large scale measurements to analyze data gathering and sharing. They also introduce methodologies that can help analyze specific server-side second degree data flows.

3 Background

In this section, we will get to know about tracking techniques used for gathering and sharing user data. Trackers have been expanding and improving techniques used in gathering user data. Whereas, there is only one client-side data sharing technique. Server-side data sharing does not even have a fixed technique which is why there is limited work on them.

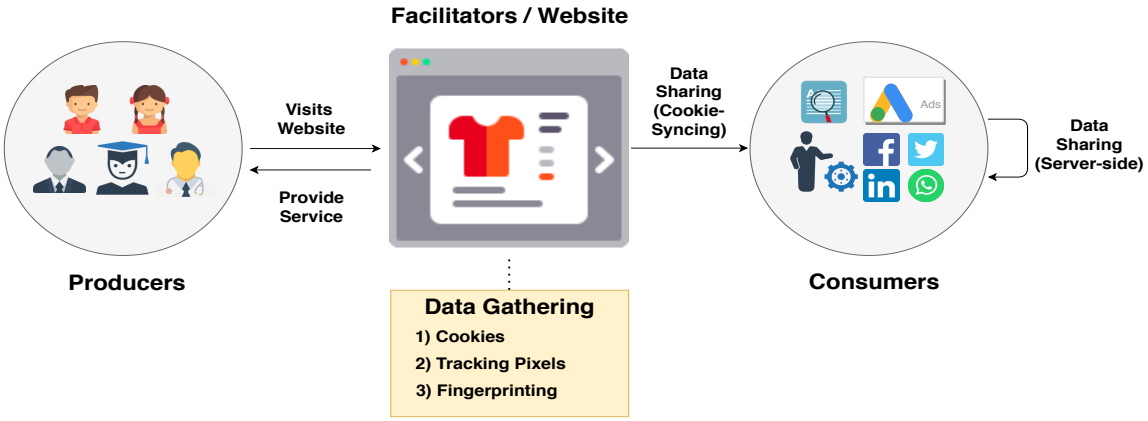

Figure 1: All participants in the online data flow ecosystem

3.1 Data Gathering Techniques

Research has identified various tracking techniques that help in gathering user data. We will now discuss three of these tracking techniques used by data gatherers:

3.1.1 Cookies

Cookies [8] were introduced to record a users online activity and provide better user experience in return. A website contains its cookie for each user and all the cookies of third parties integrated on the website. All of these cookies can observe users browsing activities and interactions. However, one entity can not access the cookies of another entity.

3.1.2 Tracking pixel

Tracking pixels operate similar to a cookie in the sense that it is present on websites and records users activity [21]. However, these pixels are 1x1 images inserted on the website with the only purpose of gathering user data.

3.1.3 Fingerprinting

Browser fingerprinting is a persistent tracking technique [27] that collects a different kind of user data as compared to the previous two techniques. Fingerprinting is achieved by collecting user data such as location, browser type, screen resolution, font types, etc. These data-points are aggregated together to create a fingerprint, unique to each user. Research has shown several occurrences of browser fingerprinting in the real world using HTML canvas [28], canvas font [18] and battery API [30].

3.2 Data Sharing Techniques

Once user data is collected by trackers, it can be saved on their backend servers or it can be shared with other entities. Research has shown there are two types of data sharings: 1) Client-Side: trackers share user data with consumers via cookie syncing 2) Server-Side: Consumers can share this data with other consumers further.

3.2.1 Client-side Data Sharing

Since cookies are domain specific, reading someone else's cookie is impossible. Cookie syncing/matching [9] is a technique that circumvents this limitation and allows two entities to share data. For example, *Consumer C1* identifies a user as *U1* and *Consumer C2* identifies the same user as *U2*. *C1* and *C2* exchange their identifiers for this user which allows them to share data about that user in the future. In addition to gathering user data via tracking on the website, consumers extend their knowledge of users by using cookie syncing [12, 14, 31].

3.2.2 Server-side Data Sharing

Other than cookie syncing, all data sharing takes place at server-side, invisible to the user. To our knowledge, only one other previous work analyzes backend data sharing [17]. Tripwire [17] registers accounts on 2300 websites with a unique email address and password each. They then monitor possible login attempts on the email accounts which share the same email address and password as the one used for account registration. This helps them identify data breaches on specific websites, which caused unintentional data sharing of users' email and password with adversaries. In the two years they monitored the email accounts, they found 19 site compromises. They conclude that most of these breaches were on small, understaffed websites. However, Tripwire was able to found some large scale data breaches that were undisclosed to the clients.

4 First Degree Data Flows

In this section, we will talk about the steps required in analyzing first degree data flows including browser automation and instrumentation. We will go on to discuss how a tool (OpenWPM) can perform large scale analysis on first degree data flows and extrapolate interesting insights.

