OpenWPM: An automated platform for web privacy measurement

Draft — March 15, 2015

Steven Englehardt, Chris Eubank, Peter Zimmerman, Dillon Reisman, Arvind Narayanan
Princeton University
{ste,cge,peterz,dreisman,arvindn}@cs.princeton.edu

ABSTRACT

Web measurement techniques have been highly influential in online privacy debates and have brought transparency to the online tracking ecosystem. Due to its complexity, however, web privacy measurement remains a specialized research field. Our aim in this work is transform it into a widely available tool.

First, we analyze over 30 web privacy measurement studies, identify several methodological challenges for the experimenter, and discuss how to address them. Next, we present the design and implementation of OpenWPM, a flexible, modular web privacy measurement platform that can handle any experiment that maps to a general framework. It supports parallelism for speed and scale, automatic recovery from failures of the underlying browser, and realistic simulation of users. OpenWPM is open-source[1] and has already been used as the basis of several published studies on web privacy and security. We show how our generic platform provides a common foundation for these diverse experiments, including a new study on the "filter bubble" which we present here.

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, web privacy measurement has repeatedly forced companies to improve their privacy practices due to public pressure, press coverage, and regulatory action [8, 15]. Meanwhile, the scope of web privacy measurement has expanded beyond simply uncovering tracking to uncovering how personal data is used. Specifically researchers have characterized various types of tracking-based personalization: price discrimination and steering [27], behaviorally targeted advertising [34], search results [61], and more.

On the other hand, web privacy measurement (hereafter, WPM) has several challenges, making it difficult for newcomers. First, there are many methodological pitfalls such as the difficulty of rigorously establishing a causal relationship between user actions, such as visiting certain websites, and a later observation, such as being shown a certain ad. Solutions to these difficulties are often "folk knowledge". Second, building the technical infrastructure for WPM — using automated browsers to simulate users browsing the web and extracting measurements of interest — is a significant research and engineering challenge. As a result **there is no usable, generic platform for WPM**, and it has been largely confined to a niche community of researchers.

At the same time, there has been increasing demand from many groups outside academic researchers to be able to do studies on online privacy: regulators, self-regulators, the press, activists, and even the general public. Such broader participation is necessary to achieve effective, continual oversight and transparency of web privacy practices. After all, researchers typically only conduct a study if there are new methodologies to be demonstrated.

Our contributions. Put simply, we seek to transform web privacy measurement from a niche research field to a widely available tool.

- We identify the experimental aspects shared by all WPM studies and identify over 30 that fit our framework (Section 2). Based on a thorough methodological analysis, we identify seven challenges. We synthesize the best practices and make recommendations based on prior works as well as our own findings and experience (Section 3). We identify cases where methodological challenges remain unresolved, leading to avenues for future work. We seek to make this a methodological guidebook for future studies.
- Our second and key contribution is OpenWPM, a flexible web privacy measurement platform (Section 4). The python-based platform is designed to give researchers the ability to launch reliable measurements with real browsers at web-scale. OpenWPM has been designed around stability, modularity, and scalability; providing instrumented, extensible browsers as isolated instances. We have incorporated solutions to many of the practical challenges in scaling real browser automation and measurement. OpenWPM exposes a scriptable command-line API to researchers, allowing them to launch parallel measurement instances and drive them using a high-level language. The platform is bundled with browser and network level instrumentation and can be expanded through the browser plugin interface to include custom or pre-existing instrumentation.

[1] `https://github.com/citp/OpenWPM`

- OpenWPM has been used for six completed studies so far, both by us and other researchers [51, 6, 50, 28], and several others are in progress, both at our institution and elsewhere. The studies encompass tracking, personalization, and web security, and several of them have already led to improvements and fixes to privacy and security. These experiments have crucially benefited from several advanced features of OpenWPM such as the ability to automatically log into websites using specified credentials. Using these case studies, we show in detail how OpenWPM's capabilities enable quickly designing and running an experiment.
- Two of these six studies have not previously been reported. 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. Based on experiments on nine major publisher sites, we find that only some sites perform any history-based personalization at all, and when they do, the effect sizes are very small. Thus, our results contradict the narrative of pervasive personalization leading users into feedback loops of reinforcing viewpoints and ideological "bubbles."

  Our final study is on security and authentication practices. Contrary to the best practice of using IP address, browser fingerprints, and other such features to trigger password re-entry, we demonstrate through measurement that on most websites, an attacker who steals authentication cookies is in a position to unconditionally impersonate the user.

In developing OpenWPM, our vision for the future of the web is one where tracking and personalization are readily apparent, thus aligning incentives between publishers, third parties, and consumers. The most egregious privacy violations will be quickly detected based on measurement and blocked by browser privacy tools. Finally, measurement results will crucially inform the vigorous ethical and policy debates on tracking, surveillance, and data-driven personalization that have been playing out in Washington, D.C. and around the world.

2. BACKGROUND AND RELATED WORK

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.

Personalization. The primary driver of online tracking is behavioral advertising — showing ads based on the user's past behavior, especially browsing and search history. From a technical perspective, behavioral advertising is an instance of personalization of content based on user attributes or behavior. Many other practices that raise privacy or ethical concerns fall under this umbrella: *price discrimination*, where a site shows different prices to different consumers for the same product [27]; *steering*, a gentler form of price discrimination where a product search shows differently-priced results for different users [37]; and 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. The worry is that users will be shown articles reinforcing their existing views, thus isolating them into ideological "bubbles".

Related work: web privacy measurement. There are two main ways to collect large-scale data for WPM experiments: crowd-sourcing and simulating users, i.e., running bots. Our focus is on the latter type, but there are many similarities to the two types of studies. We exhaustively surveyed both types of studies, and found over 30. They are summarized in Table 2.

