This paper appears at the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2012).

Detecting and Defending Against Third-Party Tracking on the Web

Franziska Roesner, Tadayoshi Kohno, and David Wetherall
University of Washington

Abstract

While third-party tracking on the web has garnered much attention, its workings remain poorly understood. Our goal is to dissect how mainstream web tracking occurs in the wild. We develop a client-side method for detecting and classifying five kinds of third-party trackers based on how they manipulate browser state. We run our detection system while browsing the web and observe a rich ecosystem, with over 500 unique trackers in our measurements alone. We find that most commercial pages are tracked by multiple parties, trackers vary widely in their coverage with a small number being widely deployed, and many trackers exhibit a combination of tracking behaviors. Based on web search traces taken from AOL data, we estimate that several trackers can each capture more than 20% of a user's browsing behavior. We further assess the impact of defenses on tracking and find that no existing browser mechanisms prevent tracking by social media sites via widgets while still allowing those widgets to achieve their utility goals, which leads us to develop a new defense. To the best of our knowledge, our work is the most complete study of web tracking to date.

1 Introduction

Web tracking, the practice by which websites 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. To date, however, the research community's knowledge of web tracking is piecemeal. There are many specific ways that identifying information might be gleaned (e.g., browser fingerprinting [4], ETags [2], and Flash cookies [21]) but little assessment of how tracking is integrated with web browsing in practice. Further complicating the situation is that the capabilities of different trackers depend strongly on their implementation. For instance, it is common for trackers like Google Analytics and Doubleclick to be mentioned in the same context (e.g., [14]) even though the former is implemented so that it cannot use unique identifiers to track users across sites while the latter can.

As the tracking arms race continues, the design of future web systems must be informed by an understanding of how web trackers retask browser mechanisms for tracking purposes. Our goal is thus to provide a comprehensive assessment of third-party tracking on the web today, where a third-party tracker is defined as a website (like `doubleclick.net`) that has its tracking code included or embedded in another site (like `cnn.com`). We focus on third-party tracking because of its potential concern to users, who may be surprised that a party with which they may or may not have chosen to interact is recording their online behavior in unexpected ways. We also explicitly focus on mainstream web tracking that uses cookies and other conventional local storage mechanisms (HTML5, Flash cookies) to compile records of users and user behavior. This is the most prevalent form of tracking today. More esoteric forms of tracking do exist — such as tracking with Etags [2], visited link coloring [10], and via the cache — and would threaten privacy if widely deployed, but they are not commonly used today. We similarly exclude inference-based browser and machine fingerprinting techniques, commonly used for online fraud detection [22, 24], in favor of explicit tracking that pinpoints users with browser state.

Our approach is to detect tracking as it is observed by clients; we achieve this goal by integrating web tracking detection directly into the browser. We began by looking at how real tracker code interacts with browsers, and from there distill five distinct behavior types. In our system, we are able to distinguish these different sets of behaviors, for example classifying Google Analytics and Doubleclick as distinct.

We then developed a Firefox browser extension to measure the prevalence of different web trackers and tracking behaviors. We aimed our tool at the 500 most popular and 500 less popular websites according to the Alexa rankings, as well as real user workloads as approximated with web traces generated from publicly available AOL search logs. Our measurements reveal extensive tracking. Pages are commonly watched by more than one of the over 500 unique trackers we found. These trackers exhibit a variety of nondeterministic behaviors, including hierarchies in which one tracker hands off to another. Several trackers have sufficient penetration that they may capture a large fraction of a user's browsing activity.

Our method also allowed us to assess how today's defenses reduce web tracking. We found that popup blocking, third-party cookie blocking and the Do Not Track header thwarted a large portion of cookie-based tracking without impacting functionality in most browsers, with the exception of tracking by social media widgets. Disabling JavaScript is more effective but can significantly impact the browsing experience. Tracking by social media widgets (e.g., Facebook) has rapidly grown in coverage and highlights how unanticipated combinations of browser mechanisms can have unexpected effects. Informed by our understanding of this kind of tracking, as well as the inadequacy of existing solutions, we developed the ShareMeNot extension to successfully defend against it while still allowing users to interact with the widgets.

To summarize, we make several contributions. Our classification of tracking behaviors is new, and goes beyond simple notions of first- and third-party tracking. Our measurements of deployed web trackers and how much they track users give the most detailed account of which we are aware to date of tracking in the wild, as well as an assessment of the efficacy of common defenses. Finally, our ShareMeNot extension provides a new defense against a practical threat. We now turn to additional background information in Section 2.

2 Background

Third-party web tracking refers to the practice by which an entity (the tracker), other than the website 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. Before diving into the mechanisms of third-party tracking, we briefly review necessary web-related background.

2.1 Web-Related Background

Page Fetching. 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, and others) that are processed for display and which may trigger HTTP requests for additional resources. This process continues recursively until loading is complete.

Execution Context. A website can embed content from another domain in two ways. The first is the inclusion of an iframe, which delegates a portion of the screen to the domain from which the iframe is sourced — this is considered the *third-party domain*. The *same-origin policy* ensures that content from the two domains is isolated: any scripts running in the iframe run in the context of the third-party domain. By contrast, when a page includes a script from another domain (using `<script src=...>`), that script runs in the domain of the embedding page (the *first-party domain*), not in that of the script's source.

Client-Side Storage. Web tracking relies fundamentally on a website'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[1], 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.

In addition to traditional cookies, HTML5 introduced new client-side storage mechanisms for browsers. In particular, *LocalStorage* provides a persistent storage area that sites can access with API calls, isolated by the same-origin policy. Plugins like Flash are another mechanism by which websites can store data on the user's machine. In the case of Flash, websites can set *Local Storage Objects* (LSOs, also referred to as "Flash cookies") on the user's file system.

2.2 Background on Tracking

Web tracking is highly prevalent on the web today. From the perspective of website owners and of trackers, it provides desirable functionality, including personalization, site analytics, and targeted advertising. A recent study [6] claims that the negative economic impact of preventing targeted advertising — or the underlying tracking mechanisms that enable it — is significant. 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 websites) and to the user herself (e.g., in the form of personalization).

[1]On October 3, 2011, the Gibson Research Corporation cookie statistics page (`http://www.grc.com/cookies/stats.htm`) showed that almost 100% of 70,834 unique visitors in the previous week had first-party cookies enabled.

From the perspective of users, however, larger browsing profiles spell greater loss of privacy. 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.