4.1 Re-creating first degree data flows

Browser automation. As defined in §2 gathering user data is a first degree data flow. The first aspect of analyzing first degree data flows is simulating user interactions with browsers. Interacting with the browser here refers to browsing a website. Manually simulating these interactions is time consuming, does not scale well, and is expensive if outsourced. To overcome this problem researchers automate browser interactions through tools such as Selenium and scripting languages such as Python. This task is easier said than done since browsers were never made to be automated. Furthermore, the instability of tools such as Selenium aside, the key issue with simulating user interactions is maintaining users' state. Stateful measurements refer to maintaining user cookies between page visits as opposed to stateless measurements where every page visit looks like a new user to data gatherers. Stateful crawls are significant as they represent real users more accurately. They are also important because measuring data flows such as cookie syncing is only possible if the user has cookies while interacting with the browser.

Browser instrumentation. The second aspect of analyzing first degree data flows is observing all interactions. Most tools that analyze first degree data flows are unable to

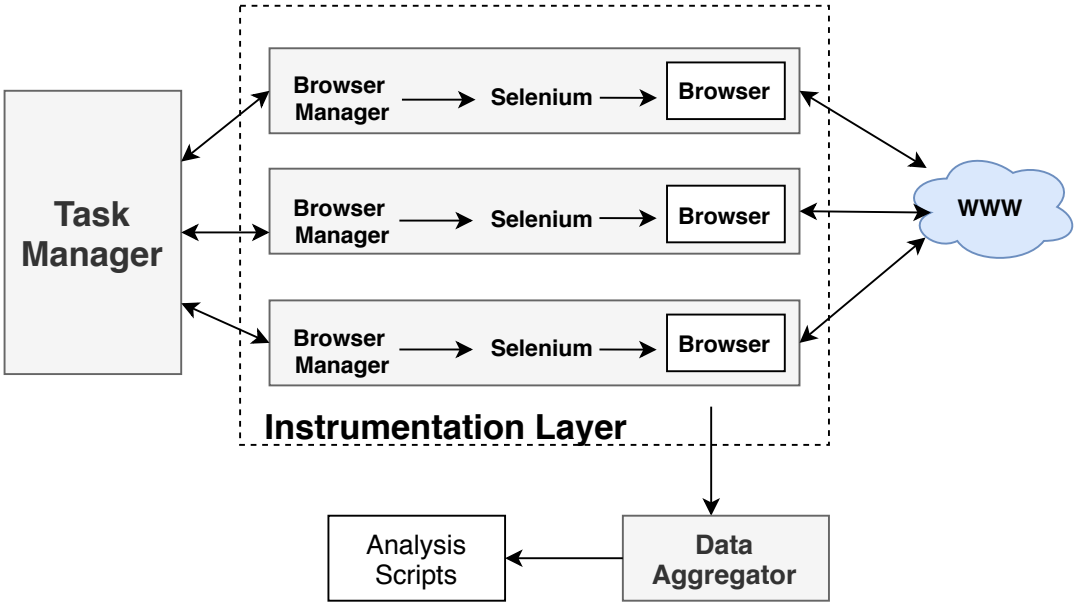

Figure 2: This figure shows how the three main modules of OpenWPM operate

monitor all of these interactions. As a result, some of these interactions are not being analyzed. Being able to instrument the browser from various vantage points allows observing more interactions, and hence, gives a more representative and complete picture of the real world.

OpenWPM. Englehardt et al. [19] introduces a tool called OpenWPM that can perform analysis of first degree data flows on a large scale. Not only does OpenWPM offer stateful crawls but it also provides an interface to add browser extensions such as adblockers. Unlike other tools, OpenWPM is able to observe browser interactions from three points including network data, javascript data and disk data. They achieves these goals by dividing OpenWPM in three major modules:

4.1.1 Task Manager

As shown in Fig. 2 the outermost module that interacts with humans, is the task manager. This module allows OpenWPM to control multiple browsers simultaneously. Different commands can be issued to different browsers via the command line interface provided by the task manager.

4.1.2 Browser Manager

Browser manager is how OpenWPM takes human commands from the task manager and translates them into Selenium commands to automate the browser. Each browser manager has its own isolated process to separate one threads failure from stopping the entire measurement. The browser manager is also responsible for maintaining the state of the browser which allows OpenWPM to conduct stateful measurements.