We see three types of measurement targets: (1) tracking, whether active (cookies, Local Shared Objects, etc.) or passive (fingerprinting); (2) data flow between trackers — leakage identifying information and sensitive attributes, and exchange of pseudonymous cookies (cookie syncing) and (3) various types of personalization and targeting — contents of ads, bid prices of ads, prices of products, listings of products, and search results. A variety of technical tools were used for measurement. Unfortunately, there was a great deal of duplication of effort, which served as an additional motivation for our platform. Researchers tested the variation of tracking and personalization based on a number of variables: location, type of browser (user-agent), demographics, behavioral interest categories, and use of privacy tools.

Related work: WPM platforms. An infrastructure for automated web privacy measurement has three components: (i) input — simulating users who visit and interact with web sites and services, (ii) output — recording how these sites treat the user (tracking, content personalization, etc.) and (iii) analysis — using the recorded data to determine correlations or other conclusions. We briefly discuss how OpenWPM compares to other platforms that have been built. We consider a tool to be a platform if is is publicly available and there is a some generality to the types of studies that can be performed using it.

FourthParty is a Firefox plug-in for instrumentation and does not handle automation [40]. OpenWPM provides different options for instrumentation (Section 4). *TrackingObserver* is a Chrome extension that detects tracking and exposes APIs for extending its functionality such as measurement and blocking [52].

XRay is a platform for *differential correlation*: inferring input-output relationships in any personalized web service [33]. XRay handles the analysis phase of WPM in a generic way, but not driving the browser or instrumentation. This is precisely the converse of OpenWPM, suggesting the exciting possibility of using the two tools in concert to achieve an even greater degree of generic automation.

AdFisher is a tool for running automated experiments on personalized ad settings [16]. It contains a barebones automation framework with similar components as ours (Sele-

nium, xvfb), but the key technology is a machine-learning system for causality attribution. Again there is the possibility of running OpenWPM's automation and instrumentation together with AdFisher's analytic component.

Several research groups have built or deployed crowd-sourcing platforms for WPM, including $heriff and Bobble [61, 42]. Some challenges here include providing value to users to incentivize participation, participant privacy, etc.

Web security measurement often uses similar methods as WPM, and the boundary is not always clear. Yue and Wang modified the Firefox browser source code in order to perform a measurement of insecure Javascript implementations on the web [62] . Nikiforakis et al. utilized a headless browser to measure the amount of third-party JaveScript inclusions across many popular sites and the vulnerabilities that arise from how the script is embedded [45]. Zarras et al. used Selenium to drive crawls that measured and categorized malicious advertisements displayed while browsing popular sites [63]. Other studies have analyzed Flash and JavaScript elements of webpages to measure security vulnerabilities and error-prone implementations [58, 48].

3. METHODOLOGICAL CHALLENGES

We present a set of methodological challenges in designing WPM experiments and recommendations for addressing them. This list is based on an analysis of over 30 web privacy measurement studies that met our criteria (Table 2) and interviews with the authors of several of these studies. These methods are necessary to ensure valid and accurate results.

3.1 Statistical rigor

When making inferences about personalization (whether of ads, prices, search results), statistical rigor is paramount. There are four levels of inferences one can make: detecting personalization, measuring associations, inferring causality, and quantifying effect sizes. We illustrate with an example. A crowd-sourced study of an online retailer might find that some users see higher prices for the same product than others (see below for how to rule out the effect of A/B testing). Further analysis might reveal being logged in is correlated with higher prices. However, at this stage we cannot rule out a spurious association due to a *confound*: perhaps users on desktops (vs. mobile) are shown higher prices and are also more likely to log in. Further testing using simulated users and randomized, controlled experiments reveals that logging in causes displayed prices to increase. Finally, increasing the sample size of the experiment allows us to estimate the average price increase due to logging in as 10% $\pm$ 2%.

As this example shows, *crowd-sourcing is ideal for detecting the existence of personalization and, with some care, for detecting associations. Randomized experiments are essential for causal inference*, but even so, they are methodologically tricky, as we now discuss.

Handling confounds. 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 [26]. 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. The normal load-balancing architecture of these systems hides the identity of the node from the client, necessitating workarounds.

There is a long history and extremely well-developed theory of causal inference in statistics [23, 5]. The general principle for avoiding confounds is *control what you can, randomize what you can't.* 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 [42]. 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 [42], they blocked the referer header, 3rd party cookies, flash cookies and other stateful trackers. Hannak et al. additionally controlled for browser fingerprints [27].

A/B testing is a tricky confound. Suppose we wish to test if a privacy tool has an impact on targeted advertising, so we create a "treatment unit" which has the privacy tool and a "control unit" which has no privacy tool. Unluckily for us, the ad network assigns the treatment unit to bucket 'A' which receives no targeted advertising and the control unit to bucket 'B' which does. Then we will conclude that the privacy tool prevents targeted advertising, even if it in fact has no effect. The solution, of course, is to average across multiple treatment and control units. To summarize, *we recommend that the experimenter exhaustively enumerate all directly or indirectly observable client-side attributes and either control or randomize them, and further, ensure adequate sample sizes of each experimental condition.*

Ensuring non-interference. It is important to ensure that the activities carried out by one experimental unit (browser instance) do not affect other units. This is the *non-interference* assumption. Tschantz et al. provide evidence of interference, which they term "cross-unit effects", in the measurement of the diversity of Google text ads [56] when different browser instances are running on the same machine. To mitigate such effects, we must ensure that different experimental units are in fact recognized as different by the service. One way to do this is to randomize browser fingerprints and other pseudo-identifiers across experimental units. Thus, while the traditional theory elevates controlling variables over randomization, in the WPM context randomization is sometimes the better choice.

3.2 Realistic user simulation

Often the simulated users must mimic the browsing patters of different demographics of real users. A study of price discrimination modeled histories of "affluent" and "budget" shoppers based on Alexa categories [42] and numerous studies simulated browsing histories of users with topical interests [42, 49]. Other choices include the Quantcast list of top sites with demographic breakdowns [26] and Google search

results for topical keywords [42].