Despite the prevalence of tracking and the resulting public and media outcry — primarily in the United States and in Europe — there is a lack of clarity about how tracking works, how widespread the practice is, and the scope of the browsing profiles that trackers can collect about users. Tracking is often invisible; tools like the Ghostery Firefox add-on[2] aim to provide users with insight into the trackers they encounter on the web. What these tools do not consider, however, are the differences between types of trackers, their capabilities, and the resulting scope of the browsing profiles they can compile. For example, Google Analytics is commonly considered to be one of the most prominent trackers. However, it does not have the ability to create cross-site browsing profiles using the unique identifiers in its cookies. Thus, its prevalence is not correlated with the size of the browsing profiles it creates.

Storage and Communication. Our study focuses on *explicit* tracking mechanisms — tracking mechanisms that use assigned, unique identifiers per user — rather than *inferred* tracking based on browser and machine fingerprinting. Other work [25] has studied the use of fingerprinting to pinpoint a host with high accuracy. More specifically, all trackers we consider have two key capabilities:

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

The pseudonym may be stored using any of the client-side storage mechanisms described in Section 2.1 — in a conventional browser cookie, in HTML5 LocalStorage, and in Flash LSOs, as well as in more exotic locations such as in ETags. There are multiple ways in which the browser may communicate information about the visited site to the tracker, e.g., implicitly via the *HTTP Referrer header* or explicitly via tracker-provided JavaScript code that directly transmits the results of an `document.referrer` API call. In some cases, a script running within a page might even communicate the visited page information in the GET or POST parameters of a request to a tracker's domain. For example, a tracker embedded on a site might access its own cookie and the referring page, and then pass this information on to another tracker with a URL of the form `http://tracker2.com/track?cookie_value=123&site=site.com`.

[2] `http://www.ghostery.com`

Different Scales of Tracking. Depending on the behaviors exhibited and 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 website is traversed, but not to visits to any other site. Cross-site browsing profiles link visits to multiple different websites to a given user (identified by a unique identifier or linked by another technique [16, 25]).

Behavioral Methodology. In this paper, we consider tracking behavior that is observable from the client, that is, from the user's browser. Thus, we do not distinguish between "can track" and "does track" — that is, we analyze trackers according to the capabilities granted by the behaviors we observe and not, for example, the privacy policies of the tracking sites.

From the background that we have introduced in this section, we step back and consider, via archetypical examples, the set of properties exhibited by trackers (Section 3.1); from these properties we formulate a classification of tracking behavior in Section 3.2.

3 Classifying Web Tracking Behavior

All web trackers that use unique identifiers are often bundled into the same category. However, in actuality diverse mechanisms are used by trackers, resulting in fundamentally different tracking capabilities. Our observations, based both on manual investigations and automated measurements, lead us to believe that it is incorrect to bundle together different classes of trackers — for example, Google Analytics is a within-site tracker, while Doubleclick is a cross-site tracker. To rigorously evaluate the tracking ecosystem, we need a framework for differentiating different tracker types. We develop such a framework here (Section 3.2). To inform this framework, we first dive deeply into an investigatory analysis of how tracking occurs today (Section 3.1), where we identify different properties of different trackers. We use our resulting framework as the basis for our measurements in Section 4.

3.1 Investigating Tracking Properties

In order to understand patterns of tracking behavior, we must first understand the properties of different trackers. We present several archetypal tracking examples here and, from each, extract a set of core properties.

Throughout this discussion, we will refer to cookies set under a tracker's domain as *tracker-owned* cookies. We introduce this term rather than using "third-party cookies" because a given cookie can be considered a first-party or a third-party cookie depending on the current browsing context. (For example, Facebook's cookie is a first-party cookie when the user visits `facebook.com`, but it is a third-party cookie when a Facebook "Like" button is

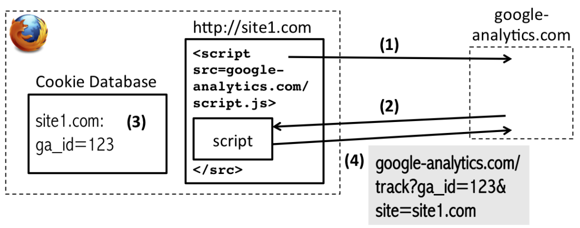

Figure 1: *Case Study: Third-Party Analytics.* Websites commonly use third-party analytics engines like Google Analytics (GA) to track visitors. This process involves (1) the website embedding the GA script, which, after (2) loading in the user's browser, (3) sets a site-owned cookie. This cookie is (4) communicated back to GA along with other tracking information.

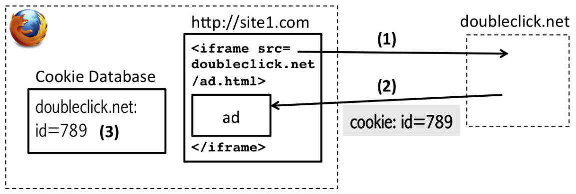

Figure 2: *Case Study: Third-Party Advertising.* When a website (1) includes a third-party ad from an entity like Doubleclick, Doubleclick (2-3) sets a tracker-owned cookie on the user's browser. Subsequent requests to Doubleclick from any website will include that cookie, allowing it to track the user across those sites.

embedded on another webpage.) Similarly, a cookie set under the domain of the website embedding a tracker is a *site-owned* cookie.

3.1.1 Third-Party Analytics

For websites that wish to analyze traffic, it has become common to use a third-party analytics engine such as Google Analytics (GA) in lieu of collecting the data and performing the analysis themselves. The webpage directly visited by the user includes a library (in the form of a script) provided by the analytics engine on pages on which it wishes to track users (see Figure 1).

To track repeat visitors, the GA script sets a cookie on the user's browser that contains a unique identifier. Since the script runs in the page's own context, the resulting cookie is site-owned, not tracker-owned. The GA script transfers this identifier to `google-analytics.com` by making explicit requests that include custom parameters in the URL containing information like the embedding site, the user identifier (from the cookie), and system information (operating system, browser, screen resolution, geographic information, etc.).

Because the identifying cookie is site-owned, identifiers set by Google Analytics across different sites are different. Thus, the user will be associated with a different pseudonym on the two sites, limiting Google Analytics's ability to create a cross-site browsing profile for that user.

