Web Privacy Measurement: Scientific principles, engineering platform, and new results

Draft – Jun 1, 2014

Steven Englehardt, Christian Eubank, Peter Zimmerman, Dillon Reisman, Arvind Narayanan

{ste,cge,peterz,dreisman,arvindn}@cs.princeton.edu

ABSTRACT

The results of web privacy measurement have been very influential in online privacy debates. As a research field, however, web privacy measurement is immature and fragmented, and has not yet acquired an identity as a unified discipline. There are significant scientific and engineering challenges but the solutions tend to be ad-hoc.

We identify 32 web privacy measurement studies, cast them as instances of a generic experimental framework, and perform a thorough methodological analysis. We analyze design and implementation alternatives and make recommendations based on considerations of experimental rigor and engineering feasibility.

Next, we present a flexible, modular web privacy measurement platform that supports any experiment that fits the framework. It is also highly scalable and avoids many common pitfalls. Finally, as a case study of our methods and infrastructure, we measure the "filter bubble", i.e., the extent of personalization based on a user's history, by crawling approximately 300,000 pages across nine news sites and present evidence that this personalization effect has been greatly overstated in the popular press.

1. INTRODUCTION

Web Privacy Measurement — observing websites and services to detect, characterize and quantify privacy-impacting behaviors — has proved tremendously influential. While tools that block web tracking are used only by a small minority and address only a part of the problem, and solutions based on voluntary co-operation like Do Not Track have floundered, web privacy measurement has repeatedly forced companies to improve their privacy practices due to press coverage and regulatory action [6, 13].

On the other hand, web privacy measurement (hereafter, WPM) is hugely technically challenging, for both scientific and engineering reasons. First, WPM studies typically aim to attribute causality (i.e., to establish claims such as "the use of privacy feature X results in a 20% decrease in targeted advertisements"). Yet, the web is a complex, dynamic, interacting system with a multitude of actors. This introduces an incredible array of confounds, making such inferences very problematic to tease out.

Second, due to the scale of the web and the necessity of repeating experiments to establish statistically valid conclusions, WPM must typically be automated. This has proved a somewhat elusive goal. The web seems to resist automation for a variety of reasons. Published studies do not always report the engineering challenges involved and, as we will see, these often pose serious barriers.

These difficulties call for a principled approach with standardized methodology and the co-operative development of a measurement infrastructure. Instead, WPM has developed in a hurry in response to the rapid growth of privacy incursions online. As a consequence, there is a great degree of methodological inconsistency and reinvention. Sometimes authors appear to be unaware of solving almost identical problems in slightly different contexts compared to previous work, reflected by the lack of citation. From an engineering standpoint, there is a rampant replication of effort in building measurement tools and solving common problems. All this leads to results that are difficult to compare and sometimes contradictory, and little in the way of standard methods or infrastructure that a newcomer to the field can utilize.

We seek to remedy this situation. We start by framing the empirical component of WPM studies as instances of a general experimental framework consisting of an input phase and a measurement phase. We identify 32 WPM studies that fit this framework, favoring those that use automated methods. The aims of the studies fall into three categories: measurements of online data *collection*, *flow*, and *use*.

Our first contribution is a thorough methodological analysis of these studies (Section 3). We identify dozens of methodological and infrastructure considerations, highlighting 14 in particular. In each case, we discuss the alternatives and analyze the trade-offs, and make recommendations when possible. We seek to make this text a methodological guidebook for future studies.

Our second contribution is the development of a flexible WPM platform (Section 4). Its key features are a modular, plug-in based architecture to support a variety of types of WPM experiments, scalability, and an abstraction layer that exposes high-level functionality to the experimenter. We have incorporated solutions to many of the practical challenges in scaling browser automation and measurement. We are planning to open-source this platform with the aim of minimizing duplication of effort and allowing easy comparison of results.

As a case study of the power of our platform, we examine the "filter bubble," the supposed effect that occurs when online information systems personalize what is shown to a user based on what the user viewed in the past. While prior WPM studies have looked for the effect in web search results [19, 25, 56], curiously we are not aware of attempts to measure personalization by news websites based on click history.

Our third contribution is to fill this gap in the literature.

Based on experiments on nine major publisher sites, we find that (i) only some sites perform any history-based personalization at all, and only in a select few content boxes on news pages (ii) where personalization exists, the effect sizes are quite small — at most an 20% difference in categorical recommendations observed within a given publisher, much smaller than extent of contextual (i.e., non-personalized) recommendations, and (iii) while some areas of sites and some topics are personalized to be *more* similar to the user's click history, others are personalized to be *less* similar and more diverse. Thus, our results contradict the narrative of pervasive personalization leading users into feedback loops of reinforcing viewpoints and ideological "bubbles."

2. BACKGROUND

Online tracking. As users browse and interact with websites, they are observed by both "first parties," which are the sites the user interacts with directly, and "third parties" which are typically hidden trackers such as ad networks embedded on most web pages. Third parties can obtain users' browsing histories through a combination of cookies and other tracking technologies that allow them to uniquely identify users, and the "referer" header that tells the third party which first-party site the user is currently visiting. Other sensitive information such as email addresses may also be leaked to third parties via the referer header.

Data use and ethical concerns. The primary driver of online tracking is behavioral advertising. The ethics of the practice have come under fire [27]; in surveys users tend to express a dislike of targeted ads, frequently seeing them as "creepy" [32, 51, 52]. Legal scholar Calo worries that it allows firms to exploit consumers' cognitive limitations and manipulate perception and demand in unprecedented ways [11].

From a technical perspective, behavioral advertising is an instance of personalization. Many other online practices that involve some sort of personalization based on users' activities also raise ethical questions. Price discrimination is an obvious one, and in fact Andrew Odlyzko has argued that it is the main driver of online data collection, whether or not firms are aware of it [39]. He claims that perfect tracking will allow firms to almost perfectly infer consumers' willingness to pay for an item, and individualize prices accordingly.

A gentler form of price discrimination is "steering," where users are presented different search results for products based on an inferred willingness to pay. A prominent example is Orbitz who were found to show different hotels to users based on User-Agent (Mac vs. PC) [28].

In addition to search results for products, web search results, social media news feeds, and news websites have all been criticized for personalization, a.k.a. the "filter bubble". These criticisms have led to a TED talk [41], a book [42], and a great deal of press attention.

Finally, commercial actors are not the only ones personalizing content in worrisome ways — political campaigns are known to send customized emails to voters based on databases purchased from data brokers like Acxiom [4].

Web privacy measurement: a unified framework. At first glance measurement studies of targeted ads, price discrimination, or the filter bubble may appear to be different from measurements of cookies or PII leakage. **The central thesis that drives our work is the surprising fact that there are deep similarities between all these studies, both conceptually and in terms of engineering, and yet this has not been exploited by the WPM research community.**

To present our generic framework, let us first discuss web *personalization measurement*, and then see how the other types of studies differ from it. We treat the target system as a black box to which we have partial control over the inputs and the ability to observe outputs; the task of the experimenter is to learn the relationship between the two. In general these systems use machine learning for personalization and learn user attributes passively instead of (or in addition to) users inputting attributes actively.

The experimenter's task, then, has two stages: an input phase to "train" the system by interacting with it in a certain way and to allow the learning algorithm to operate. In addition, the input phase may also involve directly revealing information about simulated users such as entering demographic attributes into a form. The second phase is the measurement or "testing" phase to personalization. Typically, there are several experimental units (browser instances), some of which are randomly assigned to be controls which either don't participate in training or perform some default actions during training.

Data flow experiments also fit this framework, with two differences: the input phase usually consists of explicitly entering information into the target system, and the measurement phase may be more tightly coupled with the input phase; the measurements consist of checking if the inputs propagate to various target areas of the system.

In experiments on data collection, i.e., measurements of the mere existence of tracking elements on websites, the input phase is even more truncated. In some cases there is no input necessary; in other cases it may consist of different settings applied locally to browser instances (e.g., User-Agent string, Do Not Track preference) to test for systematic differences in tracking as a function of these inputs.

The similarities in infrastructure are even greater than these conceptual similarities would suggest. Broadly, all these experiments require the ability to navigate between pages and perform actions on them, and the ability to extract certain elements of interest from web pages (typically using a browser plugin or web driver). Either phase may be manual, but is typically automated.

The above model captures all controlled experiments with websites as targets. An interesting exception is the ProPublica Message Machine project [1] which analyzes personalized emails and is not controlled, which we include in our analysis anyway because it serves as an illustrative methodological counterpoint. We note that our model is inspired by prior work including Balebako et al and Tschantz et al [9, 49].

3. METHODOLOGY

We now present our analysis of the 32 web privacy measurement studies that met our criteria. We emphasize that this is not a survey of results (in fact, we don't mention results at all) but rather an analysis of methodological issues. Our goals are to lay out the menu of choices for study design and infrastructure, analyze their pluses and minuses from a scientific and engineering perspective, provide recommendations of alternatives when possible, highlight challenges and avenues for future research, standardize practices, and sur-

face information on engineering pitfalls that are typically not presented as part of published work.

3.1 Targets

The most fundamental aspect of a web privacy measurement study is what to measure. The studies in our survey fall into three categories: (1) measurement of data *collection*, i.e., the mere presence of trackers and tracking activity; (2) measurement of data *flow*, i.e., leakage of specific identifying information and sensitive attributes, typically to third parties, and (3) measurement of data *use*. The latter encompasses various types of personalization on web pages, such as ads, search results and prices.

Over time we see an interesting migration in the focus of studies from data collection and data flow to data use. There are several possible reasons for this. First, the privacy community has accepted that online tracking is rampant and is here to stay; measurements of cookie prevalence are simply not interesting anymore. Newer studies on data collection focus on more intrusive types of tracking such as browser fingerprinting [5]. Second, measuring data use and personalization requires more complex infrastructure and sophisticated analytical techniques, and the community has gradually acquired these capabilities over the years. Third, potentially discriminatory uses of online tracking have been coming to the public attention (e.g. Pariser's theory of the filter bubble [42]), incentivizing the latter kind of study.