We must note a prominent limitation: none of the studies obtained real users' browsing histories for controlled experimentation. This limits realism to a degree. *Search* history on a small scale was obtained and studied in Majumder and Shrivastava, [35]. Another study simulated users issuing search queries (to Bing) of pseudonymous users in the leaked AOL search log dataset, followed by visiting the top 5 results returned by Bing. This is arguably an interesting way to obtain an approximation to real browsing history [34].

Studies outside the web privacy measurement community have used various datasets of real user logs for studying other questions [24, 21, 4]. We attempted to obtain these datasets, but they were either no longer available or prohibitively expensive. As yet there is no ideal solution here.

3.3 Practical and ethical challenges

Full-fledged vs. stripped down browsers. We discuss this issue in Section 4. We recommend that researchers use full-fledged automated browsers; we have anecdotal reports of differences in behavior when using stripped-down browsers such as PhantomJS. If a stripped-down browser is necessary, we recommend explicit checks to see if it accurately mimics a full-fledged one.

APIs vs. content extraction. Authors, such as Hannak, et. al. [26], have reported that using APIs, even when available, is problematic because of rate limits or because they don't return the same results as the web interface. Therefore scraping content from web interfaces is usually necessary. This imposes several practical challenges: it is buggy and scales poorly across different websites. Our content extraction module (Section 4) is a step toward mitigating this problem.

Ethical challenges. Two sets of ethical challenges are frequently reported: collecting personal information in crowdsourcing studies and the ethics of running bots. The former is a well-understood privacy challenge, and typically handled by a combination of informed consent, data minimization, and anonymization. For the latter, researchers have universally avoided clicking on ads and have sometimes sought IRB approval. However, even viewing ads may cost advertisers money, and not being able to click on ads limits the kinds of experiments that can be done.

Since greater transparency is in the interest of companies that wish to compete on privacy, developing a mutually acceptable ethical protocol for experiments would be a huge step forward. For example, researchers could execute unfettered experiments on services that have opted into this protocol, and after the fact, submit a list of cookies/fingerprints that belong to their bots, so that ad companies can discount those clicks from their click reporting.

4. MEASUREMENT PLATFORM

An infrastructure for automated web privacy measurement has three components: simulating users, recording outputs (tracking, content personalization, etc.), and analysis. We set out to build a platform that can automate the first two components and can ease the researcher's analysis task. We sought to make OpenWPM general, modular, and scalable enough to support essentially any WPM experiment. This core vision translated into a few key design requirements.

4.1 Requirements

4.1.1 Browser and driver

We considered 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 three categories of choices are HTTP libraries like curl or wget, lightweight browsers like PhantomJS (an implementation of WebKit), and full-fledged browsers like Firefox and Chrome.

Realism vs. performance and stability. HTTP libraries are clearly unsuitable for WPM — they do not support basic browser functionality such as JavaScript. Lightweight browsers are stable and were designed to have low overhead. While they might be suitable for an individual study, a generic platform must be able to accurately reproduce the range of functionality of a full browser. Lightweight browsers don't support plugins and may have subtle differences arising from non-standard rendering and JavaScript engines.

Instead, we chose to use Selenium, a cross-platform web driver for Firefox, Chrome, Internet Explorer, and PhantomJS. Selenium provides a common interface to drive all major browsers, allowing studies to fully reproduce the conditions of a real user browsing. Since it is ultimately an unaltered consumer browsing visiting each page, all technologies that a typical user would support (e.g., HTML5 storage options, Adobe Flash) are also supported by measurement instances. A key disadvantage of Selenium is that Selenium frequently crashes or hangs indefinitely due to its blocking API [9], as it was designed to be a tool for webmasters to test their own sites rather than an engine for automated crawls. To fix this, we need to take additional steps to ensure stability and scalability.

> **Design Requirement: 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.

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 or showing only pro bono ads.

> **Design Requirement: 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 pages. Websites detecting automated browsers may act in a pathological manner, thus weakening experimental results.

Interacting with the platform. Selenium provides a complex API for interacting with sites and parsing web content. However, the researcher is typically interested in higher-level commands, essentially mimicking interactions a user is likely to take as they browse the web. Examples include finding the login page of a site or finding all of the links or search results on the page.

Design Requirement: Abstraction. 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 and limits code duplication for scripts used to drive experiments.

4.1.2 Browser instrumentation

When running studies, there is a need to collect outputs from various locations in the browser or from the visited websites. *Instrumentation* can be done in one of three ways: modifying the browser source code directly, extending the browser with an extension, or wrapping a data collection layer around the browser (for example, with a proxy).

Instrumentation choices. A popular instrumentation tool is FourthParty [40] which has been used in several studies [39, 38, 14, 19, 22]. Other studies used custom plug-ins [34, 43, 49]. The advantage of plugin-based instrumentation is that the researcher has access to the javascript running on the page, which can be used to both make measurements and interact with the page.

Researchers can also use a *proxy* instead of or in addition to browser instrumentation. We have seen past work use Mitmproxy and Fiddler, although custom proxies were also seen [26, 42]. The benefit of using a proxy for instrumentation is that it has the ability to intercept all network traffic into and out of the browser, as well as the ability to alter traffic after it leaves the browser. For example, in Mikians et al [42] 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.

Finally, researchers can modify the source-code of browsers directly and perform measurements using the custom browser [6, 7]. The has the benefit of complete access to all browser calls and functions beyond what one would achieve through a browser extension in JavaScript.

Design Requirement: Compatibility & Modularity. The platform must support any instrumentation option the researcher prefers. It should also easily interface with existing instrumented browsers and browser extensions. The system should be modular, allowing the researcher to swap out instrumentation components between studies or in the course of a single study.

4.1.3 Platform Requirements

In addition to the browser, driver, and instrumentation requirements, there are platform requirements that the entire system should satisfy.