Tracker Properties. We extract the following set of properties defining trackers like Google Analytics:

1. The tracker's script, running in the context of the site, sets a site-owned cookie.
2. The tracker's script explicitly leaks the site-owned cookie in the parameters of a request to the tracker's domain, circumventing the same-origin policy.

3.1.2 Third-Party Advertising

The type of tracking most commonly understood under "third-party tracking" is tracking for the purpose of targeted advertising. As an example of this type of tracking scenario, we consider Google's advertising network, Doubleclick. Figure 2 shows an overview of this scenario.

When a page like `site1.com` is rendered on the user's browser, Doubleclick's code will choose an ad to display on the page, e.g., as an image or as an iframe. This ad is hosted by `doubleclick.net`, not by the embedding page (`site1.com`). Thus, the cookie that is set as the result of this interaction (again containing a unique identifier for the user) is tracker-owned. As a result, the same unique identifier is associated with the user whenever any site embeds a Doubleclick ad, allowing Doubleclick to create a cross-site browsing profile for that user.

Tracker Properties. We extract the following properties defining trackers like Doubleclick:

1. The tracker sets a tracker-owned cookie, which is then automatically included with any requests to the tracker's domain.
2. The tracker-owned cookie is set by the tracker in a third-party position — that is, the user never visits the tracker's domain directly.

3.1.3 Third-Party Advertising with Popups

A commonly recommended first line of defense against third-party tracking like that done by Doubleclick is third-party cookie blocking. However, in most browsers, third-party cookie blocking applies only to the setting, not to the sending, of cookies (in Firefox, it applies to both). Thus, if a tracker is able to maneuver itself into a position from which it can set a first-party cookie, it can avoid the third-party cookie blocking defense entirely.

We observed this behavior from a number of trackers, such as `insightexpressai.com`, which opens a popup window when users visit `weather.com`. While popup windows have other benefits for advertising (e.g., better capturing a user's attention), they also put the tracker into a first-party position without the user's consent. From there, the tracker sets and reads first-party cookies, remaining unaffected by third-party cookie blocking.

Tracker Properties. We extract the following properties defining trackers like Insight Express:

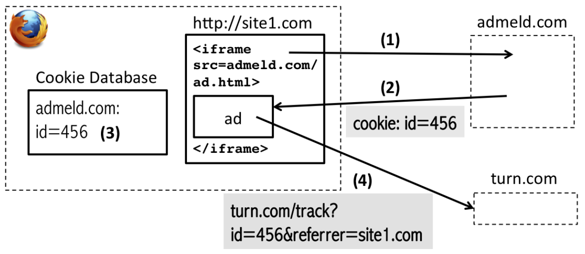

Figure 3: *Case Study: Advertising Networks.* As in the ordinary third-party advertising case, a website (1-2) embeds an ad from Admeld, which (3) sets a tracker-owned cookie. Admeld then (4) makes a request to another third-party advertiser, Turn, and passes its own tracker-owned cookie value and other tracking information to it. This allows Turn to track the user across sites on which Admeld makes this request, without needed to set its own tracker-owned state.

1. The tracker forces the user to visit its domain directly, e.g., with a popup or a redirect, allowing it to set its tracker-owned cookie from a first-party position.
2. The tracker sets a tracker-owned cookie, which is then automatically included with any requests to the tracker's domain when allowed by the browser.

3.1.4 Third-Party Advertising Networks

While, from our perspective, we have limited insights into the business models of third-party advertisers and other trackers, we can observe the effects of complex business relationships in the requests to third-parties made by the browser. In particular, trackers often cooperate, and it is insufficient to simply consider trackers in isolation.

As depicted in Figure 3, a website may embed one third-party tracker, which in turn serves as an aggregator for a number of other third-party trackers. We observed this behavior to be common among advertising networks. For example, `admeld.com` is often embedded by websites, and it makes further requests to trackers like `turn.com` and `invitemedia.com`. In these requests, `admeld.com` includes the information necessary to track the user, including the top-level page and the pseudonym from `admeld.com`'s own tracker-owned cookie. This means that `turn.com` does not need to set its own client-side state, but rather can rely entirely on `admeld.com`.

Tracker Properties. We extract the following properties defining trackers of this type:

1. The tracker is not embedded by the first-party website directly, but referred to by another tracker on that site.
2. The tracker relies on information passed to it in a request by the cooperating tracker.

3.1.5 Third-Party Social Widgets

An additional class of trackers doubles as sites that users otherwise visit intentionally, and often have an account with. Many of these sites, primarily social networking sites, expose social widgets like the Facebook "Like" button, the Twitter "tweet" button, the Google "+1" button and others. These widgets can be included by websites to allow users logged in to these social networking sites to like, tweet, or +1 the embedding webpage.

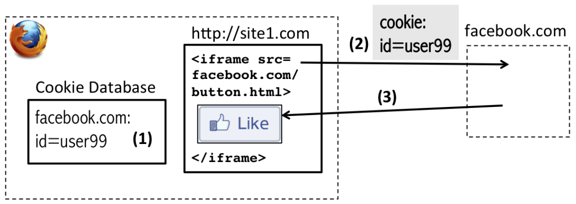

Figure 4: *Case Study: Social Widgets.* Social sites like Facebook, which users visit directly in other circumstances — allowing them to (1) set a cookie identifying the user — expose social widgets such as the "Like" button. When another website embeds such a button, the request to Facebook to render the button (2-3) includes Facebook's tracker-owned cookie. This allows Facebook to track the user across any site that embeds such a button.

Figure 4 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 just as Doubleclick can — though note that unlike Doubleclick, Facebook sets its tracker-owned cookie from a first-party position when the user voluntarily visits `facebook.com`.

Tracker Properties. We extract the following set of properties, where the important distinction to the Doubleclick scenario is the second property:

1. The tracker makes use of a tracker-owned cookies.
2. The user voluntarily visits the tracker's domain directly, allowing it to set the tracker-owned cookie from a first-party position.

3.2 A Classification Framework

We now present a classification framework for web trackers based on observable behaviors. This is in contrast to past work that considered business relationships between trackers and the embedding website rather than observable behaviors [9] and past work that categorized trackers based on prevalence rather than user browsing profile size, thereby commingling within-site and cross-site tracking [14]. In particular, from our manual investigations we distilled five tracking behavior types; we summarize these behaviors below and in Table 1. Table 2 captures the key properties from Section 3.1 and their relationships 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, as we discuss in more detail below.

