ShareMeNot

Balancing Privacy and Functionality of Third-Party Social Widgets

FRANZISKA ROESNER, CHRISTOPHER ROVILLOS, TADAYOSHI KOHNO, AND DAVID WETHERALL

Franzi Roesner is a fourth-year PhD student in Computer Science and Engineering at the University of Washington, where she is advised by Yoshi Kohno. Her research focuses on helping users better understand and control how their data is used and shared in a range of scenarios, from third-party tracking on the Web to permission granting for applications in modern operating systems (such as smartphones). She received her BS in Computer Science from the University of Texas at Austin and has done internships at Amazon.com, Microsoft Research, and Google.
franzi@cs.washington.edu

Chris Rovillos is a second-year undergraduate student at the University of Washington. He is studying computer engineering and human-centered design and engineering.
crovillo@cs.washington.edu

Tadayoshi Kohno is an Associate Professor in the University of Washington's Department of Computer Science and Engineering and an Adjunct Associate Professor in the UW Information School. Kohno received his PhD from the University of California at San Diego and his BS from the University of Colorado. His research focuses on helping protect the security, privacy, and safety of users of current and future technologies.
yoshi@cs.washington.edu

David Wetherall is a Professor of Computer Science & Engineering at the University of Washington and led Intel's former Seattle research lab from 2006 to 2009. Wetherall received his PhD in computer science from MIT and his BE in electrical engineering from the University of Western Australia. His research interests are focused on network systems, especially wireless networks and mobile computing, and Internet measurement and protocol design. He is known for pioneering research on programmable networks, Internet mapping, network de-duplication, and denial-of-service.
djw@cs.washington.edu

Web tracking, the practice by which Web sites identify and collect information about users, generally in the form of some subset of Web browsing history, has become a topic of increased public debate. In this article, we summarize what we have learned about the Web tracking ecosystem and describe a taxonomy for understanding Web tracking behavior. In particular, we found that no existing browser mechanisms prevent tracking by social media sites via widgets (such as the Facebook "Like" button) while still allowing those widgets to achieve their utility goals. We then describe ShareMeNot, a browser extension that we developed to balance privacy with the intended functionality of social widgets.

Social widgets like the ones shown in Figure 1 allow their providers (ßFacebook, Google, Twitter, and others) to track a user's online browsing activities on every site that includes one of these buttons. This tracking is possible even if users never interact with the widgets and (in most browsers) even if users employ common defenses for third-party tracking, such as disabling third-party cookies.

To close the gap in defenses available to users, we introduce ShareMeNot. ShareMeNot is a browser extension (for Firefox and for Chrome) that aims to find a middle ground between allowing social widgets to track users wherever they appear and retaining the functionality of these widgets when users explicitly choose to interact with them (e.g., to "like" or to "tweet" a page). ShareMeNot for Firefox works by removing the browser cookies attached to requests made while loading the buttons, and ShareMeNot for Chrome works by replacing the buttons entirely with local replacement buttons. When users choose to click on a button, ShareMeNot necessarily allows the widget provider to identify that user to carry out the widget's functionality.

In this article, we provide background on how Web tracking works and summarize the taxonomy of tracking behavior that we introduced in our recent paper [8]. We then motivate ShareMeNot by assessing the effectiveness of defenses currently available to users and describe in detail its functionality and effectiveness.

Figure 1: Example social widgets. Social media sites expose social widgets that can be used to track users across all the sites on which these widgets are embedded. We refer to this type of tracking behavior as "personal tracking," as users visit these Web sites directly during their normal browsing behavior and are often logged in and thus are not anonymous to these trackers.

What Is Web Tracking?

Third-party Web tracking refers to the practice by which an entity (the tracker), other than the Web site directly visited by the user (the site), tracks or assists in tracking the user's visit to the site. For instance, if a user visits cnn.com, a third-party tracker like doubleclick.net embedded by cnn.com to provide, for example, targeted advertising can log the user's visit to cnn.com. For most types of third-party tracking, the tracker will be able to link the user's visit to cnn.com with the user's visit to other sites on which the tracker is also embedded. We refer to the resulting set of sites as the tracker's *browsing profile* for that user. In this section, we briefly review necessary Web-related background and summarize the taxonomy introduced in our recent paper [8].