We found that many prior studies deployed distributed versions of their crawlers — either to increase scale or to measure personalization based on geographic location by deploying crawlers to PlanetLab [61, 26, 42, 49].

Design Requirement: Scalability. The primary advantage of browser automation is that it enables researchers to repeatedly visit sites at a rate infeasible for humans. The platform should allow increasing speed through browser parallelization and the ability for different instances to write to the same database.

Finally, easy repeatability is important for several reasons. Research studies are typically conducted in a "one-off" fashion. For example, "evercookies" were at the center of fierce debates when discovered in use 2010 [53], resulting in a lawsuit and a $500,000 settlement [17]. However, a 2014 found numerous websites using these tracking technologies[6]. Second, enforcement agencies need to be able to easily reproduce research findings in order to be able to take action [20].

Design Requirement: Repeatability. To promote scientific rigor, the platform should enable researchers to easily reproduce experiments. It should also log data 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. In doing so, we have also created a high-level language for issuing crawl commands which can be extended as researchers add custom commands. The entire platform is built using Python and Python-compatible libraries.

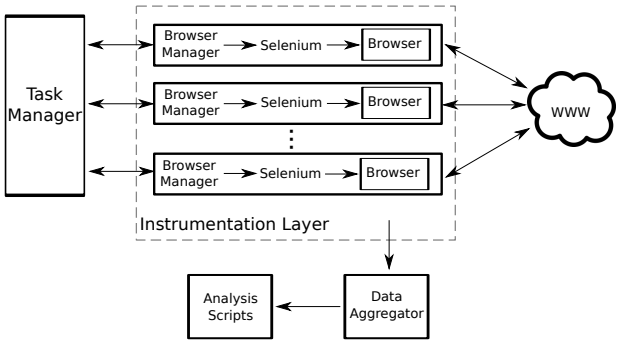

Figure 1: High-level overview of OpenWPM
The task manager monitors browser managers, which convert high-level commands into automated browser actions. The data aggregator receives and pre-processes data from instrumentation

Browser managers. Requiring browser managers as an abstraction layer stems from our choice to use Selenium for automation. As discussed in Section 4.1.1, Selenium is susceptible to frequent crashing and hanging. As such, one of

our design requirements is to isolate Selenium from the rest of the components of the platform.

Each browser manager instantiates a Selenium instance with a specified configuration of user preferences (e.g. turning on Do Not Track). It is responsible for converting high-level platform commands (e.g. visiting a site) into specific Selenium subroutines. The use of a separate manager for each browser enables us to save per-browser preferences and state which stay consistent between any browser failures. Since each browser manager exists within it own process, this abstraction layer also protects other platform components from browser failures. All Selenium-specific code is contained in the browser manager, making a future conversion to a different web-driver easier if it is ever necessary.

Browser managers have the ability to launch measurement instances in a headless container. Full browsers have no headless option, as they are built for graphical user interaction. To solve this problem, we leverage pyvirtualdisplay to interface with Xvfb, which draws the graphical interface of the browser to a virtual frame buffer. Headless crawls allow greater parallelization due to lower memory consumption, deploying crawls to remote machines, and easily generating screenshots of rendered sites when necessary.

Task manager. The task manager provides a scriptable command-line interface for controlling multiple browsers simultaneously during a study. It allows the researcher to distribute commands to browser managers while transparently abstracting away any stability code or error handling. 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).

The Task Manager API exposes four options for issuing commands to multiple browser managers: either on a first-come-first serve basis, or to individual browser managers or to all browsers. In the latter case there is the option of executing them synchronously or asynchronously. This allows the researcher to easily launch multi-browser measurements that scale to the available system resources by only specifying the number of parallel instances to run.

The task manager achieves this by launching a per-browser command execution thread each time the researcher issues a command to that particular browser. This thread is given a specific lifetime and is set to monitor the state of the corresponding browser manager process. If the thread either times out or sees that the browser manager process has ended, it enters a crash recovery mode where it:

1. Dumps all browser local state to a temporary location
2. Kills any remaining processes related to that browser manager (the browser process, the Xvfb process, and any custom per-browser processes the researcher launched)
3. Initializes a new browser manager with all of the pre-crash configuration variables
4. Loads the browser state from the temporary location
5. Calls the browser manager to restart a measurement instance with that state

Data aggregator. 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. Access to the data aggregator is over a socket interface which can easily be connected to from any number of browser managers or instrumentation processes. The data aggregator serializes data from an arbitrary number of sources without the need for database access on a per-source basis. Furthermore, isolating the data aggregator behind a network interface ensures that browser crashes in other processes do not corrupt any data or deny access to other data collecting processes.

In comparison to client-server databases, local SQLite databases do not require end-user setup of a database server and enable researchers to easily share data. However, socket connections to the data aggregator easily provide support for other frameworks such as MySQL or cloud hosted databases as studies grow in scale.

Instrumentation. We evaluated different instrumentation options, including: FourthParty, several proxies, and custom plugins. As described in Section 4.1.2 each instrumentation option has positives and negatives depending on the research application. We settled on including Mitmproxy as the primary means of instrumenting the browser, as nearly all of our studies have required monitoring what is transmitted through network traffic. We also built a custom browser plugin for JavaScript-based instrumentation as a fallback.

Workflow. As an example workflow, the researcher issues a command to the task manager and specifies that it should synchronously execute on all browser managers. The task manager checks all of the command execution threads and blocks until all browsers are available to execute a new command. Then it creates new command execution threads for all browsers and sends the command and command parameters over the process boundary to the browser manager. The browser manager interprets this command and runs the necessary Selenium code to execute the command in the browser. Once the Selenium code returns, the browser manager can send returned data (e.g. the parsed contents of a page) to the data aggregator. Similarly, any browser instrumentation tools, such as mitmproxy, are continually sending data to the data aggregator. Once the command is complete, the browser manager notifies the task manager that it is ready for a new command.