1. **Behavior A (Analytics)**: The tracker serves as a third-party analytics engine for sites. It can only track users within sites.
2. **Behavior B (Vanilla)**: The tracker uses third-party storage that it can get and set only from a third-party

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.* Trackers may exhibit multiple behaviors at once, with the exception of Behaviors B and E, which depend fundamentally on a user's browsing behavior: either the user visits the tracker's site directly or not.

	Behavior				
Property	*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.

position to track users across sites.

3. **Behavior C (Forced)**: The cross-site tracker forces users to visit its domain directly (e.g., popup, redirect), placing it in a first-party position.
4. **Behavior 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.
5. **Behavior E (Personal)**: The cross-site tracker is visited by the user directly in other contexts.

This classification is based entirely on tracker behavior that can be observed from the client side. Thus, it does not capture backend tracking behavior, such as correlating a user's browsing behavior using browser and machine fingerprinting techniques, or the backend exchange of data among trackers. Similarly, the effective type of a tracker encountered by a user depends on the user's own browsing behavior. In particular, the distinction between Behavior B and Behavior E depends on whether or not the user ever directly visits the tracker's domain.

Combining Behaviors. Most of these behaviors are not mutually exclusive, with the exception of Behavior B (Vanilla) and Behavior E (Personal) — either the user directly visits the tracker's domain at some point or not. That is, a given tracker can exhibit different behaviors on different sites or multiple behaviors on the same site. For example, a number of trackers — such as `quantserve.com` — act as both Behavior A (Analytics) and Behavior B (Vanilla) trackers. Thus, they provide site analytics to the embedding sites in addition to gathering cross-site browsing profiles (for the purposes of targeted advertising or additional analytics).

Through our analysis, we identified what was to us a surprising combination of behaviors — Behavior A (Analytics) and Behavior D (Referred) — by which a within-site tracker unintentionally gains cross-site

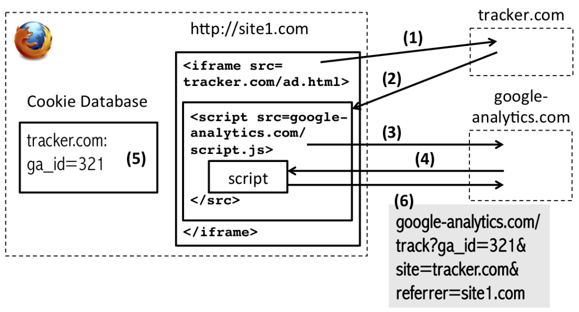

Figure 5: *Combining Behavior A and Behavior D.* When a Behavior A tracker like Google Analytics is embedded by another third-party tracker, rather than by the visited website itself, Behavior D emerges. The site-owned cookie that GA sets on `tracker.com` becomes a tracker-owned cookie when `tracker.com` is embedded on `site1.com`. The tracker then passes this identifier to Google Analytics, which gains the ability to track the user across all sites on which `tracker.com` is embedded.

tracking capabilities. We discovered this combination during our measurement study. For example, recall that Google Analytics is a within-site, not a cross-site, tracker (Behavior A). However, suppose that `tracker.com` uses Google Analytics for its own on-site analytics, thus receiving a site-owned cookie with a unique identifier. If `tracker.com` is further embedded on another site, this same cookie *becomes a tracker-owned cookie*, which is the same across all sites on which `tracker.com` is embedded. Now, when the usual request is made to `google-analytics.com` from `tracker.com` when it is embedded, *Google Analytics becomes a Behavior D tracker* — a cross-site tracker. Figure 5 shows an overview of this scenario. Note that we did not observe many instances of this in practice, but it is interesting to observe that within-site trackers can become cross-site trackers when different parties interact in complex ways. This observation is further evidence of the fact that the tracking ecosystem is complicated and that it is thus difficult to create simple, sweeping technical or policy solutions.

Robustness. We stress that this classification is agnostic of the practical manifestation of the mechanisms described above — that is, client-side storage may be done via cookies or any other mechanism, and information may be communicated back to the tracker in any way. This separation of semantics from mechanism makes the classification robust in the face of the evolution of specific client-side storage techniques used by trackers.

4 Detecting Trackers

Based on this classification framework, we created a tool — TrackingTracker — that automatically classifies trackers according to behavior observed on the client-side. TrackingTracker runs as a Firefox add-on, interposes on all HTTP(S) requests, and examines conventional cookies, HTML5 Local Storage, and Flash LSOs to detect and categorize trackers. It has support for crawling a list of websites to an arbitrary link depth and for performing a series of search engine keyword searches and visiting the top hit of the returned search results. We used this tool to perform a series of analyses between September and October of 2011; unless otherwise noted, our discussion reflects only behaviors observed during that time.

In presenting the results of these measurements, we make a distinction between *pages* and *domains*. Two pages may belong to the same domain (e.g., `www.cnn.com/article1` and `www.cnn.com/article2`). Which we use depends on whether we are interested in the characteristics of websites (domains) or in specific instances of tracking behavior (pages).

Note that the tracking behavior that we observe in our measurements is a lower bound, for several reasons. First, we do not log into any sites or click any ads or social widgets, which we have observed in small case studies to occasionally trigger additional tracking behavior. Second, we have observed that tracking behavior can be nondeterministic, largely due to the interplay of Behavior B (Vanilla) and Behavior D (Referred) trackers; we generally visit pages only twice (see below), which may not trigger all trackers embedded by a given website.

Finally, the mere presence of a cookie (or other storage item) does not by itself give a tracker the ability to create a browsing profile — the storage item must contain a unique identifier. It is difficult or impossible to identify unique identifiers with complete certainty (we do not reverse-engineer cookie strings), but we identify and remove any suspected trackers whose cookies or other storage contain identical values across multiple measurements that started with a clean browser. We also remove trackers that only use session cookies, though we note that these can equally be used for tracking as long as the browser remains open.

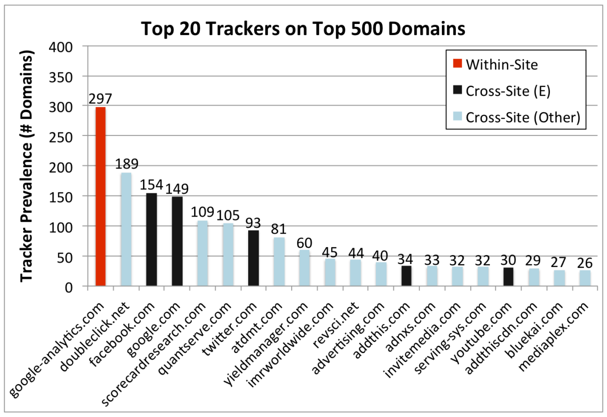

Figure 6: *Prevalence of Trackers on Top 500 Domains.* Trackers are counted on domains, i.e., if a particular tracker appears on two pages of a domain, it is counted once.