Category	Name	Profile Scope	Summary	Example	Visit Directly?
A	Analytics	Within-Site	Serves as third-party analytics engine for sites.	Google Analytics	No
B	Vanilla	Cross-Site	Uses third-party storage to track users across sites.	Doubleclick	No
C	Forced	Cross-Site	Forces user to visit directly (e.g., via popup or redirect).	InsightExpress	Yes (forced)
D	Referred	Cross-Site	Relies on a B, C, or E tracker to leak unique identifiers.	Invite Media	No
E	Personal	Cross-Site	Visited directly by the user in other contexts.	Facebook	Yes

Table 1: Classification of tracking behavior. This table summarizes the taxonomy of tracking behavior that we developed in prior work [8]. Note that trackers may exhibit multiple behaviors at once.

Property	Behavior				
	A	B	C	D	E
Tracker sets site-owned (first-party) state.	✓				
Request to tracker leaks site-owned state.	✓				
Third-party request to tracker includes tracker-owned state.		✓	✓		✓
Tracker sets its state from third-party position; user never directly visits tracker.		✓			
Tracker forces user to visit it directly.			✓		
Relies on request from another B, C, or E tracker (not from the site itself).				✓	
User voluntarily visits tracker directly.					✓

Table 2: Tracking behavior by mechanism. In order for a tracker to be classified as having a particular behavior (A, B, C, D, or E), it must display the indicated property. Note that a particular tracker may exhibit more than one of these behaviors at once.

Web-Related Background

When a page is fetched by the browser, an HTTP request is made to the site for a URL in a new top-level execution context for that site (that corresponds to a user-visible window with a site title). The HTTP response contains resources of several kinds (HTML, scripts, images, stylesheets, iFrames, and others) that are processed for display and that may trigger HTTP requests for additional resources. Resources (such as iFrames) fetched from another domain and embedded on the page are known as *third-party content*.

Web tracking relies fundamentally on a Web site's ability to store state on the user's machine, as do most functions of today's Web. Client-side state may take many forms—most commonly, traditional browser cookies. A *cookie* is a triple (domain, key, value) that is stored in the browser across page visits, where domain is a Web site, and key and value are opaque identifiers. Cookies that are set by the domain that the user visits directly (the domain displayed in the browser's address bar) are known as *first-party cookies*; cookies that are set by some other domain embedded in the top-level page are *third-party cookies*.

Cookies are set either by scripts running in the page using an API call, or by HTTP responses that include a Set-Cookie header. The browser automatically attaches cookies for a domain to outgoing HTTP requests to that domain, using Cookie headers. Cookies may also be retrieved using an API call by scripts running in the page and then sent via any channel, such as part of an HTTP request (e.g., as part of the URL). The same-origin policy ensures that cookies (and other client-side state) set by one domain cannot be directly accessed by another domain.

Users may choose to block cookies via their browser's settings menu. Blocking all cookies is uncommon, as it makes today's Web almost unusable (e.g., the user cannot log into any account), but blocking third-party cookies is commonly recommended as a first line of defense against third-party tracking.

Background on Tracking

Web tracking is highly prevalent on the Web today. From the perspective of Web site owners and of trackers, it provides desirable functionality, including personalization, site analytics, and targeted advertising. From the perspective of a tracker, the larger a browsing profile it can gather about a user, the better service it can provide to its customers (the embedding Web sites) and to the user herself (e.g., in the form of personalization).

While some users may benefit from the results of this tracking, larger browsing profiles spell greater loss of privacy for users. A user may not, for instance, wish to link the articles he or she views on a news site with the type of adult sites he or she visits, much less reveal this information to an unknown third party. Even if the user is not worried about the particular third party, this data may later be revealed to unanticipated parties through court orders or subpoenas.

In our recent paper [8], we investigated tracking in the wild today and introduced a taxonomy of third-party Web tracking behavior. This taxonomy (summarized in Table 1) focuses on *explicit* tracking mechanisms, i.e., tracking mechanisms that use assigned, unique identifiers per user rather than *inferred* tracking based on browser and machine fingerprinting. Other work [9] has studied the use of fingerprinting to pinpoint a host with high accuracy.

More specifically, all trackers we considered have two key capabilities:

- The ability to store a pseudonym (unique identifier) on the user's machine.
- The ability to communicate that pseudonym, as well as visited sites, back to the tracker's domain.

The pseudonym may be stored using any client-side storage mechanism, including conventional browser cookies, HTML5 LocalStorage, Flash cookies, and others. Stored values are communicated to the tracker either automatically when the browser includes a cookie with a request, or explicitly by tracker-provided JavaScript code that accesses and transmits the stored values. Similarly, the browser may communicate information about the visited site to the tracker, either implicitly via the HTTP Referrer header, or explicitly via tracker-provided code using the document.referrer API call.