4.3 Advanced capabilities

Automatic login. Studies which require a measurement of data flow under an authenticated session would benefit greatly from the ability automatically login to sites. We have made progress implementing an automated login utilizing Facebook's federated login infrastructure (formerly called Facebook Connect, now Facebook Login). We have used this to successfully login to half of the top-200 Alexa sites that support user accounts. The capability is useful for studies which do not require a full coverage of logins but rather simply require access to some sample of sites. We describe a sample study which uses this capability in Section 5.

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. The inclusion of additional countermeasures is reserved for future work.

Browser fingerprinting mitigation. For researchers running multiple browsers from the same machine, experiments may suffer from interference 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 [46] note that randomizing user-agent can cause a rendering issues, adopting their suggestion of subtle variations in properties implemented at the browser-level is one future direction in fingerprint mitigation.

Tracking cookie detection. Differentiating identifying cookies from non-identifying cookies is essential to many studies aiming to quantify the level of tracking. The simplest approach is to treat cookie values that are different in two different crawls as unique [41]. Olejnik et al. filter out cookies with values shorter than 10 characters [49]. The detection method developed as part of one of our previous studies [6] has been adapted and included as a postprocessing step of OpenWPM.

Fine-grained profiles. We support programmatic access to browser preferences and storage vectors (such as HTTP cookies, HTML5 LocalStorage, HTML5 Indexed DB and Adobe Flash objects) during a crawl. This fine-grained control is useful for reverse engineering website behavior. We also provide a simple interface for saving and loading these profiles between crawls to enable simulating a specific user over longer studies.

Content extraction. In many types of personalization studies, the content to be extracted or measured consists of search results, product listings, or other such lists. Structurally, elements of these lists are *subtrees of the page DOM tree that have a high pairwise structural similarity.* We exploited this observation to create a clustering algorithm that automatically extracts the list with the greatest amount of text from any given page or container in an unsupervised fashion. When run on a Google or Bing search results page, it extracts the search results. When run on say reddit.com, it extracts the main list of articles. This module frees the researcher from the need to write site-specific DOM parsing code for extracting lists.

Additionally, OpenWPM can automatically infer and log the Xpath of located elements (such as buttons) or extracted content on websites, allowing tasks such as login to be carried out through automated means at later time.

Browser extension integration. Instead of requiring researchers to re-implement existing tools, the platform affords the ability to incorporate existing automation and privacy browser extensions. For example, during one of our previous studies we installed the iMacros browser extension in each instantiated browser as a method of automated login. Researchers may also want greater visibility into which elements of the DOM (e.g., included javascript elements) perform calls to third-party websites. Our platform has allowed researchers to install a custom extension that logs what calls are made to which third-parties, and which elements of the DOM performed them. These studies are described in Section 5. In future work, we plan to implement an interface with which browser extensions can input commands directly from the task manager. This will complement our current support for exporting data collected from extensions.

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. By sharing these history logs, researchers can enable others to re-play and verify studies.

4.4 Evaluation

As we describe in Section 5, we have utilized OpenWPM to conduct multiple large-scale experiments. We now present an evaluation of OpenWPM's performance and generality.

Stability. Using vanilla Selenium in a variety of settings, 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. Using the isolation provided by our browser manager and task manager, we recover from all browser crashes and have observed no data corruption during crawls of 10,000 pages while retaining browser state. During the course of our news personalization study described in Section 5, we successfully crawled 281,494 pages during which we 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, stateful parallel crawls are memory limited while stateless parallel crawls 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. OpenWPM 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.

5. APPLICATIONS

We describe six recent studies that utilized OpenWPM to answer research questions in the fields of security, privacy, and personalization. Table 1 gives an overview of the capabilities of OpenWPM that each study utilized.

The ability to automate a browser and keep state while browsing are the two most commonly used features of Open-

Study	Type	Browser automation	Stateful crawls	Persistent profiles	Fine-grained profiles	Advanced plugin support	Automated login	Detect tracking cookies	Monitor state changes	Content extraction
HSTS and key pinning misconfigurations [28]	Security	●	●			●	○			●
FB Connect login permissions [51]	Privacy	●					●			○
Persistent tracking mechanisms [6]	Privacy	●	●	●	●			●	●	
Surveillance implications of web tracking [50]	Privacy	●	●			●		●		
Secondary authentication mechanisms	Security	●	●	●	●	●	●			○
News personalization ("filter bubble")	Personalization	●	●	●						○

Table 1: Studies carried out using OpenWPM
An unfilled circle indicates that the feature was useful but application-specific programming or manual effort was still required.

WPM by the studies we examine. This is because these tasks are fundamental to web measurement; users are tracked and profiled by their browser's properties and the history they accumulate. Providing real browser automation in a robust way allows researcher of the platform to skip that low-level step and build application specific features on top.

5.1 Privacy

Persistent Tracking Mechanisms. Tracking mechanisms have moved beyond simple cookie storage to extend to a diverse set of locations within the browser's client-side storage. Similarly, tracking has extended beyond a single domain as more domains begin to share cookie identifiers through cookie synchronization. OpenWPM was used in a study which measured the use of canvas fingerprinting, evercookies, and cookie synchronization [6]. This study found that canvas fingerprinting was occurring on 5% of the top 100,000 sites, that cookie respawning was still in use on many popular non-US sites, and that nearly 40% of cookie identifiers are shared between third-parties. The study further analyzes how the use of these technologies together increases the difficulty in achieving a "clean state".

Several of OpenWPM's features were essential to both the evercookie and cookie synchronization measurements. An evercookie is an identifier that is stored in many locations which respawns after partial-deletions and cookie synchronization describes the processes by which two parties share their tracking identifiers. In order to detect these processes automatically, we first need to be able to create and save an accumulated browser state from all tests sites. For everookie analysis, we then need to clear all portions of the state except the one we are testing, and revisit each test site to see if identifiers are respawned in other locations. For cookie synchronization, we simply need to identify which cookie IDs are shared between two different domains via network traffic.