Methodological question 1. *Which specific sites, business, and web pages to study?*

We found a variety of choices in terms of which specific sites or businesses to study. The most common choice was to study the most popular N websites by Alexa ranking, for some N; many studies broke down the measurements by Alexa category. One study was able to crawl the top 1,000,000 sites, essentially constituting a census [5], but this was prohibitive for most studies because of scaling issues. Since the most popular sites may behave differently from less popular sites, McDonald and Cranor pick a random 500 sites (from the top 1,000,000) in addition to the top 100 [33]. By performing crawls on these different sets of sites they found that the popular sites were more likely to set HTTP cookies and flash local shared objects than random sites. Roesner et al. use a similar technique [45]. Given such differences we recommend this two-set approach.

Many studies limit the crawl to the top-level page of a domain; counterexamples include Malandrino et al, Soltani et al, Ayenson et al and others [26, 46, 8, 38, 20, 55, 26, 45]. At least in some cases this was because the crawling infrastructure was not powerful enough to support clicking on links. Many sites behave differently on the front page compared to other pages, so we recommend that newer studies perform deeper crawls when this makes sense. The infrastructure issue is easily solved. Some studies limit the target to a single domain or entity — often a Google-owned service: text ads [9, 47], Doubleclick [24, 40], or search [56, 19, 25]. We discuss this further in Appendix C. □

3.1.1 Measuring data collection

The most basic type of measurement in our survey is the prevalence of stateful tracking technologies, especially cookies, on web pages. While in some studies this measurement is the primary objective [14, 40, 33, 21, 46, 45, 16], many other studies measure tracker prevalence as a side-goal [24, 40].

Methodological question 2. *How to detect which trackers belong to the same entity?*

We see at least three approaches. Mayer considers third-parties as having a shared "PS+1" [30]. Krishnamurthy et al. use a more complex heuristic involving querying ADNS records; they do this to account for CNAME aliases and the fact that different entities may use the same content-delivery network [21]. Finally, the authors of AdReveal simply use the Ghostery tracker database [24]; the limitation is that the database may not have full coverage of trackers. There doesn't seem to be an ideal solution here as of yet. □

Methodological question 3. *How to detect if a cookie is an identifying cookie?*

Again we see a variety of solutions. The simplest approach is to simply treat cookie values that are different in two different crawls as unique [33]. Olejnik et al. filter out cookies with values shorter than 10 characters [40]. Reisman et al. seem to have the most sophisticated heuristic: they look for cookies that have long expiry times, vary across different crawls, are sufficiently dissimilar between crawls as measured by the Ratcliff-Obershelp algorithm, have stable values within a crawl despite browser restarts, and are of constant length [44]. □

Several studies have measured Flash cookies (LSO's) [33, 46, 8] and other studies have looked at remarketing scripts [24] and scripts in general [21]. Measuring *stateless* tracking such as browser fingerprinting is harder [5]: this involves analyzing the behavior of scripts, and we discuss it in Section 3.2.

3.1.2 Observing data flow

Studies have examined the leakage of various types of identifying information and sensitive attributes from first parties to third parties. Identifying information includes name, username, and email address, and sensitive attributes include gender, age, location data, relationship status, Date Of Birth (DOB), Social Security Number (SSN), personal cellphone, education information, and other lifestyle information (sexual orientation, political beliefs, etc.) [30, 20, 26]. Naturally, inputting all these attributes into websites is a manual process. Once input, however, some of the measurement can potentially be done in an automated way. Some studies measure the leakage of information that is not explicitly input but rather implicitly collected, such as geographic co-ordinates, smartphone UDID [23], and browsing history [40].

Methodological question 4. *How do we tell if an attribute is being leaked?*

A simple approach is to store the set of strings corresponding to the sensitive attribute, perform a pattern search on the stored HTTP headers and other network communications, and manually identify false positives [20, 26]. An interesting technique is "diffing": to modify the values of the sensitive attributes stored with the first party and see which fields of communication change [30]. Since values may be encoded or even encrypted, such a technique seems very useful. We are not aware of serious efforts to automatically decode encoded values that potentially contain sensitive attributes.

Methodological Focus	Aspects	Examples / Tools / Techniques
Targets (§3.1)	Data Collection	stateful tracking (e.g., cookies), fingerprinting
	Data Flow	leakage of PII, personalization
	Data Use	price discrimination, targeted advertising
Infrastructure (§3.2)	Browser and driver	user-agents, web drivers (e.g., Selenium)
	Browser instrumentation	proxies, add-ons, plug-ins
Variables & Analysis (§3.3)	Attributing causality	randomized, controlled tests; avoiding confounds/cross-unit effects
	Demographic variables	browsing histories based on gender, race, ethnicity, topical interest

Table 1: Methodological components of web privacy measurement studies

Eubank et al. showed that "growing cookies" contain either lists of visited sites or behavioral segments, but were not able to decode them [14]. □

Often leakages of sensitive attributes are accidental, caused by poor engineering decisions. Other times they are deliberate efforts to share information with third parties. It is likely impossible to infer intent by analyzing the site; the best that one can do is to check for leakage via URL or POST parameters, as opposed to the referer (the former is more likely to be deliberate than the latter) [22, 20].

Methodological question 5. *How to detect cookie syncing?*

Olejnik et al. developed a method to detect a particularly problematic type of data flow known as *cookie syncing* (also known as "cookie matching") [40]. Cookie syncing allows two different third parties to link their pseudonymous cookie IDs of the same user; the technique is for one third party to include the cookie ID as a parameter of a URL which is a redirection to the other third party. The authors first detect this causal relationship between two successive HTTP requests and then scan the requests for shared parameters that could represent the cookie to be synced. We suggest that a more rigorous approach would be to use Reisman et al's method for ID detection (described above) and only check for parameter values that represent unique IDs. □

3.1.3 Inferring data use based on personalization

The primary way to detect uses of sensitive information is to detect personalization of web pages. A variety of targets have been considered: behavioral ads, bid prices in online auctions, web search results, product prices, and product search results. These studies typically have the dual purpose of confirming that data collection/flow is happening and measuring the specific uses to which it is being applied.

Ads have attracted a lot of attention since there is a multi-billion dollar industry dedicated to tailoring ads to individuals. Balebako et al. examined Google text ads [9] on a few specific publisher pages, AdReveal examined Flash and image ads from Doubleclick [24], Guha et al. looked at text ads in various contexts [18], and Wills and Tatar examined all types of ads from Google [55]. While these studies largely focused on ads from Google properties, Barford et al. expanded their analysis to harvest over 175K distinct display ads and analyze them to uncover the amount of ad targetting present in the *Adscape* [10].

Measuring ad personalization raises several major challenges. The first is to detect ads and reliably identify the same ad in different contexts. In the studies above, this was either done manually by visual inspection [18], by analyzing ad text content [9, 47], or by manually identifying URL patterns corresponding to ads [24]. Adscape utilized Easylist filters and also used heuristics such as standardized dimensions for image ads [10].

Detecting and uniquely identifying the same ad across contexts (especially image and Flash ads) automatically in a ad-network-independent manner appears to be well beyond the capabilities developed so far. This task is not specific to privacy measurement; for example, the ad search company `moat.com` is attempting to tackle this challenge [private communication].

Second, ads vary based on a large variety of factors including inventory churn and A/B testing. Confounds and noise apply to most types of personalization measurement, and we discuss this issue further in Section 3.3. That said, the level of noise in ads seems particularly high [18]. Third, ad-based studies often need to identify the landing pages of ads without being able to click on them (since that might constitute click fraud).

A fascinating recent study by Olejnik et al. analyzed the *bid prices* of real-time online auctions using a combination of crowdsourcing and crawling [40]. Key to this study is the fact that winning price notifications for auctions are sent from one server to another (ad exchange to bidder) *via the browser*. The authors were able to reverse engineer a number of URL patterns and price formats corresponding to these notifications (since there are a relatively small number of firms involved, and there is some standardization of patterns and formats, the authors were able to get a good coverage). The reason for this architectural decision (server-browser-server communication rather than server-server communication) by ad exchanges is unclear. Furthermore, the authors were greatly aided by the fact that several of the companies failed to encrypt these notifications (which they are supposed to do for competitive reasons). If either of these changes, it is entirely possible that the ability of independent researchers to study auction prices will disappear.

Price discrimination was studied by Mikians et al [34]; they looked at listed prices on 200 online vendors. They found that extracting prices in an automatic way is difficult and required vendor-specific scripts. In follow-up work they increased the scale to 600 vendors via crowdsourcing [35]. By building a plugin that allows users to highlight prices on a page, they built templates for extracting prices from a page. In the former work they also considered search results and prices associated with them, both product search results from web search engines such as Google and Bing, and searches on e-commerce sites.

Three studies measure web search personalization[19, 56, 25]. Here the research challenges are not in content extraction, which is relatively straightforward, but in applying the right controls, which we discuss in Section 3.3.

3.2 Infrastructure

Any infrastructure for web privacy measurement has two components: input (visiting sites/pages) and output (measurement of relevant targets). The former can be done manually by the researcher, by crowdsourcing, or can be automated. The latter typically involves at least some automation. Our focus in this section is on automation methods.

3.2.1 Browser and driver

The three choices for the user agent are HTTP libraries like curl or wget, lightweight browsers like PhantomJS (an implementation of WebKit), and full-fledged browsers. None of the studies we examined used the first option, for obvious reasons: the results will have little relevance to what real users experience on the web.