Depending on the mechanisms used by a tracker, the browsing profiles it compiles can be *within-site* or *cross-site*. Within-site browsing profiles link the user's browsing activity on one site with his or her other activity only on that site, including repeat visits and how the Web site is traversed, but not to visits to any other site. Cross-site browsing profiles link visits to multiple different Web sites to a given user (identified by a unique identifier or linked by another technique [6, 9]).

Our tracking taxonomy categorizes tracking behavior based on client-side observable mechanisms. It distinguishes between within-site and cross-site trackers, and it further distinguishes different types of cross-site trackers. This is in contrast to past work that considered business relationships between trackers and the embedding Web site [4] and past work that categorized trackers based on prevalence rather than user browsing profile size [5]. These distinctions are important, because they have different implications for how to detect and defend against the various behaviors.

We summarize the behavior types defined in our taxonomy in Table 1 and below. Table 2 captures the relationships of specific observable tracking mechanisms to

these behavioral categories. In order to fall into a particular behavior category, the tracker *must* exhibit (at least) all of the properties indicated for that category in Table 2. A single tracker may exhibit more than one of these behaviors.

- **A (Analytics):** The tracker serves as a third-party analytics engine for sites. It can only track users within sites.
- **B (Vanilla):** The tracker uses third-party storage that it can get and set only from a third-party position to track users across sites.
- **C (Forced):** The cross-site tracker forces users to visit its domain directly (e.g., popup, redirect), placing it in a first-party position.
- **D (Referred):** The tracker relies on a B, C, or E tracker to leak unique identifiers to it, rather than on its own client-side state, to track users across sites.
- **E (Personal):** The cross-site tracker is visited by the user directly in other contexts.

In the remainder of this article, we focus in more detail on personal tracking behavior.

Personal Trackers

Personal trackers are defined as those whose domains the user otherwise visits intentionally (e.g., facebook.com). Many of these sites, primarily social networking sites, expose social widgets such as the Facebook "Like" button, the Twitter "Tweet" button, the Google "+1" button and others (see Figure 1). These widgets can be included by Web sites to allow users logged in to these social networking sites to Like, Tweet, or +1 the embedding Web page. These widgets allow the corresponding tracker to create a cross-site browsing profile of a user across any Web sites that he or she visits that includes such a widget.

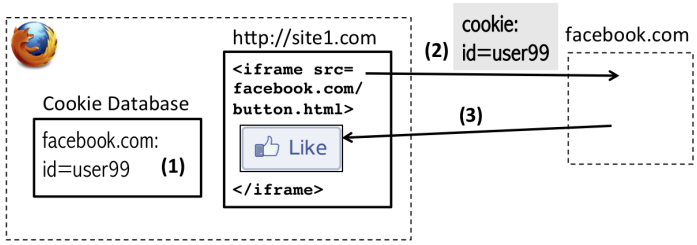

Figure 2: Personal tracking via social widgets. Social sites like Facebook, which users visit directly in other circumstances allowing the site to (1) set a cookie identifying the user, expose social widgets such as the "Like" button. When another Web site embeds such a button, the request to Facebook to render the button (2-3) includes Facebook's cookie. This allows Facebook to track the user across any site that embeds such a button.

These buttons present a privacy risk for users because they track users even when they choose not to click on any of the buttons. Like traditional third-party tracking content, simply loading a social widget provides the tracker with sufficient information to create a cross-site browsing profile for the user. That is, cookies and referrer information are included with requests to load the widget, with no user interaction required.

Furthermore, because users are often logged into the Web sites that expose such widgets (e.g., Facebook or Google), this tracking may not be anonymous. These

trackers have the ability to link the cross-site browsing profiles they collect about users with the personal information users have entered directly into their accounts at those Web sites.

As an example, Figure 2 overviews the interaction between Facebook, a site embedding a "Like" button, and the user's browser. The requests made to facebook.com to render this button allow Facebook to track the user across sites. Unlike vanilla tracking behavior, Facebook sets its cookie from a first-party position when the user voluntarily visits facebook.com. As a result, defenses like third-party cookie blocking are ineffective against personal trackers. In the next section, we explore the weaknesses of existing defenses available to users.

Existing Defenses Against Personal Trackers

We find that existing defenses against third-party Web tracking available to users today are not well suited to defend against personal trackers. In particular, existing defenses either fail to prevent personal tracking behavior or disable desired functionality (e.g., the user's ability to "Like" a Web page and share it back to his or her Facebook account).