4.1 Tracking on Popular Sites

We collected a data set using the top 500 websites (international) from Alexa as published on September 19, 2011. We also visited four random links on each of the 500 sites that stayed within that site's domain. We visited and analyzed a total of 2098 unique pages for this data set; we did not visit a full 2500 unique pages because some websites do not have four within-domain links, some links are broken or redirect to other domains or to the same page, etc. This process was repeated twice: once starting with a clean browser, and once more after priming the cache and cookie database (i.e., without first clearing browser state). This experimental design aims to ensure that trackers that may only set but not read state the first time they are encountered are properly accounted for by TrackingTracker on the second run. The results we report include tracking behavior measured only on the second run.

Most of the 2098 pages (500 domains) embed trackers, often several. Indeed, the average number of trackers on the 1655 pages (457 domains) that embed at least one tracker is over 4.5 (over 7). Of these, 1469 pages (445 domains) include at least one cross-site tracker.

Overall, we found a total of 524 unique trackers appearing a cumulative 7264 times. Figure 6 shows the twenty top trackers across the 500 top domains. This graph considers websites as domains — that is, if a given tracker was encountered on two pages of a domain, it is only counted once in this graph. The most prevalent tracker is Google Analytics, appearing on almost 300 of the 500 domains — recall that it is a within-site tracker, meaning that it cannot link users' visits across these pages using cookies. The most popular cross-site tracker that users don't otherwise visit directly is Doubleclick (also owned by Google), which can track users across almost 40% of the 500 most popular sites. The most popular Behavior E tracker (domains that are themselves in the top 500) is Facebook, followed closely by Google, both of which are found on almost 30% of the top sites.

Tracker	Top 500 Sites		Non-Top 500 Sites		Popups Blocked		Cookies Blocked		No JavaScript		DNT Enabled	
		Instances		*Instances*		*Instances*		*Instances*		*Instances*		*Instances*
Type	*#*	*(Min, Max)*	*#*	*(Min, Max)*	*#*	*(Min, Max)*	*#*	*(Min, Max)*	*#*	*(Min, Max)*	*#*	*(Min, Max)*
A	17	49 (1, 9)	10	34 (1, 18)	17	34 (1, 10)	40	158 (1, 38)	–	–	10	39 (1, 9)
A B	18	152 (1, 21)	11	104 (1, 37)	20	338 (1, 123)	–	–	–	–	14	105 (1, 17)
A B D	1	317 (317, 317)	1	155 (155, 155)	1	319 (319, 319)	–	–	–	–	1	274 (274, 274)
A E	8	47 (1, 17)	2	25 (5, 20)	8	51 (1, 20)	10*	95 (1, 23)	–	–	6	33 (1, 17)
A E D	1	21 (21, 21)	–	–	1	19 (19, 19)	–	–	–	–	1	18 (18, 18)
A D	3	902 (1, 896)	2	908 (55, 853)	2	906 (10, 896)	1	844 (844, 844)	–	–	2	900 (81, 819)
B	357	3322 (1, 375)	299	3734 (1, 777)	336	2859 (1, 382)	1	15 (15, 15)	161	1697 (1, 263)	320	2613 (1, 305)
B C	3	79 (6, 64)	–	–	3	29 (3, 22)	5*	48 (2, 21)	–	–	6	47 (2, 15)
B D	8	703 (1, 489)	7	60 (1, 25)	22	1235 (1, 494)	–	–	2	23 (10,13)	13	1299 (3, 551)
E	101	1564 (1, 397)	41	1569 (1, 446)	101	1625 (1, 405)	96*	1509 (1, 383)	49	707 (1, 195)	100	1412 (1, 338)
E C	1	34 (34, 34)	1	23 (23, 23)	–	–	–	–	–	–	1	31 (31, 31)
E D	1	1 (1, 1)	1	417 (417, 417)	1	5 (5, 5)	1*	1 (1, 1)	–	–	1	4 (4, 4)
C	4	4 (1, 1)	4	4 (1, 1)	–	–	5	8 (1, 4)	–	–	7	13 (1, 4)
D	1	(69, 69)	3	60 (1, 57)	2	80 (1, 79)	1*	71 (71, 71)	–	–	1	42 (42, 42)
Total	524	7264 (1, 896)	382	7093 (1, 853)	514	7505 (1, 896)	160	2749 (1, 844)	212	2427	483	6830 (1, 819)

Table 3: *Measurement Results.* Each set of columns reports the results for the specified measurement, each run with the Alexa top 500 sites except the Non-Top dataset. The lefthand column for each set reports the number of unique trackers of each type observed; the righthand column reports the number of occurrences of that tracker type. A value of `X (Y, Z)` in that column means that `X` occurrences of trackers of this type were observed; the minimum number of occurrences of any unique tracker was `Y` and the maximum `Z`. Values with asterisks would be zero (or shifted to another type) for Firefox users, due to that browser's stricter third-party cookie blocking policy. Some of the variation in counts across runs is due to the nondeterminism of tracking behavior.

Recall that a tracker may exhibit different behavior across different occurrences, often due to varying business and embedding relationships. We thus consider occurrences in addition to unique trackers; this data is reported in the first set of columns in Table 3. In this table, a tracker classified as, for instance, type A B D, may exhibit different combinations of the three behaviors at different times.

We find that most trackers behave uniformly across occurrences. For example, the most common tracking behaviors are Behavior B (Vanilla) and Behavior E (Personal), which these trackers exhibit uniformly. Some trackers, however, exhibit nonuniform behavior. For instance, sites may choose whether or not to use Quantserve for on-site analytics (Behavior A) in addition to including it as a Behavior B (cross-site) tracker. Thus, Quantserve may sometimes appear as Behavior A, sometimes as Behavior B, and sometimes as both. When other trackers include Quantserve, it can also exhibit Behavior D (Referred) behavior. Similarly, Google Analytics generally exhibits Behavior A behavior, setting site-owned state on a site for which it provides third-party analytics. However, when a third-party tracker uses Google Analytics itself, as described in Section 3.2, Google Analytics is put into the position of a Behavior D (cross-site) tracker.