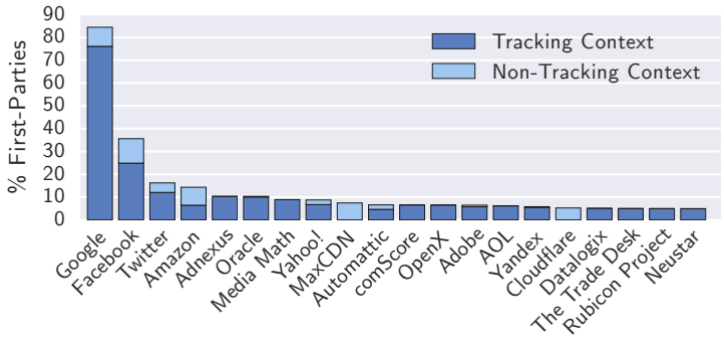

Figure 3: Organizations with the highest third party presence on the top 1 million sites. Not all third parties are classified as trackers, and in fact the same third party can be classified differently depending on the domain appearing in the tracking-protection list

4.1.3 Data Aggregator

Finally, this module receives data from all browser managers and logs it in one central database after processing it. This module is accessed through a socket interface and can be accessed through any number of browser managers. Since each browser manager has its browser id and page visit details, this module aggregates new data from each browser manager with its previous data with ease. The data received by this module comes from three main areas: (1) HTTP data: to record all HTTP requests and responses; (2) Disk data: to monitor creating, or accessing cookies; and (3) Javascript access data: to monitor access to javascript objects used for browser fingerprinting.

4.2 Million Site Crawl

OpenWPM was used to analyze Alexa's [1] top 1 million sites for first degree data flows. The homepage of each website was visited and all data flows that resulted in this interaction were stored. For the 100,000 stateful crawls, they used a seed profile that was trained on the top 10,000 sites. Analyzing data flows from this large scale measurement leads to some interesting insights.

4.2.1 Tracker Prevelance

Every third party is considered a potential tracker. But to further solidify the nature of third party as a tracker, OpenWPM uses 2 tracking-protection lists, EasyList [5] and EasyPrivacy [4]. All third party domains present in these lists are considered as trackers. Analyzing third party data collected, Englehardt et al. were able to show that there is a long but thin tail of trackers present on the top 1 million sites. The tail as shown in Fig. 3 is considered long because most of the trackers cover at least 10% of the top million sites. On the other hand, the tail is thin because there are only a few trackers that have high prevalence (40% - 70%).

This insight shows that only a handful of large entities have high coverage of trackers on the top million sites. This suggests that regulating just a few entities will have huge consequences on the tracking ecosystem.

4.2.2 Fingerprinting

OpenWPM records all interactions with the browser, including scripts used for fingerprinting. OpenWPm defines custom getters and setters to record access to certain objects in the browser that can be used for fingerprinting. This methodology allows the monitoring of newer suspected scripts by just adding a few lines of code. In their measurements, they show increased usage of previously identified fingerprinting scripts. On top of this, they measured two fingerprinting methodologies that were previously never measured on a large scale.

1) Canvas and canvas font fingerprinting. HTML canvas allows websites to draw graphics on websites in real time. Accessing the pixels or fonts that make these graphics on the canvas can be very useful in fingerprinting a user. How a pixel is drawn on the canvas or more specifically, how different fonts draw a pixel on the canvas can fingerprint a user. OpenWPM shows that canvas and canvas font fingerprinting has gained considerable popularity as compared to previous studies (7 folds for canvas font). For canvas font, it shows that it exists on less than 1% of the top million sites but it is present on 2.5% of the top 1000 sites.

2) WebRTC fingerprinting. WebRTC is a technology accessible via javascript, that enables peer-to-peer real time communication online. OpenWPM shows that WebRTC is being used to extract a user's IP without any need. Gathering a user's IP is a clear cut way to fingerprint a user. Tracking via WebRTC was found in 625 sites out of 1 million.

3) AudioContext and battery API fingerprinting. Using the insight that fingerprinting usually takes place in conjunction with each other, this analysis introduces a new semi-automated way to find fingerprinting scripts. They examine already fingerprinting labeled scripts and find out which other scripts are most often used with these scripts. This heuristic helps them identify audiocontext and battery API scripts as fingerprinting scripts. Several fingerprinting scripts only check the status of audiocontext interface. Other more complex scripts process an audio signal using this audiocontext interface to fingerprint a user. Furthermore, gathering battery discharge patterns of users device is known to have the potential to fingerprint users [30]. This measurement concludes that both of these are infrequent ways of fingerprinting a user.

4.3 Takeaways

OpenWPM has provided a tool that makes measuring first degree data flows on large scale possible. It highlights the prevalence of trackers and fingerprinting scripts on the top 1 million sites. It also introduces new ways to uncover fingerprinting scripts. Finally, the stability, robustness, and completeness of this tool sets the stage for researchers to analyze second degree data flows using OpenWPM.

5 Second Degree Data Flows

Client-side data sharing via cookie syncing has been extensively analyzed and its prevalence is well explored. In this section, we focus on server-side data sharings between consumers and how they are analyzed.