Third-party cookie blocking is insufficient for personal trackers, for a number of reasons. First, different browsers implement third-party cookie blocking with different degrees of strictness. While Firefox blocks third-party cookies from being *set* as well as from being *sent*, most other browsers (including Chrome, Safari, and Internet Explorer) only block the setting of third-party cookies. So, for example, Facebook can set a first-party cookie when the user visits facebook.com; in browsers other than Firefox, this cookie, once set, is available to Facebook from a third-party position (when embedded on another page).

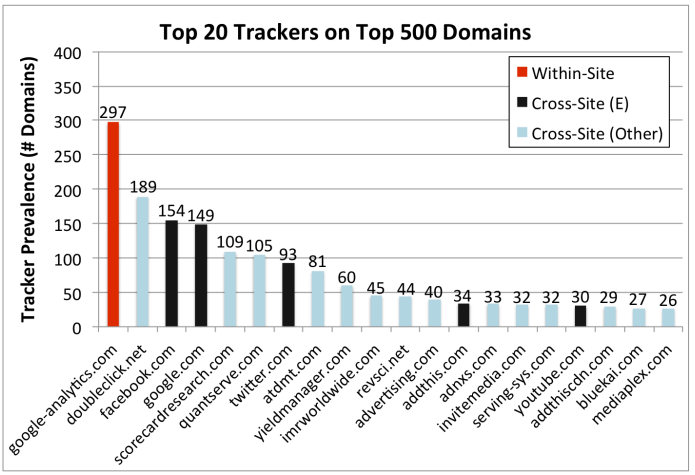

Figure 3: Prevalence of trackers on top 500 domains [8]. This graph shows the prevalence of the top 20 trackers on the Alexa top 500 domains from our 2011 measurement study with no defenses enabled. Compare to Figure 4, in which third-party cookies are blocked.

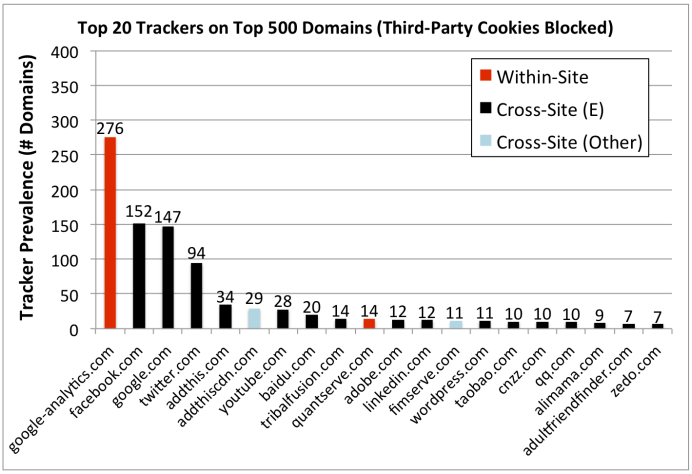

Figure 4: Prevalence of trackers on top 500 domains with third-party cookies blocked [8]. When third-party cookies are blocked, personal trackers dominate the set of top 20 trackers. Personal trackers are not affected by third-party cookie blocking, because users visit the trackers' Web sites directly, allowing them to set cookies from a first-party position.

Thus, in most browsers, third-party cookie blocking protects users only from trackers that are never visited directly. Figures 3 and 4 show data from our measurement study described in [8]. Notice that third-party cookie blocking is effective for many cross-site trackers, but it is ineffective for personal trackers, leaving them as the prominent remaining trackers when third-party cookies are blocked (Figure 4).

Firefox's strict policy provides better protection, but at the expense of functionality like social widgets and buttons (thus prompting Mozilla to opt against making this setting the default [7]).

The recently proposed Do Not Track header and legislation aim to give users a standardized way to opt out of Web tracking via a browser setting that appends a DNT=1 heading to outgoing requests. DNT has prompted a debate over the definition of tracking, as its conclusion determines to which parties the legislation will apply. Facebook, for instance, argues that personal tracking should not be subject to Do Not Track because users already have an explicit relationship with the tracker [3].

Users can attempt to minimize the size of the browsing profiles trackers can create by frequently clearing the client-side state that contains unique identifiers. However, when users log into the Web sites of personal trackers, the new identifier is by definition linked with the old identifier, as both are linked to the user's account on the tracker's Web site.

Logging out of a social media site may not prevent the tracking of a user, or prevent the linking of a user's browsing profile while they are not logged in with their browsing profile while logged in. Logging out of these sites often does not clear any or all cookies containing unique identifiers [1], allowing them to continue to be used for tracking.