Methodological question 6. *How to trade-off performance, simplicity, and stability vs. fidelity and programmability in driving crawls?*

Lightweight and full-fledged browsers were both quite popular. The trade-off is performance vs. fidelity. PhantomJS has minimal resource consumption and allows 200 parallel instances on a commodity desktop [5]. The drawback is that it doesn't support plugins and may differ in other unknown ways from consumer browsers. Websites may also specifically detect stripped-down browsers and treat them as bots. Performing various measurements using both sets of browsers and comparing the results would yield valuable insights into the validity of using stripped-down browsers; this hasn't been done, to our knowledge.

There are a variety of choices to *drive* crawls, i.e., to instruct the browser to visit a set of pages (and possibly to perform a set of actions on each). The simplest choice is to use Javascript to launch different pages in an iframe or a separate window [14, 34]. The main advantage of this is engineering simplicity; further, it is platform-agnostic and makes comparisons between different browsers easier. The disadvantage, of course, is that there is no programmability beyond simply instructing the browser to visit a page.

More sophisticated choices are browser automation frameworks such as CasperJS for PhantomJS and SlimerJS or Selenium for Firefox, Chrome and PhantomJS. Selenium, in particular, is extremely popular in the studies we analyzed, since it is a cross-platform web driver for FireFox, Chrome, Internet Explorer, and PhantomJS. But it comes at cost of stability and scalability, with frequent crashes, memory leaks, and process-blocking. All these frameworks provide programmability, but only some come with built-in support for headless browsing (e.g., PhantomJS but not Selenium). Our platform, described in Section 4 drives a full commercial browser but provides the stability of the lightweight frameworks.

Several studies used a custom browser plugin to drive the crawl, including Mikians et al, Krishnamurthy et al and Roesner et al [34, 21, 45]. This makes sense mainly if a plugin is being used for the measurement task anyway. Finally, if the focus is on automating tasks within a page rather than visiting different pages, iMacros is a viable choice [34]. □

Methodological question 7. *How to avoid bot detection?*

Many websites have systems in place to detect clients that are bots. The usual intention is to prevent undesirable behaviors such as click fraud, spam, and content extraction via scraping. The most common response is blocking, but we have heard unconfirmed reports of more subtle behavior changes by websites such as disabling personalization.

Naturally, for WPM studies to be valid the experimenter must ensure that the targets treat the measurement browser instances as human users rather than bots. There is no guarantee that this can be achieved in automated studies — the measurement instances are bots, after all. One factor in the experimenter's favor is that bot detection techniques are typically intended to defend against distributed operations of a massive scale [43]. In WPM, on the other hand, the large scale typically arises from the need to interact with numerous (frequently, thousands) of targets in a single study. From the viewpoint of any one target, the experimenter's actions are at a scale which a human, albeit a very patient one, could execute in a matter of hours or days. □

3.2.2 Browser instrumentation and proxying

Most studies require some automated way of saving information from the browser. Modifying the browser to do this is called *instrumentation*.

Methodological question 8. *Should instrumentation be done using a plugin, proxy, source-code modification, or a combination?*

Instrumentation is typically done via an add-on or extension, though editing the source code of open-source browsers such as Chromium is sometimes preferred. Perhaps the best browser instrumentation tool is FourthParty [31] which has been used in several studies either directly or as a forked version [30, 29, 12, 14, 16]. Several studies used custom plug-ins [24, 35, 40]. This strategy is particularly useful in the case of crowdsourced studies so that the plug-in can double as a tool to extract information, visualize results, or both.

Several studies used a *proxy* instead of or in addition to browser instrumentation. Mitmproxy and Fiddler were the two popular choices, although custom proxies were also seen [19, 34]. When automating measurements, a proxy is essential if browser instrumentation is not done (as was the case with Mikians et al [34]). But proxies were also used to *modify* traffic to get around limitations of the instrumentation or automation framework. In Mikians et al [34] a proxy was used to modify DNT headers to test the impact of Do-Not-Track, as well as to strip X-Frame-Options headers to force sites to be framed (they needed sites to be framed because they used Javascript to drive the crawl). An ingenious application of a proxy is seen in Hannak et al [19] where they used crowdsourcing but had the users route traffic through a proxy controlled by researchers for easy data collection. □

3.2.3 Other tasks

There are various subtasks that occur frequently and may be considered part of the infrastructure. For example, dumps from proxies are at a raw level and don't contain much helpful information, so researchers must build these heuristically. The authors of FPDetective report detecting Flash files by looking for "magic numbers" [5].

Request hierarchy detection is a frequently encountered task. The idea is to detect the tree of causal relationships between all resource loads triggered by a web page. As a simple example, if a publisher embeds an ad iframe, there would be an edge from the top-level page URL to the iframe URL. Of course, web pages are complex, dynamic objects,

so this is tricky. The technique in Olejnik et al [40] describes the process of detecting the browser redirection that is evidenced in the *Referer* header and the ability to prove a causal relationship between HTTP requests. While this is a good heuristic, it is not foolproof (requests made from JavaScript, for instance, will not display the referer of the script provider, but rather just the visited site).

Fingerprinting detection requires behavioral analysis of web pages, both static analysis and dynamic analysis. Mayer and Mitchell claimed that detecting fingerprinting is infeasible [31]; it is a testament to how fast the field evolves that fingerprinting detection has already been carried out on a web-wide scale [5]. In that paper, The authors used a tool, JPEXS, to decompile Flash objects, search for various code strings, and performed manual analysis to look for suspicious calls, including calls that could send information back to server or opening a socket to bypass system-wide proxy settings. For dynamic analysis, they modified the browser (WebKit) source code. The rationale was to be able to do an in-depth behavioral analysis. This required deeper and more fine-grained access to the browser than the plug-in API provides, in addition to residing at a layer that JavaScript cannot tamper with.

Several studies executed distributed versions of their crawls [56, 19, 34, 40]. In almost every case this was done via PlanetLab. While several studies used only a handful of nodes (6 in Mikians et al [34], 10 in Hannak et al [19]), the browser plug-in Bobble used 308 PlanetLab nodes to measure Google search results [56].

3.3 Variables and analysis

Now we get to the methodologically trickiest part of web privacy measurement studies: varying an input parameter while keeping others fixed, and analyzing the behavior of the site(s) to be able to causally attribute the differences to the change in the input parameter.

3.3.1 Attributing causality

The primary tool available to the researcher to attribute causality between input and output is being able to run randomized, controlled experiments. This need is crucial; most of the studies that used crowdsourcing (which doesn't allow controlled experiments) used it either as a preliminary data-gathering phase or as a way to observe personalization in the wild, without attributing it to specific factors.

Methodological question 9. *Is scientifically rigorous causality attribution possible with crowdsourced data gathering?*

We found exactly one exception: ProPublica's MessageMachine, a project that reverse-engineers microtargeting in political campaign emails [1]. The project collects emails received by project participants from the Obama campaign, together with participants' demographic attributes. It then fits a decision-tree model to this data based on said attributes (age, sex, household income, state, and whether the user is a likely voter), and some features gleaned from previous emails. While this is an interesting approach, we caution that this type of analysis is not methodologically rigorous and could easily result in overfitting and causal misattribution.

Even with the ability to do randomized controlled experiments, there are several confounding factors. Location, User-Agent, etc. are obvious ones. Ads and search indices are subject to temporal variation (churn). A tricky type variation of temporal churn is the "carry-over effect" in web searches where the search history in the last 10 minutes affects results [19]. Distributed systems such as search engines have small but important consistency issues between the different nodes, so the identity of the node that is queried affects the results. Of course, the normal load-balancing architecture of these systems hides the identity of the node from the client, necessitating workarounds.

In studies on price discrimination, it is insufficient to demonstrate that changing some user attribute (say, location) affects the observed price. This is because a supply-side variable could be a confound: perhaps the firm incurs different costs in shipping to different locations, and these cost differences are sufficient to explain price variations.

Only one study so far has been able to convincingly attribute variations to demand-side variable [53]. They showed that the *distance between the center of the user's ZIP code and the nearest competitor store* explained 90% of price differences. While this doesn't mathematically prove that there isn't a confound, it makes it extremely unlikely. Actually exhibiting a demand-side model that predicts price differences is a superior standard than considering possible supply-side variables and arguing that they are infeasible [35]. □

Methodological question 10. *How to handle tricky confounds such as A/B testing?*

A/B testing is another nasty confound. While the effect of random noise can be nullified by making repeat observations from the same profile, the effect of A/B testing cannot. For example, perhaps some search engine segments new users into one of two categories for testing, one with and the other without personalization.[1] The only way around this is to average across different instances of synchronized crawlers rather than different measurements from the same instance.

Together, the studies we analyzed used a variety of techniques to increase the validity of causal attribution. The exact set of techniques necessary depends on the goals. A very common approach was to execute different crawls in lockstep. For distributed crawls (to measure the effect of location) this is challenging; syncing using NTP can be used [34]. Also common was to use the same IP or subnet to control for effect of location. When the effect of history tracking needed to be eliminated in Mikians et al [34], they blocked the referer header, 3rd party cookies, flash cookies and other stateful trackers. However this doesn't account for browser fingerprinting. Finally, one study found it important to use the Google search web interface rather than the API, since these are known to return different results. □

Profile pollution is a problem that affects history-based personalization measurement [10]. Suppose we wish to measure if a history reading sports articles causes the reader to be recommended more sports articles, regardless of the page that the user is currently visiting (i.e., behavioral as opposed to contextual targeting). Doing the measurements (testing phase) involves visiting articles in various categories. Unfortunately this continues to train the system, and there is no way to avoid it. Worse, the profile consisting of only sports articles that the system has built for the observer now gets "diluted" with non-sports articles. This problem seriously

[1] Of course, web privacy measurements themselves are a form of A/B testing, so both sides end up A/B testing each other.

limits the number of test page visits as a fraction of training page visits, which can only be compensated for by increasing the sample size.