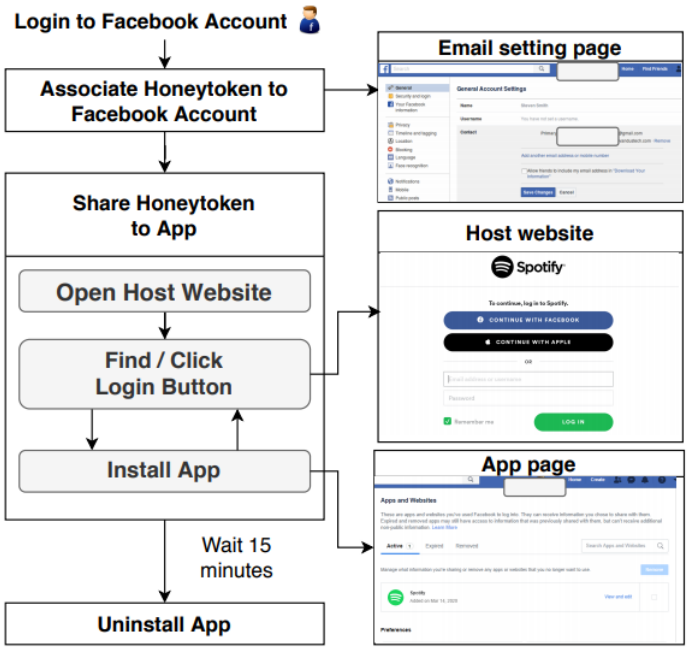

Figure 4: Automated workflow of our CanaryTrap implementation to associate a Facebook account to honeytokens that are then shared with applications

5.1 Honey Traps

Inspired by Tripwire [17], Farooqi et al. [20] introduces Canarytrap, a technique that leverages email address (honeytoken) of a user to monitor server-side data sharing. Canarytrap aims to find cases of data misuse caused by unauthorized server-side data sharing. Canarytrap also introduces a more complex implementation of its framework that can help scale this technique exponentially.

Overview. Canarytrap is motivated by the increasing amount of trackers that gather user data from social media and share it forward without proper authorizations. Canarytrap is deployed using 1024 applications that can be installed on Facebook. These applications are monitored for a year, via an email server set up by the authors. As a result, authors were able to find 16 cases of server-side data sharing that resulted in ransomware attempts, spam, and targeted advertisements.

5.1.1 Array Framework

Selecting honeytoken. Canarytrap depends on leaking user data (honeytoken) to applications that cause server-side data sharing. According to the authors, the user data that can be used in Canarytrap has to: 1) have enough value to instigate *exploitation*; 2) allow *sound* monitoring channels for misuse detection; 3) be *feasible* in allowing these monitoring channels to scale up; and 4) be easily *available* in the account information for the applications to acquire it. Keeping these 4 properties in mind, the authors choose email address as the appropriate honeytoken for the experiment. An email address is a unique identifier for a user which makes it *exploitable*. It allows *sound* monitoring channel in the form of emails received which can be scaled *feasibly* by setting up an email server. Finally, it is highly

available in the account information which is shared with applications.

Leaking honeytoken. To deploy Canarytrap, the authors select 1024 applications that can be installed on Facebook. They assign each application a honeytoken (email address). The set of honeytokens is created by randomly combining a first and last name and adding the authors' email domain at the end e.g firstnamelastname@mydomain.com. The 1024 applications are processed like an array. To start, a honeytoken is added as the primary email address to a Facebook account the authors created earlier Fig. 4. As primary email is shared with applications this confirms leakage of honeytoken. Once the honeytoken is set, they open the applications host website to install the application. After locating the "Login via Facebook" button on the website, they install the application. Once the application is installed, and the honeytoken is leaked to the application, they remove the application and the honeytoken from the Facebook account. They repeat the process for all applications on the same Facebook account. Successful execution of this experiment means each of these applications now knows exactly 1 honeytoken that is known to no one else.

Data misuse attribution. After all the honeytokens are shared with the associated applications they start monitoring their email server. All received emails are checked for data misuse. If an email is addressed to a known honeytoken, Canarytrap checks if the associated application is the sender or not. To match a honeytoken with an application they generate keywords using the application name and its host website's domain name. They search the keywords in various parts of the received email. If any of the keywords is found in the email, they label the email as recognized, otherwise this suggests the sender is someone who the application shared the user data with.

Results. Canarytrap is able to identify 16 cases of data misuse that were caused by applications sharing honeytokens with external entities. As seen in Table 1, the 16 applications are spread across the globe, with rank as high as 870 in the world. Out of these 16 cases, 12 are of varying extremes of data misuse. 3 out of these 12 applications send emails containing ransomware, scam, or spam. After some manual inspections, the authors classified the remaining 4 applications as authorized cases of data sharing.