Several possible defenses exist in the form of browser extensions that allow users to block trackers. These defenses, including NoScript (http://noscript.net), Ghostery (http://www.ghostery.com), and Disconnect (http://disconnect.me) (which targets personal trackers directly), work by simply blocking the tracker's scripts and their associated buttons from being loaded by the browser at all. This approach effectively removes the buttons from the user's Web experience entirely and thus removes potentially desired functionality.

ShareMeNot

We introduce the ShareMeNot browser extension to protect users from tracking by social widgets while still allowing these widgets to be used. The use of ShareMeNot shrinks the profile that the supported personal trackers can create to only those sites on which the user explicitly clicks on one of the buttons, at which point the button provider must necessarily know the user's identity in order to link the "Like" or the "+1" action to the user's profile. No other existing approach can shrink the profile a personal tracker can create while also retaining the functionality of the buttons, although concurrent work on the Priv3 Firefox add-on [2] adopts the same basic approach; as of May 2012, Priv3 supports fewer widgets and, to our knowledge, was not iteratively refined through measurement.

ShareMeNot supports social widgets from Facebook, Google, Twitter, AddThis, YouTube, LinkedIn, Digg, and Stumbleupon. We chose to support these sites based in part on our initial, pre-experimental perceptions of popular third-party trackers, and in part based on our experimental discovery of the top trackers.

ShareMeNot for Firefox

ShareMeNot for Firefox works by stripping cookies from third-party requests to any of the supported personal trackers that are made during the loading of a social widget. ShareMeNot strips cookies from two types of requests:

- *Requests to a tracker's domain that have another domain as the referrer.* Most requests made during the loading of a social widget fit this rule.
- *Specific blacklisted requests, of referrer.* This rule is necessary because of the complexity of some of the social widgets, which include multiple chained requests. For example, loading the Facebook "Like" button involves requests to facebook.com with the referrer facebook.com, rather than the embedding site. ShareMeNot's blacklist covers these requests.

When ShareMeNot detects that a user has clicked on a button by recognizing the characteristic request that follows a click on each supported widget, it allows the cookies to be included with the request. In most cases, this allows the button click to function as normal and as transparent to the user. The Facebook "Like" button is more complex, however, and must first be reloaded in the logged-in state to be active. Thus, a ShareMeNot user must click on this button twice: the first click will reload the button with personalized content (e.g., "5 of your friends have liked this") and the second will actually "like" the page.

ShareMeNot for Firefox does not fully block requests to the trackers, instead only removing cookies from the relevant requests. Thus, it may expose the user's IP address and other fingerprinting information that can be used for implicit tracking. It also does not block programmatic access to document.cookie, which would

allow personal trackers attempting to circumvent ShareMeNot to continue accessing cookie values. ShareMeNot for Chrome addresses these weaknesses.

ShareMeNot for Chrome

Unlike ShareMeNot for Firefox, ShareMeNot for Chrome blocks entire HTTP requests to tracker buttons. The buttons are replaced with locally stored versions of the buttons that offer the same functionality. ShareMeNot for Chrome works in two phases: first, blocking HTTP requests to tracker buttons, then inserting replacement buttons where the tracker buttons were to originally have been.

ShareMeNot leverages the newly introduced WebRequest API in Chrome to monitor HTTP requests before they are sent by the browser. When a user visits a Web page, the HTTP requests sent as the page is loaded are compared to a predefined set of URL patterns that identify requests for tracker buttons; if a URL matches a pattern, the entire HTTP request is blocked. In that case, ShareMeNot displays an icon in the Chrome location bar notifying the user, who can choose to unblock certain sites by clicking on the icon. To avoid impacting functionality when users directly visit social media sites such as Facebook, ShareMeNot does not block requests for top-level pages. Instead, it only blocks requests for resources that are requested by the page that is loading, such as scripts, images, and other Web pages embedded via iFrames.

In the second phase, ShareMeNot inserts replacement buttons. A Chrome extension content script is executed in the context of the current page after it has loaded. As the original widgets were blocked from loading in the first phase, the content script must search the page using a predefined set of CSS selectors for HTML tags or other clues about where the buttons should have been. It replaces them with iFrame elements that point to the replacement buttons stored within the extension. These replacement buttons either directly activate (for example, opening the appropriate Twitter sharing page if the user clicks on the "Tweet" button) or load the original button when clicked. For example, clicking on the replacement Facebook "Like" button loads the actual "Like" button; as in ShareMeNot for Firefox, the user must click twice to actually "like" the page. This is necessary because some social media sites don't have direct links for button actions; the user must use that social media site's real buttons.