Methodological question 11. *How to eliminate "cross-unit effects?"*

The independence assumption is implicit in most studies: the effects (say, ad categories) observed are assumed to be independent identically distributed samples from an underlying distribution, so that averaging will allow approximating that distribution. A related assumption is that the effect of variables changed in one browser instance do not affect the training step of other instances, even those on the same machine. Tschantz et al. question both these assumptions, and provide evidence of "cross-unit effects" in the measurement of the diversity of Google text ads [50]. These cross-unit effects could be due to browser fingerprinting or IP-based tracking. If confirmed for other targets, this would be problematic for rigorous web privacy measurement. Their solution is a minimal set of statistical assumptions that can only detect but not quantify effects. Given the goals of most of the studies we considered, this solution is unsatisfactory. Another possible solution is to run measurements from a variety of machines with different IPs in the same subnet, and average across those. This would greatly complicate the engineering task. □

3.3.2 Variables: non-user-specific

Now we consider the techniques for changing input variables. First we consider those that are not inherent user attributes; these are easier to change.

Methodological question 12. *What are realistic ways to vary the User-Agent and location attributes?*

To change the *User-Agent* (to simulate different browsers, operating systems, or a combination), there are two approaches. The trivial way is to simply change the User-Agent string [19]. A more rigorous way is to actually change the user agent [14, 34, 28, 47, 56]. This is much harder to do because the researchers' automation and measurement infrastructure may only work on some platforms/browsers. Nevertheless, this is worth striving for: we are aware that some sites use scripts that check for the validity of the User-Agent string, say by fingerprinting the browser JavaScript engine, but it is not clear how widespread their use is.

Simulating a different *location* generally involves running crawls from an IP at that location. As mentioned in 3.2, PlanetLab has been a popular choice. Additional studies used PlanetLab to vary the IP-geolocation of a crawl [56, 19, 34]. Some used cloud hosting, but there have been reports of EC2 instances in particular either being located at a different place than advertised, or bugs in geo-location APIs used by measurement targets.

One study proxied its traffic through machines at different IPs instead of running the crawler out of them [53]. The same study also found a hack that applied to their specific target, `staples.com`: they found the user's inferred ZIP code stored in a cookie, and that modifying the cookie had an effect identical to measuring from an IP in the corresponding ZIP. This type of "active" as opposed to passive measurement has potential for other targets/measurements as well, and seems to be underutilized. □

Some studies examined the effect of page context [34], and other variables [40]. Other studies looked at the effect of using privacy tools including blocking tools, opt-out cookies, and Do Not Track [9, 16, 29, 26].

3.3.3 Variables: demographics

In terms of user attributes, demographic variables such as name, age, gender, race/ethnicity and affluence are the most basic. When these variables are explicitly input into the system such as by editing a profile page, it is straightforward [19, 30, 26]. But when the variables are inferred, it can be tricky. Sweeney used white- and black-identifying first names (i.e., names highly correlated with certain ethnicities) as web search inputs [47].

Methodological question 13. *How to create realistic browsing histories corresponding to different demographic and interest categories?*

Especially of interest is creating browsing histories corresponding to different demographics, since various systems have been hypothesized to learn these demographic categories from browsing patterns (behavioral ads in particular). A study of price discrimination modeled histories of "affluent" and "budget" shoppers based on Alexa categories [34]. Another study used the Quantcast list of top sites with demographic breakdowns [19]. However, this data is incomplete and has limitations.

3.3.4 Variables: interest categories

Since behavioral segments often represent interest categories, numerous studies simulated browsing histories of users with topical interests [34, 40]. As a practical matter, this choice was probably driven at least as much by the availability of data from Alexa's categorization of websites as by any scientific considerations. Another choice was to use Google search results for topical keywords [34]. (Additionally, it is also possible to simulate *intent*: in Olejnik et al [40] this was done by visiting product pages on e-commerce sites.)

While "extreme profiles" of interest categories are a useful tool for *detecting* personalization, it is important to ask how relevant the results are to the differences that real users may observe. In other words, how much of a topical skew exists between the interest categories of typical users? This is not clear. □

Methodological question 14. *How to obtain browsing histories of real users for research?*

The last point brings us to an important limitation: none of the studies obtained real users' browsing histories for controlled experimentation. *Search* history on a small scale was obtained and studied in mMjumder and Shrivastava, [25]; perhaps this is viewed as less privacy sensitive than browsing history. If browsing histories tagged with demographic attributes (say, race) were available, it would allow the experimenter to repeat those histories in different browser instances, controlling other variables like location, and test the hypothesis that race is inferred by existing advertising systems based on browsing history and is used to target ads. Whether or not the inference is explicit or a side-effect of machine learning may not be possible to infer, but the difference would be immaterial if one is attempting to determine disparate impact [54].

Browsing histories *aggregated* by demographic category, together with a generative model for simulating an individual history from the aggregate, would be largely sufficient for this purpose, although perhaps not as scientifically rigorous, since all generative models have limitations.

Individual browsing histories *without* demographic attributes would allow testing at least the *extent* of personalization seen by real users attributable to browsing history. This is similar to crowdsourcing studies, but with better controls. However, this hasn't been done either. One concern with this approach, even if technically feasible, is that revisiting the entirety of real user browsing history is ethically problematic — some URLs may incorporate capability-based access control or may include identifying/sensitive information, and visiting them may impact the user due to poor engineering practices.

Studies outside the web privacy measurement community have used various datasets of real user logs for studying other questions [17, 15, 2]. We attempted to obtain these datasets, but they were either no longer available or prohibitively expensive.

An interesting way to obtain an approximation to real browsing history, but without demographic attributes, appeared in Liu et al [24]. This is based on the AOL search query log dataset, which consists of the queries made by 650,000 pseudonymous users over a three month period (March–May 2006). For a subset of users, they submitted each user's search query to Bing and visited the top 5 results returned. Of course, only a subset of real users' web browsing results from web searches. Nevertheless, such profiles model two important aspects of real browsing histories: the distribution of popularity of web pages visited, and the topical interest distribution of real users. The reason to recreate the users' searches on a current search engine rather than simply using the sites visited by the AOL users (which are recorded in the AOL dataset) is that the distribution of websites visited by real users changes over time as websites rise and fade in popularity, whereas the distribution of users' interests can be expected to be more stable over time.

Finally, it is possible to obtain an approximation to real browsing histories (tagged with demographic attributes) from Twitter, under the assumption that Twitter users post links that are a random sample of the pages they visit, and using various heuristics to infer demographic characteristics from user profiles. These heuristics were suggested in Mislove et al [36].

3.3.5 The use of machine learning.

Machine learning is useful in personalization measurement in a few ways. First, if the experiments are not randomized and controlled, it is the only way to determine the effect of the input on the output [1]. Second, even in controlled experiments, in order to study the combined impact of more than one variable learning may be necessary [55].

A quite different use appears in Guha et al [18] and Franklin [16]. If the output is not a single variable or distribution (say, the number of ads in each category), it becomes tricky to determine if there is any impact of the input on the output. In other words, if the experimenter does not have a specific hypothesis for how the output is affected by the input, there is no test to apply. Machine learning can help: showing that the *input*, framed as a hidden variable, can be learned from the *output* demonstrates the existence of an effect, although it still doesn't help determine what the effect is. We explore this further in Appendix A.

3.4 Engineering pitfalls and challenges

In our own experimental work and in discussions with other researchers, we were struck by how often we face similar engineering problems in getting these measurements to work and to scale. To surface knowledge of these common pitfalls and challenges, we reached out to nine groups of researchers from our survey. Broadly, the responses fell into four categories: (i) difficulties in automating crawls, especially due to instability of available tools (ii) limitations of instrumentation tools and information extraction from web pages, (iii) limitations of APIs and rate limits, and (iv) ethical concerns. Appendix C has the details.

4. MEASUREMENT PLATFORM

By examining the challenges faced by other researchers, we determined that our WPM platform should be flexible in its ability to manage a wide variety of large-scale, automated studies. This core vision translated into a few key design requirements.

4.1 Requirements

WPM experiments consist of three major tasks. The first task is mapping high-level commands, such as visiting URLs or extracting news articles from websites, into automated browser actions. The second task is collecting and consolidating crawl data, such as cookies set by the browser and JavaScript calls, in a unified manner. The last task is to create automated tools to perform specific analyses on the data in order to answer individual research questions.

While the third component is study-specific, the browser automation and measurement components are designed to be general-purpose by adhering to the following principles.

Scalability. The primary advantage of browser automation is that it enables researchers to repeatedly visit sites at a rate infeasible for humans. Hence, despite the potential for page freezes, the platform should ensure steady progress throughout a crawl and, potentially, increased speed through browser parallelization.

Stability. During the course of a long crawl, a variety of unpredictable events, such as page timeouts or browser crashes, could halt the crawl's progress or, even worse, corrupt the data. The browser automation framework should recover from such events gracefully, quickly and intact.

Abstraction layer. The platform's automation API should serve as a user-friendly abstraction, hiding the complexities involved in performing browser tasks such as finding and clicking buttons on a page. Keeping commands at a high-level reduces the complexity of the scripts used for driving experiments.

Modularity. In the course of actually implementing our platform, we had to choose specific libraries for tasks such as driving the browser. However, adopting a modular design will enable us to easily switch the underlying tools in our platform as new technologies emerge.

Realism. Our automation framework should mimic a real person surfing the web as closely as possible—both in terms of the underlying browsing technology and the way in which the platform interacts with page. Websites detecting automated browsers may act in a pathological manner, thus weakening experimental results.