5.1.2 Matrix Framework

Honeytoken reuse. Canarytrap also implements a version of its framework called matrix framework, which uses exponentially fewer honeytokens to monitor the same number of applications. This framework takes two sets of honeytokens and creates a relationship between them that allows the monitoring of more applications. In this framework instead of one honeytoken being mapped to one application, a pair of two honeytokens is mapped to one application. This allows them to reuse one honeytoken as long as it is in a different pair next time. As shown in Fig. 5 an application a_{mn} is installed on a unique pair of honeytokens r_m and c_n. As a result, they can monitor $n*m$ applications using $n+m$ honeytoken as compared to the array framework, where they could only monitor $n+m$ applications.

Data misuse attribution. Monitoring data misuse in this version is slightly more complex since one honeytoken is shared with multiple applications. If a honeytoken receives an email that belongs to none of the associated applications, more evidence is required to find the culprit application. This is only possible when both honeytokens associated with an ap-

App Name	Host Website	Global Alexa Rank	Country Alexa Rank
Safexbikes Motor cycle Superstore	safexbikes .com	89K	8K (IN)
WeWanted	wewanted.com.tw	99K	-
Printi BR API	printi.com.br	15K	441(BR)
JustFashionNow	justfashionnow.com	51K	223 (MO)
PopJulia	popjulia.com	469K	
MyJapanBox	myjapanbox.com	766K	-
Nyx CA	nyxcosmetics.ca	258K	16K (CA)
Tom's Hardware Guide-IT Pro	tomshardware .com	870	726 (USA)
Alex's first app	beautymaker.com.sg	680K	3K (SG)
Thailand Pro-perty Login	thailand-pr-operty.com	98K	3K (TH)
Hop-On, Hop-Off	hop-on-hop -off-bus.com	161K	77K (USA)
Leiturinha	leiturinha.com.br	114K	5K (BR)
The Breast Expa nsion Story Club	bestoryclub .com	484K	-
Jacky's Electronics	jackyselectronics .com	517K	8K (UAE)
Berrykitchen.com	Berrykitchen.com	494K	-
uCoz.es Login	ucoz.es	157K	-

Table 1: Popularity of 16 Facebook apps that are responsible for the misuse of the shared honeytokens

plication are shared with the same entity by that application. If that entity sends an email to both honeytokens, Canaraytrap can catch the application that shared the honeytokens.

Evaluation of matrix framework. Using array framework as ground truth, the au-

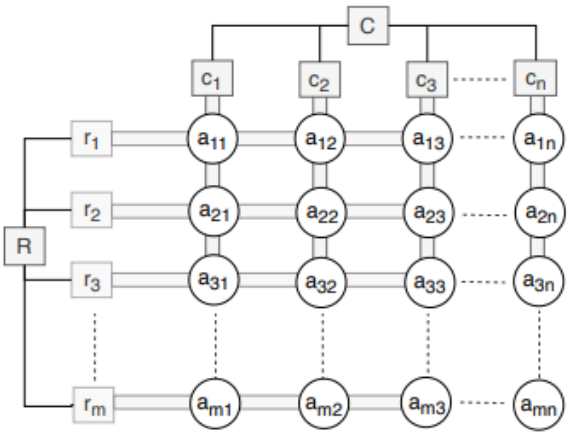

Figure 5: Illustration of the matrix arrangement used by Canarytrap. Two sets of honeytokens (r1, ..., rm and c1, ..., cn) are associated with two Facebook accounts (R [row] and C [column]). While each honeytoken is shared with multiple apps in the corresponding row or column, an app is shared two honeytokens (one from row and one from column) to create a unique two-dimensional mapping

thors evaluate their matrix framework. For this, they run the matrix framework on the same 1024 applications they used in the array framework. If an application was involved in data sharing in the array framework, it should be caught in the matrix framework as well. Otherwise, this reduces the reliability of the matrix framework. Considering the array framework as ground truth, they define the following: (1) true positive is an application that was caught sharing data, by both frameworks; (2) true negative is an application that was considered benign by both frameworks; (3) false negative is an application not caught by the matrix framework but was caught by the array framework; and (4) false positive is an application that was caught by matrix framework, but not by the array framework. Out of the 1024 applications, matrix framework produced 9 true positives and 1008 true negatives. This indicates that the matrix framework is able to perform relatively well as it catches most of the applications which the array framework did. The matrix framework had 0 false positives which is also a good sign, as this implies no application was falsely accused. The remaining 7 applications are labeled as false negatives due to implementation issues of the experiment, or un-deterministic behavior of the application. Overall this shows that the matrix framework can be scaled easily, but it will decrease the accuracy of the experiment.