By blocking all requests to tracker domains until users click on the replacement buttons, ShareMeNot for Chrome prevents the leakage of the user's IP address and other fingerprinting information, as well as access to document.cookie. We hope to update ShareMeNot for Firefox to match this more privacy-preserving design in the future.

Effectiveness

We experimentally verified the effectiveness of ShareMeNot for Firefox (we expect similar results for ShareMeNot for Chrome). As summarized in Table 3, ShareMeNot dramatically reduces the presence of the personal trackers it supports to date. ShareMeNot entirely eliminates tracking by most of these, including Twitter, AddThis, YouTube, Digg, and Stumbleupon. While it does not entirely remove the presence of Facebook and Google, it reduces their prevalence to 9 and 15 occurrences, respectively. In the Facebook case, this is due to the Facebook comments widget, which triggers additional first-party requests (containing tracking infor-

mation) not blacklisted by ShareMeNot; the Google cases appear mostly on other Google domains (e.g., google.ca).

Tracker	*Without ShareMeNot*	*With ShareMeNot*
Facebook	154	9
Google	149	15
Twitter	93	0
AddThis	34	0
YouTube	30	0
LinkedIn	22	0
Digg	8	0
Stumbleupon	6	0

Table 3: Effectiveness of ShareMeNot. ShareMeNot drastically reduces the occurrences of tracking behavior by the supported set of personal trackers.

To date, ShareMeNot does not fully support the complete set of social widgets exposed by the supported trackers. Facebook, in particular, exposes a broader set of "social plugins" that ShareMeNot renders nonfunctional (because it does not properly activate them when users attempt to interact with them) and/or for which it does not properly prevent tracking (because it has an incomplete request blacklist). We hope to address these issues in future versions.

As of May 2012, we have seen over 25,000 downloads from our own servers (http://sharemenot.cs.washington.edu/), in addition to over 8,500 daily users as reported by the official Mozilla add-on site (https://addons.mozilla.org/firefox/addon/sharemenot/).

Conclusion

We have introduced ShareMeNot, a browser extension that protects users from personal tracking by third-party social widgets while retaining the functionality of these widgets should users wish to click on them. ShareMeNot can be downloaded from our Web site, http://sharemenot.cs.washington.edu.

Acknowledgments

We thank the readers and reviewers of earlier versions of our related NSDI paper [8] for their valuable feedback throughout this work: Jon Crowcroft, Daniel Halperin, Arvind Narayanan, and Charlie Reis. We thank Brandon Lucia for naming ShareMeNot. This work was supported in part by NSF Awards CNS-0722000, CNS-0846065, and CNS-0917341, an NSF Graduate Research Fellowship under Grant No. DGE-0718124, an Alfred P. Sloan Research Fellowship, and a gift from Google.

References

[1] N. Cubrilovic, "Logging Out of Facebook Is Not Enough," 2011: http://nikcub.appspot.com/posts/logging-out-of-facebook-is-not-enough.

[2] M. Dhawan, C. Kreibich, and N. Weaver, "The Priv3 Firefox Extension,": http://priv3.icsi.berkeley.edu/.

[3] Facebook, "Facebook's Position on 'Do Not Track,'" W3C Workshop on Web Tracking and User Privacy, 2011: http://www.w3.org/2011/track-privacy/papers/Facebook.html.

[4] C. Jackson, A. Bortz, D. Boneh, and J.C. Mitchell, "Protecting Browser State from Web Privacy Attacks," WWW 2006: http://www2006.org/programme/item.php?id=3536.

[5] B. Krishnamurthy and C. Wills, "Privacy Diffusion on the Web: A Longitudinal Perspective," WWW, 2009.

[6] B. Krishnamurthy, K. Naryshkin, and C. Wills, "Privacy Leakage vs. Protection Measures: The Growing Disconnect," Web 2.0 Security and Privacy, 2011: http://www.w2spconf.com/2011/papers/privacyVsProtection.pdf.

[7] Mozilla, "Bug 417800 Revert to Not Blocking Third-Party Cookies," 2008: https://bugzilla.mozilla.org/show_bug.cgi?id=417800.

[8] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and Defending against Third-Party Tracking on the Web," NSDI '12.

[9] T.-F. Yen, Y. Xie, F. Yu, R.P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," NDSS 2012.