The ability to accumulate a profile of all local browser storage (i.e., cookies, local storage, LSOs), save this profile to disk, and load only parts of it is supported by OpenWPM's persistent and fine-grained profile management. Similarly, the per-site browser state monitoring allows us to attribute actions to each individual site without having to make independent, isolated measurements on each site. The classification of identifiers for both evercookie IDs and cookie sync IDs is provided by our ID detection algorithm.

Surveillance with cookies. Modern websites often embed content from many different hosts. Our own measurements show that the top 500 Alexa sites include resources from, on average, 62 unique hostnames each. In an updated version of a past study of network surveillance [50], we show how the inclusion of diverse and overlapping subsets on each individual site enables a passive network adversary to transitively link between 60-75% of page visits together without any additional information. We also show how different browser blocking settings and privacy extensions can help reduce, but not fully eliminate a user's exposure to surveillance.

To make this measurement, we need to generate user profiles that simulate real users as they browse to sites they would typically visit. We simulate real browsing history by picking 25 users from the AOL search logs [12], re-running their searches on `google.com`, and visiting the top 5 results. Once the profiles have been created, we need to record all network traffic that occurs as the simulated user makes the page visits. We rely on OpenWPM's stateful crawls to allow us to build that profile for all 25 users and make measurements that reflect each user's exposure to surveillance. The platform's plugin support allowed us to also test several real privacy settings (i.e., blocking cookies and setting the DNT flag) and browser extensions (i.e. Ghostery[1] and HTTPS-Everywhere[18]). Similar to the Persistent tracking mechanisms study, the ability to recognize persistent identifiers with our ID detection algorithm was fundamental to all of the analysis in the study.

Facebook login permissions. Do users understand what personal information they are giving up when they login with Facebook? A recent study utilized OpenWPM to collect the permissions requested via Facebook's federated login on the top websites which support the system [51]. This data was used to provide quantitative support for a user study examining how well users understand permissions.

This study utilized a preliminary version of our federated login tool to successfully complete the login procedure from the sign-in page on 203 of the Alexa top 500 sites that support Facebook login. Permissions were collected concur-

rently during the login procedure.

5.2 Security

Two of our studies are about web security. We emphasize that we don't aim to be a comprehensive platform for web security measurement, but some of our features end up being useful for security measurement as well. For example, our ability to manipulate tracking cookies was also useful for manipulating authentication cookies and transferring them between profiles in order to study secondary auth mechanisms.

HSTS and key pinning misconfigurations. OpenWPM has been used to conduct the first in-depth empirical study of two important new web security features, strict transport security (HSTS) and public-key pinning [28]. This study identified several errors in the real-world HSTS deployment. The authors found that "A substantial portion of HSTS domains and nearly all pinned domains leaked cookie values, including login cookies, due to the poorly-understood interaction between HTTP cookies and the same-origin policy."

This study highlights the advanced plug-in support OpenWPM provides. Web drivers such as Selenium don't provide fine-grained visibility of pages during rendering; rather, users are only provided a fully-loaded page from which to extract content. Utilizing a proxy doesn't provide a full solution either. HTTP GET requests captured by a proxy do not capture which elements of the page initiated the request. To solve this, the researchers utilized our platform and created a custom browser extension that was installed in each deployed browser. Across the 10,000 sites surveyed, the extension recorded and logged all resource calls as the page was loaded to capture dynamic resource loading. The researchers chose to only partially utilize the automated login capability due to the incomplete coverage. Since the number of sites they needed to log into was small, they were able to manually input credentials.

Secondary authentication mechanisms. Authentication cookies provide a convenient access method for online services yet, if compromised, can be as damaging as a stolen password. This allowed us to conduct a measurement with OpenWPM that analyzed which sites might detect stolen cookies through the use of secondary authentication methods. For example, if a server receives an authentication cookie from a user who suddenly changes IP address or browser fingerprint, the server might determine that the user's authentication cookies were compromised.

This study leveraged several advanced capabilities of OpenWPM. We installed the iMacros browser extension to conduct partially automated logins across 50 sites. Once authenticated, the browser cookies were saved through the platform's profile management. The browser fingerprint was then altered before again automatically logging in to the site. We varied the fingerprint by logging in to the service provider from a residential ISP (e.g., Verizon), then connecting through the Tor network or an EC2 node acting as a proxy and refreshing the browser. These capabilities of OpenWPM allowed us to simulate a passive man-in-the-middle who intercepts a user's authentication cookies for authentication from the attacker's machine.

Not only did this study determine that many sites fail to perform any secondary authentication methods, but it found that Twitter was using a single, unchanging authentication token for each user across sessions. Our findings of weakness aren't without exceptions: the online US banking site we tested successfully identified non-US IP addresses and locked the account immediately after attempting to reauthenticate. In some cases, social media and email sites identified Tor IP addresses and would subsequently present the user with additional security questions following a password login. We omit the full details for lack of space. Our findings are similar to other recent work [44].

5.3 Personalization

Online 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 [55]. As proof-of-concept of the effectiveness of our generalized infrastructure, we performed an automated, rigorous measurement across nine major news sites of whether a user who browses a specific category of news articles might be presented more articles in that category.

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. 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.

To summarize our results, we found *no statistically significant deviations* of trained instances from control instances in article category distribution for *front page visits*. On the *article pages*, we find statistically significant levels of personalization in many of the news boxes from content recommendation engines such as Outbrain[2] and Taboola[3]. However, the levels of personalization are quantitatively small, with a level of change between the control and measurement set of articles ranging from -19.2% to +10.6%. Thus, we saw both personalization toward and away from the training category. To be able to estimate these small effect sizes and reject the null hypothesis, we used large sample sizes amounting to thousands of article views for each publisher/category pair.