5.1.3 Privacy Policy Compliance

Canarytrap goes one step further and analyzes the compliance level of some of these applications when it comes to user data. They run a second small experiment, where they contact a sample of 100 applications and request data deletion. They then manually keep a track of how many of these applications respond and delete user data. Out of the 100 applications, 13 did not even provide a way to contact them. From the remaining 87 applications, 42 never responded to the request. Only 29 applications acknowledged data deletion and 13 of these 29 applications kept using the user data even after acknowledging data deletion.

5.1.4 Takeaways

Canarytrap demonstrated the possibility of uncovering second degree data sharing flows on social media networks, in this case, Facebook. They were able to show that this data sharing can result in some very serious data misuse cases. They also introduce a version of Canarytrap that can easily scale exponentially. Finally, they show that only a few applications on social media follow the privacy policy regulations, which emphasizes the importance of analyzing these data flows.

5.2 Inferring Data Sharing

In several cases, it is not possible to create one to one mappings of user data with consumers. In such cases, techniques such as Canarytrap [20] and tripwire [17] fail to uncover second degree data flows. Cook et al. [16] introduces KASHF, a tool that leverages artifacts observable on the client-side to uncover client-side and server-side data sharing flows between advertisers (a type of consumer) and trackers.

5.2.1 HeaderBidding

Headerbidding [32] is a fairly new way for online advertisement. One of the main differences between headerbidding and other online ad delivery systems is that it allows users to view all participating ads and their bidding details on the client side. How much an advertiser is bidding on a user can represent users' value for that advertiser. The authors use this bid value as the artifact that helps them uncover data sharing flows.

5.2.2 Quantifying User Value

With headerbidding, the authors can see all participating bidders. This enables them to analyze the bidding behavior of all participating advertisers. For this, they use OpenWPM to train profiles/personas and collect ads and their bidding details.

Persona training. KASHF uses a standard methodology of training personas, by visiting websites and storing the cookies. For this experiment, they select 16 categories from Alexa and train each persona on the top 50 sites from each category. They further train 16 personas that visit an extra site called "Intent Site". Intent sites have previously shown to send a stronger signal to advertisers [31]. Finally, they create a 33rd persona as control, with no browsing history.

Gathering bids. To collect headerbidding ads, the authors first assembled a list of sites from the top 10K Alexa sites, that enable headerbidding. They are able to find 25 sites that allow headerbidding. The resultant 25 sites are visited by each persona 10 times and the ads and ad details they received are stored.

Results. Analyzing the bid values, they show that the type of persona and presence of intent impacts user value. Their results also show certain bidders are inclined towards specific types of personas. They also show that all bidders show increased user value in the presence of intent site, but they all vary in the amount of increase. To summarize, they show that a user's value to an advertiser is highly dependant on the user data available to the advertiser.

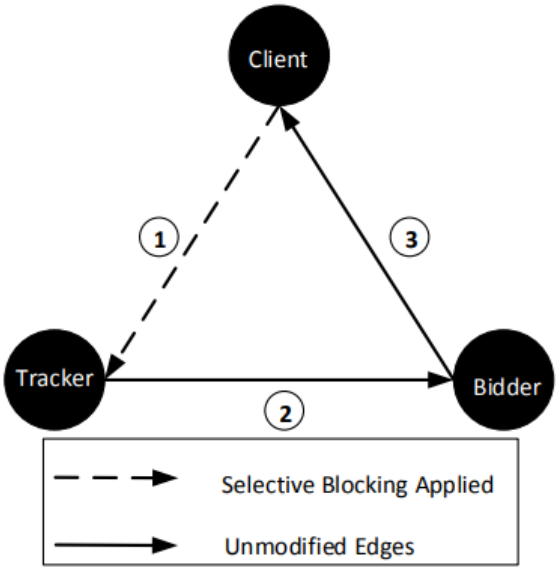

Figure 6: A simplified model of the information flows between trackers and bidders

5.2.3 Inferring Tracker Advertiser Relations

Overview. The idea behind KASHF is based on the insight that advertisers value known users more than new users. The first experiment solidifies this insight. Keeping this in mind they expose user data to a selective set of trackers Fig. 6 (1) and record bidding behavior of advertisers (3). Next, they train interpretable machine learning models with trackers as features and predict the bid value. If these models have high accuracy, their features can be analyzed to infer relationships between advertisers (bidders) and trackers (2).

Persona training. The process of training personas is mostly similar to their previous experiment. They select a persona category randomly and train it on 10 sites rather than 50. They also randomly choose to add intent site or not. The main difference in this experiment is that while training each persona they block all trackers from 1 tracking organization from a list of 20. They repeat the process 10,000 times and train 10,000 personas. The selective blocking of trackers allows them to vary how much data some of these trackers can observe.

Gathering bids. After training the persona, they visit one of the 25 header bidding sites identified in the previous experiment and gather ads. They restrict ad collection to just 1 site to make sure all tracker-advertiser data flows are generated by persona training and not ad collection.