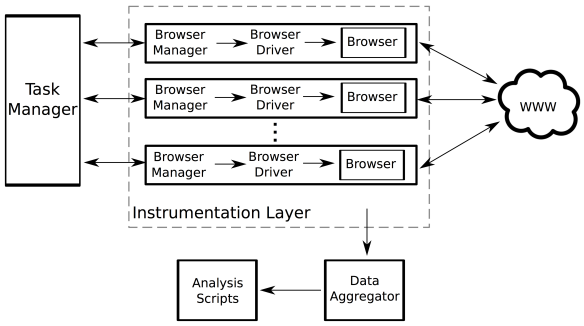

Figure 1: Generalized WPM framework
The task manager ensures crawls are robust, the browser manager converts high-level commands into automated browser actions and the data aggregator robustly receives and pre-processes data from browser instrumentation tools.

Repeatability. To promote scientific rigor, the platform should enable researchers to easily reproduce experiments. It should also log as much data as possible in a standardized format so research groups can easily run their analysis scripts on data created by other teams to verify results.

4.2 Design and Implementation

We divided our browser automation and data collection infrastructure into three main modules: *browser managers* which act as an abstraction layer for automating individual browser instances, a user-facing *task manager* which serves to distribute commands to browser managers and a *data aggregator* which acts as an abstraction layer for browser instrumentation. We implemented the entire WPM platform using Python. A high-level diagram of our WPM platform is contained in Figure 1.

Browser managers. Requiring browser managers as an abstraction layer stems from our choice to use Selenium for automation. In terms of its advantages, Selenium implements the WebDriver API for driving the major commercial browsers (i.e. Firefox, Chrome, and Internet Explorer) and support an extensive range of web technologies, such as plugins, addons, HTML5 features. By leveraging pyvirtualdisplay to interface with Xvfb and support headless browsing, we increased the number of parallel browser instances that can be run on a machine using Selenium and enabled crawls to be executed on remote machines. Beyond not supporting various web standards, lightweight alternatives, such as CasperJS-driven PhantomJS instances, are more susceptible to bot detection due to the underlying JavaScript engine and edge cases in simulating browser behaviors [3].

Browser managers wrap around Selenium—instantiating browser drivers given a configuration of user preferences (e.g. turning on Do Not Track) as well as mapping high-level commands (e.g. visiting a site) into specific Selenium subroutines. Despite its advantages, Selenium frequently crashes or hangs indefinitely due to its blocking API [7] as it was designed to be a tool for webmasters to test their own sites rather than an engine for automated crawls. Since each browser manager exists within it own process, this abstraction layer protects other platform components from browser failures and provides a clean interface for graceful recoveries.

Task manager. The task manager distributes user commands to browser managers given the assumption that crashes and freezes are inevitable over long crawls. The task manager monitors the browser managers, restarting one if it detects a crash or failure to complete a command within a time limit (which can be set on a per-command basis). In order to ensure that the individual browsers appear to originate from the same user, the task manager passes user preferences to restarted browser managers and copies over user data such as cookie and history databases. As will be discussed in the next subsection, the platform has native support for different methods of issuing commands to multiple browser managers.

Data aggregator. In Section 3, browser automation and instrumentation are presented as separate issues; however, one of these two components crashing can fatally disrupt the other. Since instrumentation tools often write data as well as collect it, browser crashes (or instrumentation failures) can severely corrupt data.

The data aggregator, which exists within its own process, receives data during the course of the crawl, manipulates it as necessary and writes it to a central SQLite database. By exposing a socket interface to all browser instances, it serializes data from an arbitrary number of browsers without the need for database access on a per-browser basis. Furthermore, isolating the data aggregator behind a network interface ensures that browser crashes in other process do not corrupt the data. As opposed to client-server databases, local databases do not require end-user setup and enable researchers to easily share data. However, socket connections to the data aggregator easily provide support for other frameworks such as MySQL as studies grow in scale.

We experimented with different instrumentation options, including FourthParty and mitmproxy, primarily leveraging the latter to drive our platform's instrumentation. When compared to plugin-based instrumentation, proxies reduce the computational requirements placed on the browser itself, enabling faster browsing.

As an example workflow, the browser manager receives a command from the task manager to extract information from a given news site. The manager translates it into a series of low-level commands that instruct the browser driver to visit the site and perform various actions on the page. Browser instrumentation tools parse the page content and send the relevant data (in this case, headlines) to the data aggregator, which logs to the central database.

4.3 Advanced capabilities

Beyond implementing the core functionality required to drive scalable crawls, we drew from the lessons of previous WPM research to add additional features useful for conducting measurement studies in a rigorous and general manner.

Native multiprocessing support. Dividing a crawl across multiple browsers speeds up the course of a measurement while running multiple crawls in a synchronized manner enables multiple measurements of the same object without variation from temporal effects. The Task Manager API exposes four options for issuing commands to multiple browser managers: (i) on a first-come-first serve basis, (ii) to individual browser managers and to all browsers (iii) synchronously or iv. asynchronously.

Bot detection mitigation. While intended to thwart click-fraud and undesired web scraping, bot detection

threatens the validity of web measurement results. In order to simulate real user behavior, we implemented countermeasures such as simulated mouse movements, screen scrolling and randomized delays when loading pages.

Browser fingerprinting mitigation. For researchers running multiple browsers from the same machine, experiments may suffer from cross-unit effects if online trackers link these browsers together through browser fingerprinting. As a countermeasure, our platform enables researchers to randomize browser settings or explicitly configure the user-agent string, extensions pre-loaded into the browser and screen resolution. Since Nikiforakis et al [37] note that randomizing user-agent can cause a browser to receive HTML designed for another type, adopting their suggestion for supporting subtle variations in properties implemented at the browser-level is one future direction in fingerprint mitigation.

Better profile support. As online tracking largely depends on a variety of client-side persistent storage vectors to the extent that these vectors have been observed respawning each other [46, 33], maintaining a persistent browsing profile the includes these additional vectors is essential for consistently simulating a user. Beyond adding support for transferring browser preferences and storage vectors, such as HTTP cookies, HTML5 localStorage, HTML5 Indexed DB and Adobe Flash objects, between browser managers within a crawl, we provide a simple interface for saving and loading these profiles between crawls to enable simulating a specific user over longer studies.

Log and replay architecture. In order to promote experiment reproducibility, the platform logs all user-configurable parameters as well as the task master command history, complete with timestamps and whether given commands executed successfully. Following the public release of our platform, researchers can use the history logs in order to replay studies using an actual instance of the infrastructure.

Towards a distributed infrastructure. Drawing from [56, 19, 34, 40], we have implemented measures to move the platform to a distributed framework. First, further decoupling the task manger from the browser managers would easily enable us to move browsers to remote machines given an effective means of inter-machine location. Expanding the socket communication interface, already used for sending information to the data aggregator, in order to isolate components of the platform is a clear next step for transitioning to a fully-distributed platform.

4.4 Evaluation

Already, we have leveraged the new WPM platform to conduct multiple large-scale experiments, including the news personalization in the next section. We now present an overview of our platform's empirical performance, especially with respect to our original design goals.

Stability. When running a light wrapper around Selenium, without the isolation provided by a browser manager and task manager, the stability was continually poor. Isolating Selenium in a separate process allows us to detect and handle all hangs or crashes and preserve a consistent profile with which to continue crawling. Without process isolation and a timeout of 40 seconds, the best average we were able to obtain was 800 pages without a freeze or crash. Even in small scale studies, the lack of recovery led to loss of training profiles and measurement data. With isolation we recover from all browser crashes and have observed no data corruption during crawls of 10,000 pages while retaining a continuous browsing profile. During the course of our news personalization study described in Section 5, we successfully crawled 281,494 pages and recovered from 2,252 failed page loads.

Resource usage. When using the headless configuration, we are able to run up to 20 browser instances on a 16GB, i7 quad-core commodity desktop. Due to Firefox's memory consumption, parallel crawls that require continuous browsing history are memory limited while parallel crawls which require a fresh browser with each page load are typically CPU limited and can support a higher number of instances. On the same machine we can run 35 browser instances in parallel if the browser state is cleared after each page load.

Generality. For the news crawls described in Section 5, the browsing and measurement was completely automated with the exception of writing custom template scripts for extracting headlines. Our platform was able to reduce the lines of code (LOC) required to automate the experiment by 70% over running the study on a a previously coded lightweight Selenium wrapper. This reduction in LOC was complemented by the modularity of the platform, where general methods (e.g., link extraction) were included as a module that can be shared across many experiments.

Modularity. Our platform abstracts away the details of the automation framework and trivially supports other browser/driver configurations, although thus far we have only done large crawls with Firefox/Selenium. We support different measurement options (instrumentation via FourthParty, or proxying via mitmproxy) and have successfully tested both in isolation and in combination.

Study reproducibility. A unified, open source platform for web measurement presents an excellent opportunity for measurement reproducibility and research transparency. Ideally, a researcher can branch off a stable version of the repository, implement the relevant study-specific changes and share the links to that repository to allow other researchers to review the code used during measurement. In addition, any bug-fixes or new platform features added can be pulled back upstream to the platform repository for the benefit of other researchers pursuing similar studies. The combination of an open, stable forked release and a full command database could be used as an input to conduct a replay of a crawl to verify a given set of results. Furthermore, our platform also saves the browser settings used during an experiment to ensure fingerprint consistency between sessions or replays.

We have built a platform that allows us to quickly launch new WPM experiments (we present such a case study in the next section) and there are many possible studies that could benefit from utilizing our infrastructure. We are eager to collaborate with other researchers on such tasks as well as new WPM studies, and avoid duplication of effort. We plan to open-source our platform at the earliest opportunity.