4.1.1 Other Storage Mechanisms

LocalStorage. Contrary to expectations that trackers are moving away from cookies to other storage mechanisms, we found remarkably little use of HTML5 LocalStorage by trackers (though sites themselves may still use it for self-administered analytics). Of the 524 unique trackers we encountered on the Alexa top 500 sites, only eight of these trackers set any LocalStorage at all. Five contained unique identifiers, two contained timestamps, and one stored a user's unsubmitted comments in case of accidental navigation away from the page.

We observed both Behavior A and Behavior B behavior using LocalStorage. Notably, we discovered that `taboolasyndication.com` and `krxd.net` set site-owned LocalStorage instead of browser cookies for Behavior A purposes.

Of the five trackers that set unique identifiers in LocalStorage, all duplicated these values in cookies. When the same identifier is stored in multiple locations, the possibility of *respawning* is raised: if one storage location is cleared, the tracker can repopulate it with the same value from the other storage location. Respawning has been observed several times in the wild [2, 21] as a way to subvert a user's intention not to be tracked and is exemplified by the proof-of-concept evercookie[3].

We manually checked for respawning behavior in these five cases and found that one tracker — `tanx.com` — indeed repopulated the browser cookies from LocalStorage when cleared. We also noticed that `twitter.com`, which set a uniquely identifying "guest id" on the machines of users that are not logged in, did not repopulate the cookie value — however, it did store in LocalStorage the entire history of guest ids, allowing the user's new guest id to be linked to the old one. This may not be intentional on Twitter's part, but the capability for tracking in this way — equivalent to respawning — exists.

We also observed reverse respawning, that is, repopulating LocalStorage from cookies when cleared. We observed this in three of the five cases, and note that it may not be intentional respawning but rather a function of when LocalStorage is populated (generally after checking if a cookie is set and/or setting a cookie).

Flash Storage. Flash LSOs, or "cookies", on the other hand, are more commonly used to store unique tracking identifiers. Nevertheless, despite media buzz about iden-

[3]http://samy.pl/evercookie/

tifier respawning from Flash cookies, we find that most unique identifiers in Flash cookies do not serve as backups to traditional cookies; only nine of the 35 trackers with unique identifiers in Flash cookies duplicate these identifiers across Flash cookies and traditional cookies.

For these nine trackers, we tested manually for respawning behavior as described above. We observed Flash-to-cookie respawning in six cases and cookie-to-Flash respawning in seven.

In one interesting case, we found that while the Flash cookie for `sodahead.com` does not appear to match the browser cookie, it is named `enc_data` and may be an encrypted version of the cookie value. Indeed, `sodahead.com` respawned the browser cookie from the Flash cookie. Furthermore, the respawned cookie was a session cookie that would ordinarily expire automatically when the browser is closed. This example demonstrates that it is not sufficient to inspect stored values but that respawning must be determined behaviorally.

4.1.2 Cookie Leaks, Countries, and More

Throughout our study, we made a number of interesting non-quantitative observations; we describe these here.

Frequent cookie leaks. We observed a large number of cookie leaks, i.e., cookies belonging to one domain that are included in the parameters of a request to another domain, thereby circumventing the same-origin policy. Fundamentally, cookie leaking enables an additional party to gain tracking capabilities that it would not otherwise have. In addition to Behavior A leaks (the leaking of site-owned state set by the tracker's code to the tracker as a third-party) and Behavior D leaks (to enable additional trackers), we observed cookie leaks indicative of business relationships between two (or few) parties.

For example, `msn.com` and `bing.com`, both owned by Microsoft, use cookie leaking mechanisms within the browser to share cookies with each other, even when the user does not visit both sites as part of a contiguous browsing session. This enables Microsoft to track a unique user across both MSN and Bing, as well as across any site that may embed one of the two.

As another example, we noticed that when a website includes both Google AdSense (a product that allows the average website owner to embed ads without a full-fledged Doubleclick contract) as well as Google Analytics, the AdSense script makes requests to Doubleclick to fetch ads. These requests include uniquely identifying values from the site's Google Analytics cookies. This practice gives Google the capability to directly link the unique identifier used by Doubleclick to track the user across sites with the unique Google Analytics identifier used to track the user's visits to this particular site. While this does not increase the size of the browsing profile that Doubleclick can create, it allows Google to relink the two profiles if the user ever clears client-side state for one but not the other.

Origin countries. In exploring the use of LocalStorage and Flash cookies, we found that trackers from different regions appear to exhibit different behaviors. The only tracker to respawn cookies from LocalStorage comes from a Chinese domain, and of the eight trackers involved in respawning to or from Flash cookies, four are US, two are Chinese, and two are Russian. The Chinese and Russian trackers seem to be overrepresented compared to their fraction in the complete set of observed trackers.

Tracker clustering. While many of the top trackers are found across sites of a variety of categories and origins, we observed some trackers to cluster around related sites. For example, in the Alexa top 500, `traffichaus.com` and `exoclick.com` are found only on adult sites (on five and six of about twenty, respectively). Similarly, some trackers are only found on sites of the same geographic origin — e.g., `adriver.ru` is found only on Russian sites and `wrating.com` only on Chinese sites.

Trackers interact with two types of users. We observed that Behavior B (Vanilla) and Behavior C (Forced) trackers sometimes do not set tracking cookies when their websites are visited directly — unlike Behavior E tracker like Facebook, which by definition set state when they are visited. In other words, for example, `turn.com` sets a third-party tracking cookie when it is embedded on another website, but not when the user visits `turn.com` directly. Some trackers, in fact, use different domains for their own homepages than for their tracking domains (e.g., visiting `doubleclick.net` redirects to `google.com/doubleclick`). Trackers that exhibit these behaviors can never be Behavior E trackers, even if the users directly visits their sites. We can only speculate about the reasons for these behaviors, but we observe that trackers interact with two types of users: users whom they track from a third-party position, and users who are their customers and who visit their website directly.

4.2 Comparison to Less Popular Sites

The measurements presented thus far give us an intuition about the prevalence and behavior of trackers on popular sites. Since it is possible that different trackers with different behavior are found on less popular sites, we collected data for non-top sites as well. In particular, we visited 500 sites from the Alexa top million sites, starting with site #501 and at intervals of 100. As in the top 500 case, we visited 4 random links on each page, resulting in a total of 1959 unique pages visited.