Interpretable machine learning models. Before creating models, they assign the continuous bids they collected a discrete bid class. Next, they train interpretable models for each advertiser with exposed trackers as features, intending to predict the bid class. Each model with high accuracy can be analyzed further to find impactful features. The high impact of a feature (tracker) can be inferred as a relationship between that feature (tracker) and the advertiser.

Results. The models trained for the top 5 most common advertisers resulted in an average

accuracy of 80%. Previous research work acheived an accuracy of 82% in predicting bid values that were encrypted [33]. Comparing the accuracy of these models with previous research models validates the accuracy levels of KASHF. Next, they generate decision trees from each model and the feature (tracker) at the top of this tree is the most impactful. Table 2 shows top 3 trackers for top 5 bidders from the experiment. These are the 15 server-side data sharing relationships this paper highlights. 4 of these 15 relationships are validated as they are involved in cookie syncing with each other. The remaining 11 are server-side second degree data flows that are not observable or disclosed.

Bidder	Tracker 1	Tracker 2	Tracker 3
AppNexus	DoubleVerify	Automattic	Comscore
IX	Sovrn	PubMatic	DoubleVerify
OpenX	Microsoft	AppNexus	Criteo
Rubicon	Verizon	DoubleVerify	Facebook
PubMatic	Alphabet	Twitter	Microsoft

Table 2: Trackers are ranked in the descending order of information gain for each of the top-5 bidders

5.2.4 Takeaways

KASHF shows the value of the user is highly impacted by the data it shares about themselves. They also show how to use this value to observe server-side second degree data flows. They uncover data sharing flows that were not possible to identify by just analyzing client-side data flows.

6 Discussion

Online users are producing data at rates never seen before. This data has become a commodity as consumers use the data to make money. To introduce some checks and balances, data protection policies such as GDPR and CCPA have started to emerge. Veritifying compilance of such policies motivates the analysis of data flows in the online ecosystem. This is done in two phases. The first phase is to analyze how data is gathered and the second phase is to analyze how this data is shared.

OpenWPM [19] is a tool that has high modularity, robustness, and a wide range of features. Using this tool to analyze the top million sites gives us a good picture of the data gathering flows. It shows the prevalence of first degree data flows via trackers and browser fingerprinting. It sets the stage for similar measurements that can be used to analyze data flows.

There is no way to know all second degree data sharing flows since it happens mainly on the server-side and is not observable. However, with the help of some insights and client-side information we can analyze specific second degree data flows on the server-side. Canarytrap analyzes one such server-side data sharing flow, that originates from

applications on Facebook and results in user data misuse. Their methodology is applicable to most of the social media platforms with minor coding changes.

Data sharing gets difficult to measure as it gets more complex. For more complex data sharing flows KASHF uses client-side information and interpretable machine learning models to infer these relationships. With this, they are able to identify several tracker-advertiser data sharing flows that were undisclosed.

Network tomography. Analyzing server-side data sharing lacks a standard framework. Canarytrap is specific to one kind of data sharing flow. However, the approach adopted by [16] can be generalized into a framework that shares the same principles as network tomography. Network tomography [10] is a way to infer internal behavior and topology of a network, based on end to end measurements. Mapping this problem to server-side data sharing, we need to have two ends that we can measure which are connected in the middle by some data sharing. On top of this, we need a way to manipulate this data sharing from the outside. In [16], the two ends can be considered as exposing user data to trackers and receiving ads, whereas selective tracker blocking is the manipulation. Keeping this in mind new kinds of server-side data sharings can be explored.

Server-side headerbidding. KASHF [16] uses headerbidding on the client-side to uncover server-side data sharing. However, this technique will fail in the absence of client-side headerbidding. It is important to look for client-side artifacts that are agnostic of ad delivery system. One such artifact can be the ad image itself. In all forms of ad delivery systems, the ads image will always be visible on the client-side. As a future direction, extracting quantifiable information from these ad images can help replace the bid value, in inferring server-side data sharing relationships.

7 Conclusion

In this report, we looked at how to measure first and second degree data flows. We looked at research that provides stable tools to perform these measurements. We go on to study work that uses these tools or their own novice techniques to identify data flows on client-side and server-side. Finally, we discussed how to possibly extrapolate a generic framework, motivated by [16] that can help in measuring server-side data flows.

References

[1] Alexa - keyword research, competitive analysis, & website ranking. https://www.alexa.com/. (Accessed on 08/30/2020).

[2] Browser fingerprinting: What is it and what should you do about it? https://pixelprivacy.com/resources/browser-fingerprinting/. (Accessed on 08/30/2020).

[3] California consumer privacy act (ccpa) — state of california - department of justice - office of the attorney general. https://oag.ca.gov/privacy/ccpa. (Accessed on 08/31/2020).