5. NEWS PERSONALIZATION

The level of personalization occurring on news outlets is a concern often raised in the discussion of privacy, filter bubbles and their potential effects. However, the current level of measurement equates to anecdotal evidence or manual study [48]. As proof-of-concept of the effectiveness of our

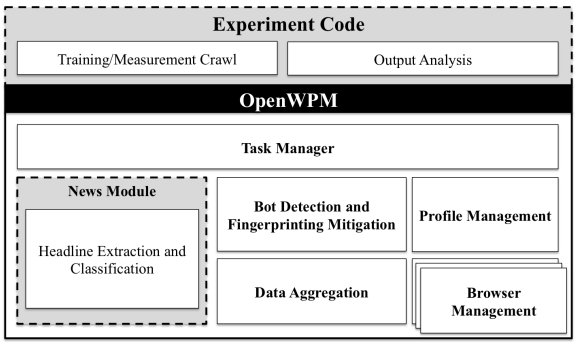

Figure 2: News Personalization Measurement Code for experiments run on OpenWPM communicate with the Task Manager. We also implemented a module to perform custom tasks specific to our news study.

generalized infrastructure, we present an automated, rigorous measurement of history based personalization of article topics on several major news sites conducted using our infrastructure (see Figure 2).

5.1 Measurement Procedure

Training phase. Measuring news personalization required creating a variety of profiles that mimic the past browsing behavior of users interested in specific topics. In order to maximize the likelihood of observing personalization, we created extreme profiles of users who browse one specific topic (e.g., sports) within each news publisher. We also maintained a control profile which contained no history or cookies at initialization and was not configured to prevent cookie setting or caching.

To create extreme profiles, we first collected links to news articles within each publisher. This was done by crawling and parsing sitemap XML files. We then utilized a text classification service, Aylien, to classify each news article. For each URL uploaded to the service, it returns the International Press Telecommunications Council (IPTC) subject code and relevancy score for the article located at the link. We then mapped each IPTC code to its first-level subject code (i.e., sport/basketball/NBA would be mapped to simply sport). Utilizing a common classifier across all publishers is important as separate publishers have the potential to categorize similar article content in different categories. Through manual inspection, we verified our classifier to be subjectively accurate when using publisher-specific categories as ground truth.

With a database containing publishers and classified articles, we created profiles for the five categories that contained the largest number of links for a given publisher and category. To accumulate browsing history over time, we automated profile generation to execute daily. For each of the six categories (five IPTC categories and one control with no history), we simulated eight users browsing to fifty randomly chosen links within each respective category. These 432 user profiles browsed over 450 links each during the training phase that spanned two weeks.

Measurement targets. We focused our efforts on nine of the most prominent news websites: Fox News, CNN, BBC, USA Today, Daily Mail, Huffington Post, Time, The Guardian, and NBC News[2]. We sought to measure personalization stemming from browsing content entirely within a given publisher (intrasite personalization).

We extracted headlines from content areas on these publisher sites, both on the front page and on article pages, based on a visual inspection to identify content areas that might possibly show topic-wise personalization. On article pages, the bulk of these content areas turned out to be from third-party recommendation boxes run by Taboola[3] and Outbrain[4], two large content recommendation and delivery engines which are found on many mainstream news publishers. These third-party providers track user interaction with the news publisher in order to provide recommended content both from within the publisher and across the web to sponsored articles. The content areas we studied can be broken down down into groups served by content recommendation engines. Taboola serves Daily Mail, USA Today, Huffington Post, and Time while Outbrain serves CNN, Fox News, The Guardian, and NBC News. BBC provides only native recommendations.

Measurement phase. For each (profile, publisher) pair, we initialize a browser instance with the corresponding profile and visit the front page of the publisher followed by a set of randomly selected articles from that publisher which are stored in our database. We then collect the headlines and links for those articles located in a "Latest Headlines" box. This list of articles contains links to content recently added to the publisher within a variety of categories. Note that each publisher labels this list with slight variations (e.g., "Recent Headlines").

On each article page visit, we collect links from two types of content boxes. "More From [Publisher]" is a list presenting recommended content that is hosted within the publisher. If no third-party content recommendation provider is used, this is the only list presented to the user. "Around the Web" provides links to articles on other publisher sites.

Preventing Confounds. To mitigate cross-unit effects, we randomized the user-agent and extensions used for each browser instance, and disabled Flash content, to ensure that different instances appear to be different users.[5] To prevent skew from churn of site contents, we synchronized all the browser instances (control and training) by exercising the native multiprocessing capability afforded by our platform. We ensured that each instance visits the same set of links at the same time during the measurement phase. Lastly, during manual inspection of the collected data we observed A/B testing of article headlines (different headlines for the same article), but since we classify based on article text this has no effect on measurement.

5.2 Experiments

Personalization. We run crawls on each publisher using browser instances trained by browsing on the same publisher's site. The training is done by visiting a set of 50 articles per interest category (e.g. technology articles) on each

[2]We omitted the New York Times and the Wall Street Journal sites because of their paywalls

[3]`http://www.taboola.com`

[4]`http://www.outbrain.com`

[5]We checked the Flash content of several pilot crawls to verify that no unique identifiers are stored for the publishers or content recommendation services studied.

site, every day or two over a span of two weeks. In the measurement phase of the study, each profile visits the front page of its respective publisher as well as 50 articles pages chosen randomly from any category. With a per-profile training phase consisting of over 450 article visits and a measurement phase consisting of 50 article visits, we maintained a ratio of at least 9 to 1 training to measurement URLs traversed.

Contextual recommendation. We run crawls on each publisher without building any browsing history to make a measurement of contextual recommendation. For each (publisher, category) pair, a clean browser instance is loaded and a single visit to an article in that category is made. This measurement is repeated up to 50 times per category.

5.3 Analysis

We measure personalization by computing the skew in observed article categories toward or away from the training category. For example, a browser instance trained on sports articles might see an overall increase in the number of sports-related articles recommended, or it might see a decrease and a corresponding increase in other categories. For each measurement we apply the binomial test for statistical significance of an increase or decrease, as opposed to the null hypothesis that there is no difference in the expected number of recommended articles in the training category compared to control instances. Each displayed article is considered to be an independent trial and a success is defined as an article belonging to the same category as the training profile.

To summarize our results, we found *no statistically significant deviations* of trained instances from control instances in article category distribution for *front page visits*. As such, the remainder of the analysis will focus on differences of headlines displayed on article pages.

On the article pages of both the Outbrain and Taboola news publishers we find statistically significant levels of personalization in many of the news boxes, as summarized in Table 2 in the Appendix.[6] However, the levels of personalization are quantitatively small, with a level of change between the control and measurement set of articles ranging from 3.1% and 19.2%. To be able to estimate these small sample sizes and reject the null hypothesis, we used large sample sizes amounting to thousands of article views for each publisher/category pair. We also limit our analysis to categories which comprise at least 10% of the overall number of links seen by the control. Our statistical confidence is reflected in the p-values.

The results for which we measure a statistically significant level of contextual recommendation are shown in Appendix 3. The levels of contextual recommendation are relatively consistent across publishers with most having between 20% and 40% increase in the number of articles that match the category of the host article over average. These results show that contextual recommendation is a very strong effect in the absence of browsing history. Note that the presence of strong contextual recommendation does not influence personalization measurements presented earlier as each set of measurements occurs on the same set of articles and therefore experienced the same contextual effects.

Aside from its contribution to the "filter bubble" dialogue, this study further underscores the potential for our platform. WPM studies previously executed were often limited in scale due to frequent crashes and data corruption which often required some amount of periodic manual intervention. Illustrating the scale of studies that are possible with our platform, our news study created 432 user profiles trained on 255,790 page visits and conducted a measurement of 25,704 articles with 14 synchronized browser instances at a time - something that would have simply been previously infeasible. Furthermore, personalization is difficult to detect and requires rigorous experiment methods which our platform provides support for out of the box.

[6]We tested for and rejected the hypothesis that our results could be skewed by other types of personalization like publishers not recommending an article that the simulated user has already read.

5.4 Limitations

It is important to point out several limitations of our case study, and to clarify that we do not claim to have conclusively proved the absence of a filter bubble. First, we only looked for history-based personalization and not location-based or other types of personalization. Of course, news websites have local editions and possibly other types of localized customization, but this is not as much of a privacy concern as history-based personalization since it doesn't require tracking of users.

Second, we used human judgment to determine which content areas of news websites had any possibility of history-based personalization based on the description. It is possible that we have missed some. Future work focuses on obtaining all the links on a page in an attempt to measure the topic skew over all articles presented to the user. However, our discussions with publishers revealed a rationale for serving personalized content within limited areas: server-side caching limitations would effectively prevent a large-scale implementation of site-wide dynamic content. Personalizing entire pages for every site visitor would be impractical.

6. CONCLUSION

Web privacy measurement has the potential to play a key role in helping academics, journalists and other neutral parties keep online privacy incursions and power imbalances in check. To achieve this potential, WPM needs scientific rigor as well as improvements in the ability to design and run experiments on a wide scale. In this work, we've taken steps to unify the field conceptually, methodologically, and by presenting a flexible experimental platform. We are keen to collaborate and are planning to open-source our platform as well as share data from our crawls.

As an agenda for the field, we present four key items. First, applying machine learning based information extraction techniques so that WPM research can be carried out on a web-wide scale instead of focusing on specific sites, providers, or trackers. Second, continuing improvements to the infrastructure, in particular, adding "high-level capabilities" such as logging into websites with a given username and password. Third, a central repository for sharing crawl data as well as other lists such as XPaths, templates, or regular expressions for extracting relevant information from pages. Finally, automated, longitudinal measurements so that privacy practices of various entities can be monitored over time — a "web privacy census."

Acknowledgements. We're grateful to numerous researchers for useful feedback. In no particular order: Joseph Bonneau, Edward Felten, and Matthew Salganik at Princeton, Fernando Diaz and many others at Microsoft Re-

search, Gunes Acar and Marc Juarez at KU Leuven, Vincent Toubiana at CNIL, France, and Lukasz Olejnik at INRIA, France.

7. REFERENCES

[1] Message Machine: Reverse engineering the 2012 campaign. http://projects.propublica.org/emails/, 2012. Accessed: 2014.