In this measurement, we observed 7093 instances of tracking across 382 unique trackers, summarized in the second set of columns in Table 3. Figure 7 shows the top 20 trackers (counted by domains) for this measurement.

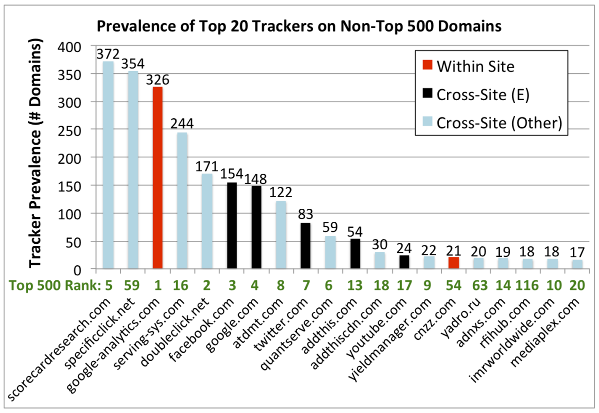

Figure 7: *Tracker Prevalence on Non-Top 500 Domains.* Trackers are counted on domains, i.e., if a particular tracker appears on two pages of a domain, it is counted once.

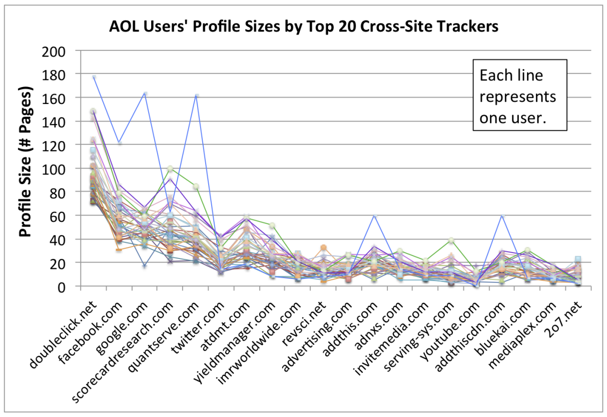

Figure 8: *Browsing Profiles for 35 AOL Users.* We report the measured profile size for each user for the 20 top trackers from the top 500, using 300 unique queries (an average of 253 unique pages visited) per user.

The numbers below each bar indicate the rank for the tracker in the top 500 domains. Note that Google Analytics and Doubleclick are no longer ranked as high, but in absolute numbers appear a similar number of times. ScorecardResearch and SpecificClick appear to be highly prevalent among among these less popular sites.

Among the non-top 500 sites, we observed less LocalStorage use — only one of the eight users of LocalStorage in the top 500 sites reappeared (`disqus.com`, which stored comment drafts); we saw one additional instance of LocalStorage, `contextweb.com`, which stored a unique value but does not duplicate it in the browser cookie. We also observed fewer Flash cookies set (68 total across all sites and trackers, compared to 110 in the top 500 measurement), finding one additional tracker — `heias.com` — that respawns its browser cookie value from the Flash cookie.

4.3 Real Users: Using the AOL Search Logs

In order to better approximate a real user's browsing history, we collected data using the 2006 AOL search query logs [20]. We selected 35 random users (about 1%) from the 3447 users with at least 300 unique queries (not necessarily clickthroughs). For each of these randomly selected users, we submitted to a search engine the first 300 of their unique queries and visited the top search result for each. This resulted in an average of 253 unique pages per user.

For the AOL users, we are interested in the size of the browsing profiles that trackers can create. Here we must consider exactly how we define "profile". In particular, a tracker receives information about the domains a user visits, the pages a user visits, and the individual visits a user makes (i.e., returning to the same page at a later time). A user may be concerned about the privacy of any of these sets of information; in the context of this study, we consider unique pages. That is, we consider the size of a browsing profile compiled by a given tracker to be the number of unique pages on which the user encountered that tracker. The reason for using pages instead of visits is that using search logs to approximate real browsing behavior involves making multiple visits to pages that a real user might not make — e.g., multiple unique queries may result in the same top search hit, which TrackingTracker will visit but a real user may not, depending on why a similar query was repeated. Though TrackingTracker may thus visit multiple pages more than once, giving more trackers the opportunity to load on that page, this is balanced by the fact that we, as before, visit each page once before recording measurements in order to prime the cache and the cookie database.

In order to compare AOL users, we focus on the top 20 cross-site trackers from the Alexa top 500 measurement. That is, we take all 19 cross-site trackers from Figure 6 as well as `serving-sys.com`, the next-highest ranked cross-site tracker. Figure 8 shows the size of the profile compiled by each tracker for each of the 35 users.

We find that Doubleclick can track a user across (on average) 39% of the pages he or she visited in these browsing traces — and a maximum of 66% of the pages visited by one user. The magnitude of these percentages may be cause for concern by privacy-conscious users. Facebook and Google can track users across an average of 23% and 21% of these browsing traces (45% and 61% in the maximum case), respectively. As many users have and are logged into Facebook and Google accounts, this tracking is likely not to be anonymous.

Two data points of note are the large profile sizes for `google.com` and `quantserve.com` for one of the users. These spikes occur because that user visited a large number of pages on the same domain (`city-data.com`), which embeds Google Maps and Quantserve.

From this data, we observe that, in general, the ranking of the trackers in the top 500 corresponds with how much real users may encounter them. In particular, `doubleclick.net` remains the top cross-site tracker; the prominence of `scorecardresearch.com` in the non-top 500 is not reflected here, perhaps because top search hits are likely biased towards more popular sites.

5 Defenses

In this section, we explore existing defenses against tracking in the context of our classification. We then present measurement results collected using the Alexa top 500 with standard defenses enabled. Finally, we propose an additional defense — implemented in the form of our Firefox add-on ShareMeNot — that aims to protect users from Behavior E tracking. Again, unless otherwise noted, we refer to observations in September/October 2011.

5.1 Initial Analysis of Defenses

Third-party cookie blocking. A standard defense against third-party web tracking is to block third-party cookies. This defense is insufficient for a number of reasons. First, different browsers implement third-party cookie blocking with different degrees of strictness. While Firefox blocks third-party cookies both 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).