Illustrating the scale of studies that are possible with OpenWPM, the 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 infeasible previously. Furthermore, personalization is difficult to detect and requires rigorous experiment methods which OpenWPM provides sup-

port for out of the box. Lastly, while OpenWPM's content extraction features were useful, we also wrote a study-specific content extraction module that located and stored news headlines and articles for offline topic classification. Without the combination of the advanced capabilities of OpenWPM, a significant level of manual effort would have been required to execute a similar study, likely limiting the scale to a single news publisher.

6. CONCLUSION

Web privacy measurement has the potential to play a key role in keeping online privacy incursions and power imbalances in check. To achieve this potential, WPM tools must be made available broadly rather than just within the research community, but without losing scientific rigor. In this work, we've tried to bring this ambitious goal closer to reality. We are happy to report that OpenWPM is currently being used independently for studies by multiple groups outside our own institution.

We are currently expanding our results in several directions. We are continuing to add capabilities to OpenWPM as we have highlighted thoughout Section 4, and several new studies are in progress. We plan to explore the use of OpenWPM in conjunection with other tools such as XRay, AdFisher, and crows-sourcing platforms.

OpenWPM is part of the broader Princeton Web Transparency and Accountability project. We've established partnerships with the Common Crawl project, for running fully web-scale studies, and with the EFF and Ghostery (via an academic license) with the aim of bringing measurements results into browser privacy tools. In the long run, we plan to operate continual, automated crawls and publish measurement results so that privacy practices of various entities can be monitored over time — a "web privacy census."

7. 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 Research, Franziska Roesner at UW, Gunes Acar and Marc Juarez at KU Leuven, Vincent Toubiana at CNIL, France, Lukasz Olejnik at INRIA, France, and the participants of the Princeton Web Privacy and Transparency workshop.

	Targets	Infrastructure			Variable						
Paper		Automation[a]	Instrumentation	Crowd-sourced Distributed	Location	User-agent	Demographics	Interests	Privacy Tools	Scale
Leakage of PII via OSN ('09) [31]	PII leaks	M*	LHH							
Privacy diffusion on the web ('09) [30]	Tracking: cookies	F,PS	Proxy							1.2K sites
Challenges in measuring ('10) [25]	Personalization: ads		Proxy		•		•			730 queries
Flash cookies and privacy ('10) [53]	Tracking: cookies, LSOs	M*								100 sites
Privacy leakage in mOSN ('10) [32]	PII leaks	M*	Proxy							
Flash cookies and privacy II ('11) [10]	Tracking: cookies, LSOs	M*								100 sites
Privacy leakage vs. protection measures ('11) [29]	PII leaks	M*	Proxy							10 sites
Respawn HTTP Cookies ('11) [41]	Tracking: cookies, LSOs	UA*			•					600 sites
Self-help tools ('11)[38]	Tracking: cookies	UA*	FourthParty						•	500 sites
Where everybody knows your username ('11) [39]	PII leaks	M*	FourthParty				•			185 sites
Detecting and defending ('12) [52]	Tracking: cookies	FF, TT	TrackingTracker							2K sites
Detecting price and search discrimination ('12) [42]	Price discrimination	SA, CH, IE, JS	Proxy	•	•	•	•	•		200 sites
Mac users steered to pricier hotels ('12) [37]	Personalization: steering					•				
Measuring the effectiveness of privacy tools ('12) [11]	Personalization: ads	F, SL							•	
Websites vary prices ('12) [57]	Personalization: prices, deals				•					
What they do with what they know ('12) [60]	Personalization: ads		Proxy							10 days
AdReveal ('13) [34]	Personalization: ads		Proxy, Ghostery				•			103K sites
Cookieless monster ('13) [47]	Tracking: fingerprinting									10K sites
Crowd-assisted search ('13) [43]	Price discrimination	F, CH	Custom plugin	•	•		•			600 sites
Discrimination in online ad delivery ('13) [54]	Ads	M, UA			•	•				2184 names
FPDetective ('13) [7]	Tracking: fingerprinting, JS	CR, SL, CJ, PJ	Proxy, Browser Code							1M sites
Know your personalization ('13) [35]	Personalization: search		Custom plugin	•			•			5K queries
Measuring personalization of web search ('13) [26]	Personalization: search	PJ		•		•				120 queries
Who knows what about me? ('13) [36]	PII leaks	F, PS, SL		•			•		•	1.5K sites
Selling off privacy at auction ('13) [49]	Cookie sync, bid prices	F, SL		•	•			•		5K sites
Shining the floodlights ('13) [19]	Tracking: cookies, JS	F, JS	FourthParty			•				500 sites
Statistical approach ('13) [22]	General tracking	F, PY	FourthParty						•	2K sites
Adscape ('14) [13]	Personalization: ads	F, SL	Custom plugin				•			10K sites
Bobble ('14) [61]	Personalization: search	CH, SL	Custom plugin	•	•	•				1K queries
Information flow experiments ('14) [56]	Personalization: ads	F, SL	Proxy				•			
Third-party OSN applications ('14) [14]	PII leaks	F, SL	FourthParty				•			997 apps
Price discrimination and steering ('14) [27]	Price disc, steering	PJ		•	•	•	•	•		16 sites
Price discrimination of airline tickets ('14) [59]	Price discrimination	CJ			•	•	•	•		21 days

[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, LHH = Live HTTP Headers, Asterisk = inferred

Table 2: Overview of prior measurement studies

8. REFERENCES

[1] Ghostery. https://www.ghostery.com/.

[2] Outbrain Content Recommendation Network. http://www.outbrain.com/.

[3] Taboola Content Recommendation Network. https://taboola.com/.

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

[5] Rubin causal model. https://en.wikipedia.org/wiki/Rubin_causal_model, 2014. [Accessed 11-2014].

[6] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS 2014)*, 2014.

[7] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.