[2] Databases in WRDS - comScore. http://wrds-web.wharton.upenn.edu/wrds/about/databaselist.cfm, 2013.

[3] Supported web standards. http://phantomjs.org/supported-web-standards.html, 2014.

[4] Nathan Abse. Big data and microtargeted political ads in election 2012: The challenge ahead. IAB presents: Innovations in web marketing and advertising., 2012.

[5] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.

[6] Julia Angwin. What they know. The Wall Street Journal. http://online.wsj.com/public/page/what-they-know-digital-privacy.html, 2012.

[7] Selenium Browser Automation. Selenium faq. https://code.google.com/p/selenium/wiki/FrequentlyAskedQuestions, 2014.

[8] Mika Ayenson, Dietrich J Wambach, Ashkan Soltani, Nathan Good, and Chris J Hoofnagle. Flash cookies and privacy II: Now with HTML5 and ETag respawning. *World Wide Web Internet And Web Information Systems*, 2011.

[9] Rebecca Balebako, Pedro Leon, Richard Shay, Blase Ur, Yang Wang, and L Cranor. Measuring the effectiveness of privacy tools for limiting behavioral advertising. In *Web 2.0 Workshop on Security and Privacy*, 2012.

[10] Paul Barford, Igor Canadi, Darja Krushevskaja, Qiang Ma, and S Muthukrishnan. Adscape: Harvesting and analyzing online display ads.

[11] M Ryan Calo. Digital market manipulation. *University of Washington School of Law Research Paper*, (2013-27), 2013.

[12] Abdelberi Chaabane, Yuan Ding, Ratan Dey, Mohamed Ali Kaafar, Keith Ross, et al. A closer look at third-party OSN applications: Are they leaking your personal information? In *Passive and Active Measurement conference*, 2014.

[13] Federal Trade Commission. Google will pay $22.5 million to settle FTC charges it misrepresented privacy assurances to users of Apple's Safari internet browser. https://code.google.com/p/selenium/wiki/FrequentlyAskedQuestions, 2012.

[14] Christian Eubank, Marcela Melara, Diego Perez-Botero, and Arvind Narayanan. Shining the floodlights on mobile web tracking - a privacy survey. W2SP 2013.

[15] Seth Flaxman, Sharad Goel, and Justin M Rao. Ideological segregation and the effects of social media on news consumption. *Available at SSRN*, 2013.

[16] Michael Franklin. A statistical approach to the detection of behavioral tracking on the web. Princeton University, 2013. Undergraduate senior thesis.

[17] Sharad Goel. Demographic diversity on the web. http://messymatters.com/webdemo/, 2010. Accessed: 2014.

[18] Saikat Guha, Bin Cheng, and Paul Francis. Challenges in measuring online advertising systems. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010.

[19] Aniko Hannak, Piotr Sapiezynski, Arash Molavi Kakhki, Balachander Krishnamurthy, David Lazer, Alan Mislove, and Christo Wilson. Measuring personalization of web search. In *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013.

[20] Balachander Krishnamurthy, Konstantin Naryshkin, and Craig Wills. Privacy leakage vs. protection measures: the growing disconnect. In *Proceedings of the Web*, volume 2, 2011.

[21] Balachander Krishnamurthy and Craig Wills. Privacy diffusion on the web: a longitudinal perspective. In *Proceedings of the 18th international conference on World wide web*. ACM, 2009.

[22] Balachander Krishnamurthy and Craig E Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*. ACM, 2009.

[23] Balachander Krishnamurthy and Craig E Wills. Privacy leakage in mobile online social networks. In *Proceedings of the 3rd conference on Online social networks*. USENIX Association, 2010.

[24] Bin Liu, Anmol Sheth, Udi Weinsberg, Jaideep Chandrashekar, and Ramesh Govindan. AdReveal: improving transparency into online targeted advertising. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM, 2013.

[25] Anirban Majumder and Nisheeth Shrivastava. Know your personalization: Learning topic level personalization in online services. In *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013.

[26] Delfina Malandrino, Andrea Petta, Vittorio Scarano, Luigi Serra, Raffaele Spinelli, and Balachander Krishnamurthy. Privacy awareness about information leakage: Who knows what about me? In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM, 2013.

[27] Aaron K Massey and Annie I Antón. Behavioral

advertising ethics. *Information Assurance and Security Ethics in Complex Systems: Interdisciplinary Perspectives*, 2010.

[28] Dana Mattioli. On Orbitz, Mac users steered to pricier hotels. `http://online.wsj.com/news/articles/SB10001424052702304458604577488822667325882`, 2012.

[29] Jonathan Mayer. Tracking the trackers: Self-help tools. `https://cyberlaw.stanford.edu/blog/2011/09/tracking-trackers-self-help-tools`, 2011.

[30] Jonathan Mayer. Tracking the trackers: Where everybody knows your username. `https://cyberlaw.stanford.edu/blog/2011/10/tracking-trackers-where-everybody-knows-your-username`, 2011.

[31] Jonathan R Mayer and John C Mitchell. Third-party web tracking: Policy and technology. In *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012.

[32] Aleecia M McDonald and Lorrie Faith Cranor. Americans' attitudes about internet behavioral advertising practices. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*. ACM, 2010.

[33] Aleecia M McDonald and Lorrie Faith Cranor. Survey of the use of Adobe Flash Local Shared Objects to respawn HTTP cookies, a. *ISJLP*, 7, 2011.

[34] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 2012.

[35] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Crowd-assisted search for price discrimination in e-commerce: first results. *arXiv preprint arXiv:1307.4531*, 2013.

[36] Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and J Niels Rosenquist. Understanding the demographics of Twitter users. *ICWSM*, 11, 2011.

[37] Nick Nikiforakis, Wouter Joosen, and Benjamin Livshits. Privaricator: Deceiving fingerprinters with little white lies.

[38] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013.

[39] Andrew Odlyzko. Privacy, economics, and price discrimination on the Internet. In *ICEC2003: Fifth International Conference on Electronic Commerce*, 2003.

[40] Lukasz Olejnik, Tran Minh-Dung, Claude Castelluccia, et al. Selling off privacy at auction. 2013.

[41] Eli Pariser. Beware online" filter bubbles"(ted talk), 2011.

[42] Eli Pariser. *The filter bubble: How the new personalized Web is changing what we read and how we think*. Penguin, 2011.

[43] Damon Layton Petta and Bradley Keith Mohs. Systems and methods to control web scraping, 2013. US Patent App. 14/061,633.

[44] Dillon Reisman, Steven Englehardt, Christian Eubank, Peter Zimmerman, and Arvind Narayanan. Cookies that give you away: Evaluating the surveillance implications of web tracking. Manuscript, 2014.

[45] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *9th USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[46] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

[47] Latanya Sweeney. Discrimination in online ad delivery. *Queue*, 11(3), 2013.

[48] Neil Thurman and Steve Schifferes. The future of personalization at news websites: lessons from a longitudinal study. *Journalism Studies*, 13(5-6), 2012.

[49] Michael Carl Tschantz, Amit Datta, Anupam Datta, and Jeannette M Wing. Information flow experiments. 2nd International Workshop on Accountability: Science, Technology and Policy, 2014. Extended Abstract.

[50] Michael Carl Tschantz, Amit Datta, Anupam Datta, and Jeannette M. Wing. A methodology for information flow experiments. Technical Report arXiv:1405.2376, ArXiv, 2014.

[51] Joseph Turow, Jennifer King, Chris Jay Hoofnatle, Amy Bleakley, and Michael Hennessy. Americans reject tailored advertising and three activities that enable it. *Departmental Papers (ASC)*, 2009.

[52] Blase Ur, Pedro Giovanni Leon, Lorrie Faith Cranor, Richard Shay, and Yang Wang. Smart, useful, scary, creepy: perceptions of online behavioral advertising. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*. ACM, 2012.

[53] Jennifer Valentino-Devries, Jeremy Singer-Vine, and Ashkan Soltani. Websites vary prices, deals based on users' information. `http://online.wsj.com/news/articles/SB10001424127887323777204578189391813881534`, 2012.

[54] Steven L Willborn. Disparate impact model of discrimination: Theory and limits, the. *Am. UL Rev.*, 34, 1984.

[55] Craig E Wills and Can Tatar. Understanding what they do with what they know. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. ACM, 2012.

[56] Xinyu Xing, Wei Meng, Dan Doozan, Nick Feamster,

Wenke Lee, and Alex C Snoeren. Exposing inconsistent web search results with Bobble.

APPENDIX

A. MACHINE LEARNING AND FAIRNESS

Determining if a personalization algorithm that uses machine learning is "fair," i.e., ethical, raises unresolved conceptual and empirical questions. For example, suppose that we agree that targeting/personalization by race is unfair. But what if the personalization algorithm uses a latent factor model and one of the latent factors that gets learned is strongly correlated with race? In other words, a sufficiently complex personalization system may behave as if it discriminates against a certain group of users, even if such behavior was not intentionally programmed into the system. In fact, this phenomenon does happen in practice and was the subject of one of the studies we analyzed [47].

Typically the personalization system is a black box, and the experimenter must be content with uncover such relationships, and will not be able to ascribe intent. Even this is tricky, in the absence of a hypothesis for just *how* the sensitive attribute affects the observed output. Here we present a generic methodology for this task.

We assume that we are able to generate training and control browser instances that simulate users who differ only in the attribute of interest X (in the study cited above, this translated to generating names that are highly statistically correlated to white and black individuals respectively). Then we follow the paradigm considered throughout this paper, and carry out the training and measurement phases of the experiment, in each case generating a vector of observables Y (for example, a set of ads seen). Finally, we frame our task as a learning problem to see if X can be learned from Y. If the learning task succeeds with an AUC of over 50% (for a binary variable), we can assert that the personalization system behaves as if it personalizes on the basis of the sensitive attribute X. This methodology is a generalization of the one considered in [16].