[4] Easylist - easyprivacy. https://easylist.to/tag/easyprivacy.html. (Accessed on 08/31/2020).

[5] Easylist - overview. https://easylist.to/. (Accessed on 08/31/2020).

[6] General data protection regulation (gdpr) compliance guidelines. https://gdpr.eu/. (Accessed on 08/30/2020).

[7] Samy kamkar - evercookie - virtually irrevocable persistent cookies. https://samy.pl/evercookie/. (Accessed on 08/30/2020).

[8] What are computer cookies? https://us.norton.com/internetsecurity-privacy-what-are-cookies.html. (Accessed on 08/30/2020).

[9] What is cookie syncing and how does it work? - clearcode blog. https://clearcode.cc/blog/cookie-syncing/. (Accessed on 08/30/2020).

[10] What is network tomography? — netbeez. https://netbeez.net/blog/network-tomography/#:~:text=Network%20tomography%20is,reach%20conclusions%20about%20its%20guts. (Accessed on 08/31/2020).

[11] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., and Diaz, C. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security* (2014).

[12] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., and Diaz, C. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security* (2014).

[13] Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gurses, S., Piessens, F., and Preneel, B. Fpdetective: Dusting the web for fingerprinters. In *Proceedings of ACM SIGSAC conference on Computer & communications security* (2013).

[14] Bashir, M. A., Arshad, S., Robertson, W., and Wilson, C. Tracing information flows between ad exchanges using retargeted ads. In *25th {USENIX} Security Symposium ({USENIX} Security 16)* (2016).

[15] Bashir, M. A., and Wilson, C. Diffusion of User Tracking Data in the Online Advertising Ecosystem. In *Proceedings on Privacy Enhancing Technologies* (2018).

[16] Cook, J., Nithyanand, R., and Shafiq, Z. Inferring tracker-advertiser relationships in the online advertising ecosystem using header bidding. In *Proceedings on Privacy Enhancing Technologies* (2020).

[17] DeBlasio, J., Savage, S., Voelker, G. M., and Snoeren, A. C. Tripwire: Inferring internet site compromise. In *Proceedings of Internet Measurement Conference* (2017).

[18] Eckersley, P. How unique is your web browser? In *International Symposium on Privacy Enhancing Technologies Symposium* (2010).

[19] Englehardt, S., and Narayanan, A. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security* (2016).

[20] Farooqi, S., Musa, M., Shafiq, Z., and Zaffar, F. Canarytrap: Detecting data misuse by third-party apps on online social networks. In *arXiv* (2020).

[21] Fouad, I., Bielova, N., Legout, A., and Sarafijanovic-Djukic, N. Tracking the pixels: Detecting web trackers via analyzing invisible pixels. In *arXiv* (2018).

[22] Gill, P., Erramilli, V., Chaintreau, A., Krishnamurthy, B., Papagiannaki, K., and Rodriguez, P. Follow the money: Understanding economics of online aggregation and advertising. In *Internet measurement conference* (2013).

[23] Jain, S., Javed, M., and Paxson, V. Towards mining latent client identifiers from network traffic. In *Proceedings on Privacy Enhancing Technologies* (2016).

[24] Kohno, T., Broido, A., and Claffy, K. C. Remote physical device fingerprinting. In *IEEE Transactions on Dependable and Secure Computing* (2005).

[25] Libert, T. Exposing the invisible web: An analysis of third-party http requests on 1 million websites. *arXiv* (2015).

[26] Mowery, K., and Shacham, H. Pixel perfect: Fingerprinting canvas in HTML5. In *Proceedings of W2SP* (2012).

[27] Mowery, K., and Shacham, H. Pixel perfect: Fingerprinting canvas in HTML5. In *Proceedings of W2SP* (2012).

[28] Mowery, K., and Shacham, H. Pixel perfect: Fingerprinting canvas in html5.

[29] Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *IEEE Symposium on Security and Privacy* (2013).

[30] Olejnik, L., Acar, G., Castelluccia, C., and Diaz, C. The leaking battery: A privacy analysis of the html5 battery status api. In *IACR Cryptol. ePrint Arch.* (2015).

[31] Olejnik, L., Tran, M.-D., and Castelluccia, C. Selling off privacy at auction. In *NDSS Symposium* (2014).

[32] Pachilakis, M., Papadopoulos, P., Markatos, E. P., and Kourtellis, N. No more chasing waterfalls: a measurement study of the header bidding ad-ecosystem. In *Proceedings of the Internet Measurement Conference* (2019).

[33] Papadopoulos, P., Kourtellis, N., Rodriguez, P. R., and Laoutaris, N. If you are not paying for it, you are the product: How much do advertisers pay to reach you? In *Proceedings of Internet Measurement Conference* (2017).