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

[9] S. B. Automation. Selenium faq. https://code.google.com/p/selenium/wiki/FrequentlyAskedQuestions, 2014.

[10] M. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle. Flash cookies and privacy II: Now with HTML5 and ETag respawning. *World Wide Web Internet And Web Information Systems*, 2011.

[11] R. Balebako, P. Leon, R. Shay, B. Ur, Y. Wang, and L. Cranor. Measuring the effectiveness of privacy tools for limiting behavioral advertising. In *Web 2.0 Workshop on Security and Privacy*, 2012.

[12] M. Barbaro and T. Zeller. A Face Is Exposed for AOL Searcher No. 4417749. http://www.nytimes.com/2006/08/09/technology/09aol.html?pagewanted=all, 2006. [Online; accessed 10-November-2014].

[13] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan. Adscape: Harvesting and analyzing online display ads. 2014.

[14] A. Chaabane, Y. Ding, R. Dey, M. A. Kaafar, K. Ross, et al. A closer look at third-party OSN applications: Are they leaking your personal information? In *Passive and Active Measurement conference*, 2014.

[15] F. T. 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.

[16] A. Datta, M. C. Tschantz, and A. Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. *arXiv preprint arXiv:1408.6491*, 2014.

[17] W. Davis. KISSmetrics Finalizes Supercookies Settlement. http://www.mediapost.com/publications/article/191409/kissmetrics-finalizes-supercookies-settlement.html, 2013. [Online; accessed 12-May-2014].

[18] Electronic Frontier Foundation (EFF). HTTPS-Everywhere. https://www.eff.org/https-everywhere.

[19] C. Eubank, M. Melara, D. Perez-Botero, and A. Narayanan. Shining the floodlights on mobile web tracking - a privacy survey. W2SP, 2013.

[20] E. Felten. Measurement, policy making, and enforcement panel. Web Privacy and Transparency, 2014.

[21] S. Flaxman, S. Goel, and J. M. Rao. Ideological segregation and the effects of social media on news consumption. *Available at SSRN*, 2013.

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

[23] A. S. Gerber and D. P. Green. *Field Experiments: Design, Analysis, and Intepretation*. Norton, W. W. and Company, Inc., 2012.

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

[25] S. Guha, B. Cheng, and P. Francis. Challenges in measuring online advertising systems. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010.

[26] A. Hannak, P. Sapiezynski, A. Molavi Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. 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.

[27] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring price discrimination and steering on e-commerce web sites. 2014.

[28] M. Kranch and J. Bonneau. Upgrading HTTPS in midair: HSTS and key pinning in practice. In *NDSS '15: The 2015 Network and Distributed System Security Symposium*, February 2015.

[29] B. Krishnamurthy, K. Naryshkin, and C. Wills. Privacy leakage vs. protection measures: the growing disconnect. In *Proceedings of the Web*, volume 2, 2011.

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

[31] B. Krishnamurthy and C. 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.

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

[33] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. Xray: Enhancing the webâĂŹs transparency with differential correlation. In *USENIX Security Symposium*, 2014.

[34] B. Liu, A. Sheth, U. Weinsberg, J. Chandrashekar, and R. Govindan. AdReveal: improving transparency into online targeted advertising. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM, 2013.

[35] A. Majumder and N. 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.

[36] D. Malandrino, A. Petta, V. Scarano, L. Serra, R. Spinelli, and B. 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.

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

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

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

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

[41] A. M. McDonald and L. F. Cranor. Survey of the use of Adobe Flash Local Shared Objects to respawn HTTP cookies, a. *ISJLP*, 7, 2011.

[42] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 2012.

[43] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Crowd-assisted search for price discrimination in e-commerce: first results. *arXiv preprint arXiv:1307.4531*, 2013.

[44] Y. Mundada, N. Feamster, B. Krishnamurthy, S. Guha, and D. Levin. Half-baked cookies: Client authentication on the modern web. 2014.

[45] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. You are what you include: Large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 736–747. ACM, 2012.

[46] N. Nikiforakis, W. Joosen, and B. Livshits. Privaricator: Deceiving fingerprinters with little white lies.

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

[48] F. Ocariza, K. Pattabiraman, and B. Zorn. Javascript errors in the wild: An empirical study. In *Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium on*, pages 100–109. IEEE, 2011.

[49] L. Olejnik, T. Minh-Dung, C. Castelluccia, et al. Selling off privacy at auction. 2013.

[50] D. Reisman, S. Englehardt, C. Eubank, P. Zimmerman, and A. Narayanan. Cookies that give you away: Evaluating the surveillance implications of web tracking. Manuscript, 2014.

[51] N. Robinson and J. Bonneau. Cognitive disconnect: understanding facebook connect login permissions. In *Proceedings of the second edition of the ACM conference on Online social networks*, pages 247–258. ACM, 2014.

[52] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *9th USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[53] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle. Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

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

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

[56] M. C. Tschantz, A. Datta, A. Datta, and J. M. Wing. A methodology for information flow experiments. Technical Report arXiv:1405.2376, ArXiv, 2014.

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

[58] S. Van Acker, N. Nikiforakis, L. Desmet, W. Joosen, and F. Piessens. Flashover: Automated discovery of cross-site scripting vulnerabilities in rich internet applications. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 12–13. ACM, 2012.

[59] T. Vissers, N. Nikiforakis, N. Bielova, and W. Joosen. Crying wolf? on the price discrimination of online airline tickets. HotPETS, 2014.

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

[61] X. Xing, W. Meng, D. Doozan, N. Feamster, W. Lee, and A. C. Snoeren. Exposing inconsistent web search results with bobble. In *Passive and Active Measurement*, pages 131–140. Springer, 2014.

[62] C. Yue and H. Wang. A measurement study of insecure javascript practices on the web. *ACM Transactions on the Web (TWEB)*, 7(2):7, 2013.

[63] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. 2014.