B. PUBLISHER CATEGORIES STUDIED

The full list of categories by publisher included within our study is included below.

- **Fox News** - Sport; Politics; Religion and Belief; Unrest, Conflicts and War; Disaster and Accident
- **CNN** - Religion and Belief; Politics; Sport; Disaster and Accident; Unrest, Conflicts, and War
- **Daily Mail** - Sport; Religion and Belief; Arts, Culture, and Entertainment; Health; Human Interest
- **BBC** - Sport; Politics; Unrest, Conflicts, and War; Economy, Business, and Finance; Lifestyle and Leisure
- **The Guardian** - Arts, Culture, and Entertainment; Economy, Business, and Finance; Religion and Belief; Sport; Politics
- **NBC** - Politics; Unrest, Conflicts, and War; Economy, Business, and Finance; Disaster and Accident; Religion and Belief
- **Time** - Politics; Arts, Culture, and Entertainment; Economy, Business, and Finance; Religion and Belief; Health
- **USA Today** - Politics; Economy, Business, and Finance; Unrest, Conflicts, and War; Religion and Belief; Health
- **Huffington Post** - Religion and Belief; Health; Arts, Culture, and Entertainment; Politics; Sport

C. ENGINEERING CHALLENGES

In our own experimental work and in discussions with other researchers, we were struck by how often we face similar engineering problems in getting these measurements to work and to scale. Published studies emphasize the design and the results, omitting the "tricks-of-the-trade".

To surface knowledge of these common pitfalls and challenges, we reached out to nine groups of researchers from our survey. All responded. Since many publications share one or more authors, this represents the majority of studies that use browser automation. Even though we asked for "a paragraph or two," to our surprise we received quite detailed responses without exception. A few respondents expressed gratitude for our effort, and one even described the opportunity to write about these engineering pitfalls as "cathartic."

Browser automation. Several groups experienced crashes and bugs with Selenium. One group was forced to abandon using MSIE with Selenium because of crashing issues. Selenium has several other issues including difficulty of obtaining third-party cookies in earlier versions. Another near-universal difficulty is not being able to know when a page is fully loaded; the typical solution is to wait some number of seconds and hope that it is sufficient. Difficulties with getting Selenium to run headless were also encountered.

Instrumentation and data extraction. Here the challenges are more task-specific since they relate to data extraction from web pages. A frequent problem is generating stable references to page elements, given that page templates are dynamic and often change. When <id> tags aren't available, XPaths need to be used, but these require repeated manual re-engineering. Some instrumentation tools had limitations and had difficulty examining nested iframes. Generally researchers used more heuristic than principled approaches. One group even tried screen captures with human coders, but this failed because screen captures were often blank when running via Selenium.

Due to automation and instrumentation difficulties, many studies that had started out in an ambitious manner ended up focusing on a small number of browser/client configurations or websites.

Request bottlenecks and API limitations. Multiple groups ran into rate limits for different websites and saw their IPs blacklisted, and so had to throttle their crawling to stay under these limits. They also distributed their infrastructure between different IPs for this reason, and/or used Tor. Many APIs also have rate limits, and in addition were unreliable so researchers found web scraping to be a better option.

Ethical concerns. In addition to the ethics of potential ToS violations inherent in web crawling, researchers pointed out a number of ethical and legal issues. Many studies on the ad ecosystem face the limitation of not being able to access ad landing pages since that would constitute click fraud. Inferring information about the ad without doing this is difficult. One group had the workaround of feeding URLs to Google AdWords and looking at the keywords returned.

This is a legal gray zone since it forces Google to click on the ad. Crowdsourced studies were limited by privacy concerns of participants. While it is tempting to over-collect instrumentation data and filter it on the server, this would violate privacy.

D. CONTEXTUAL RECOMMENDATION RESULTS

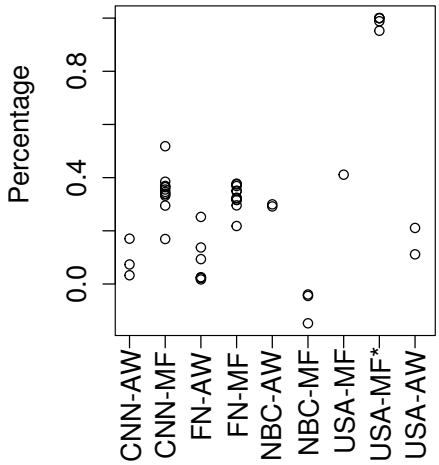

Figure 3: Contextual Recommendation
MF = More From [Publisher], AW = Around the Web
USA-MF* is an extra More From [Publisher] box which recommends a single link

This plot shows the percentage change in the number of articles matching the context of the host article over the average number of articles of that category seen in all host articles. Each point represents a category of a publisher and each column is a publisher, location combination.

E. NEWS PERSONALIZATION RESULTS

Loc.	Pub/Cat	Control	Trained(Change)	P-Value
AW	CNN/R	257/2281	390/2266(+5.9%)	< 0.01
	DM/E	322/1463	179/1439(-9.6%)	< 0.01
	DM/H	561/1463	653/1334(+10.6%)	< 0.01
	FN/S	763/2263	846/2236(+4.1%)	< 0.01
	NBC/R	74/583	95/585(+3.5%)	< 0.05
	GUA/R	48/322	63/317(+5.0%)	< 0.05
	TIME/H	92/289	58/279(-11.0%)	< 0.01
	FN/R	516/2409	428/2404(-3.6%)	< 0.01
MF	NBC/F	69/566	149/567(+14.1%)	< 0.01
	TIME/H	108/966	70/871(-3.1%)	< 0.01
	TIME/P	378/966	178/895(-19.2%)	< 0.01

Table 2: News Site Personalization

Change in the count of the number of article links matching the browsing profile compared to the amount seen by the control.

Publishers: DM - Daily Mail, FN - Fox News, NBC - NBC News, GUA - The Guardian, TIME - Time

Categories: R - Religion and Belief, E - Arts, Culture, and Entertainment, H - Health, S - Sport, F - Economy, Business, and Finance, P - Politics

F. OVERVIEW OF STUDIES ANALYZED

	Targets			Infrastructure		Variable						Experimental	
Paper	Data collection	Data flow	Data use	Browser / Automation[a]	Instrumentation	Distributed	Location	User-agent	Demographics	Interest categories	Privacy Tools	Real history	Page load time	Scale
Leakage of PII via OSN ('09) [22]		PII leaks		M*	Live HTTP Headers									
Privacy diffusion on the web ('09) [21]	Cookies			F,PS	Proxy									1.2K sites
Challenges in measuring ('10) [18]			Ads		Proxy		•		•					730 total queries
Flash cookies and privacy ('10) [46]	Cookies, local shared objects			M*										100 sites
Privacy leakage in mOSN ('10) [23]		PII leaks		M*	Proxy									
Flash cookies and privacy II ('11) [8]	Cookies, local shared objects			M*										100 sites
Privacy leakage vs. protection measures ('11) [20]		PII leaks		M*	Proxy									10 sites
Respawn HTTP Cookies ('11) [33]	Cookies, local shared objects			UA*			•						60s.	600 sites
Self-help tools ('11)[29]	Cookies			UA*	FourthParty						•			500 sites
Where everybody knows your username ('11) [30]		PII leaks		M*	FourthParty				•					185 sites
Detecting and defending against third-party tracking ('12) [45]	Cookies			FF, TT	TrackingTracker									2K sites
Detecting price and search discrimination ('12) [34]			Price descrimination	SA, CH, IE, JS	Proxy	•	•	•	•	•			2m	200 sites
Mac users steered to pricier hotels ('12) [28]			Prices					•						
Measuring the effectiveness of privacy tools ('12) [9]			Ads	F, SL							•			
Message machine ('12) [1]			Emails				•		•					$\sim$ 3K emails
Websites vary prices ('12) [53]			Prices				•							
What they do with what they know ('12) [55]			Ads		Proxy									10 days
AdReveal ('13) [24]	Cookies		Ads		Proxy, Ghostery				•					103K sites
Cookieless monster ('13) [38]	Fingerprinting													10K sites
Crowd-assisted search ('13) [35]			Price descrimination	F, CH	Custom plugin	•	•		•		•			600 sites
Discrimination in online ad delivery ('13) [47]			Ads	M, UA			•	•						2184 names
FPDetective ('13) [5]	Fingerprinting, JavaScript			CR, SL, CJ, PJ	Proxy, Browser Code								10-15s.	1M sites
Know your personalization ('13) [25]			Search results		Custom plugin				•		•			5K queries
Measuring personalization of web search ('13) [19]			Search results	PJ		•		•						120 queries
Privacy awareness about information leakage ('13) [26]		PII leaks		F, PS, SL					•		•	•		1.5K sites
Selling off privacy at auction ('13) [40]	Cookies	Cookie sync.	Bid prices	F, SL		•	•			•		•	5s.	5K sites
Shining the floodlights ('13) [14]	Cookies, JavaScript			F, JS	FourthParty			•					30s.	500 sites
Statistical approach ('13) [16]	General tracking			F, PY	FourthParty						•		10s.	2K sites
Adscape ('14) [10]			Ads	F, SL	Custom plugin				•					10K sites
Bobble ('14) [56]			Search results	CH, SL	Custom plugin	•	•	•				•		1K queries
Information flow experiments ('14) [50]	Cookie		Ads	F, SL	Proxy				•				1m.	
Third-party OSN applications ('14) [12]		PII leaks		F, SL	FourthParty				•					997 applications

[a]FF = Firefox, CH = Chrome, CR = Chromium, IE = Internet Explorer, SA = Safari, SL = Selenium, JS = JavaScript, PJ = PhantomJS, PS = PageStats, PY = Python, TT = TrackingTracker, CJ = CasperJS, UA = Unknown automation, M = manual, Asterisk = inferred

Table 3: Overview of prior measurement studies