Thus, in most browsers, third-party cookie blocking protects users only from trackers that are never visited directly — that is, it is effective against Behavior B (Vanilla) trackers but not against Behavior C (Forced) or Behavior E (Personal) trackers. Firefox's strict policy provides better protection, but at the expense of functionality like social widgets and buttons, some instantiations of OAuth or federated login, and other legitimate cross-domain behavior (thus prompting Mozilla to opt against making this setting the default [19]).

Do Not Track. The recently proposed Do Not Track header and legislation aim to give users a standardized way to opt out of web tracking. A browser setting (already implemented natively in Firefox, IE, and Safari) appends a DNT=1 header to outgoing requests, informing the receiving website that the user wishes to opt out of tracking. As of February 2012, Do Not Track is merely a policy technique that requires tracker compliance, providing no technical backing or enforcement. A major sticking point is the debate over the definition of tracking, as the conclusion of this debate determines to which parties the Do Not Track legislation will apply. As evidenced by the papers submitted to the W3C Workshop on Web Tracking and User Privacy[4], many of the parties involved in tracking argue that their behaviors should not be considered tracking for the purposes of DNT. It is our hope that the tracking classification framework that we have developed and proposed in this paper can be used to further the discussion of what should be considered tracking in the policy realm, and that a tool like TrackingTracker can be used in the browser to enforce and detect violations of Do Not Track.

[4]`http://www.w3.org/2011/track-privacy/`

Clearing client-side state. There has been some concern [26] that pervasive opt-out of tracking will create a tiered or divided web, in which visitors who opt out of tracking (via the DNT header or other methods) will not be provided with the same content as other visitors. One possible solution (also identified in [9]) to avoid tracking in the face of this concern is to constantly clear the browser's client-side state, regularly receiving new identifiers from trackers. This may be a sufficient solution for Behavior B, Behavior C, and Behavior D trackers, but it cannot protect users against Behavior E trackers to which they have identified themselves as a particular account holder (and thus logging back in will re-identify the same user). It is also hard to implement against Behavior A trackers, as they set first-party state on the websites that embed them, and it is difficult to distinguish in a robust manner the first-party state needed by the website from the state used by the Behavior A tracker. Other work [25] shows furthermore that fingerprinting techniques can re-identify a large fraction of hosts with fresh cookies.

Blocking popups. Most browsers today block popups by default, potentially making it more difficult for Behavior C trackers to maneuver themselves into first-party positions. However, websites can still open popups in response to user clicks. Furthermore, popups are only one way that Behavior C trackers can force a user to visit their site directly (and the easiest of these to detect and block). Other methods include redirecting the user's browser to the tracker's domain and back using javascript, or business relationships between the tracker and the embedding site that involve the site redirecting directly to a full-page interstitial ad controlled by the tracker's domain. These behaviors are hard or impossible to block as they are used throughout the web for other legitimate purposes.

Recent findings (February 2012) [18] furthermore revealed programmatic form submission as a new technique for Behavior C tracking in Safari, which treats form submissions as first-party interactions.

Private browsing mode. Private browsing mode, as explored in depth in [1], does not primarily address the threat model of web tracking. Instead, private browsing mode aims to protect browser state from adversaries with physical access to the machine. While the clearing of cookies when exiting private browsing mode can help increase a user's privacy in the face of tracking, private browsing mode does not aim to keep a user's browsing history private from remote servers.

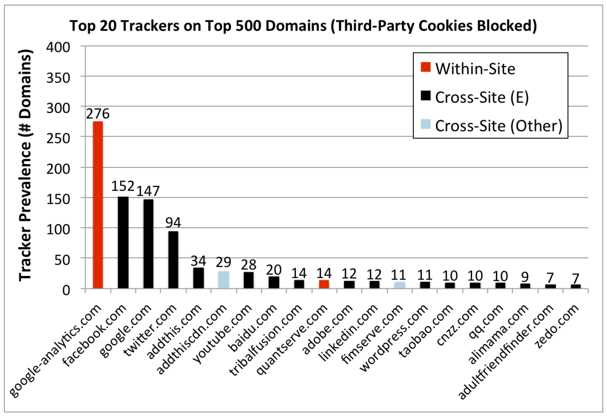

Figure 9: *Prevalence of Trackers on Top 500 Domains with Third-Party Cookies Blocked.* Trackers are counted on domains, that is, if a particular tracker appears on two pages of a domain, it is counted once.

5.2 Empirical Analysis of Defenses

As a part of our measurement study, we empirically analyzed the effectiveness of popup blocking and third-party cookie blocking to prevent tracking. The results of these measurements (run using the Alexa Top 500 domains, with 4 random links chosen from each) are summarized on the righthand side of Table 3. Overall, we find that existing defenses protect against a large portion of tracking, with the notable exception of Behavior E. We dive into the effects of each measured defense in turn.

With popups blocked, we did not observe significant differences in the tracking capabilities of most trackers. As expected, we observe fewer trackers exhibiting Behavior C (Forced) — however, Behavior C using redirects remains, leaving three trackers exhibiting such behavior. We find most Behavior C trackers exhibit this type of behavior only occasionally, acting as Behavior B (Vanilla) the rest of the time. Thus, with third-party cookies enabled, popup blocking does not affect the capabilities of most trackers. Indeed, we suspect that most popups are used to better capture the user's attention rather than to maneuver the tracker domain into a first-party position. Nevertheless, this technique is sometimes used for this purpose[5].

Third-party cookie blocking is, as expected, a better defense against tracking. However, recall that in most browsers other than Firefox, third-party cookie blocking only blocks the setting, not the sending, of cookies. Thus, if a tracker can ever set a cookie (via Behavior C or Behavior E), this cookie is available from that point forward. In Table 3, we distinguish the results for Firefox's strict cookie blocking policy: any type with an asterisk in the "Cookies Blocked" column disappears in Firefox (trackers that exhibit both Behavior A and E reduce to only A; the others disappear). Note the presence of one Behavior B tracker: this is `meebo.com`,

[5]`http://stackoverflow.com/questions/465662/`

Figure 10: *Example Social Widgets.* Behavior E trackers expose social widgets that can be used to track users across all the sites on which these widgets are embedded.

which sets unique identifiers in LocalStorage in addition to browser cookies, leaving it unaffected by cookie blocking. Figure 9 shows the top 20 trackers for this measurement (compare to Figure 6), in which it is evident that most cross-site trackers have disappeared from the top 20, leaving the prominence of Behavior E trackers like Facebook, Twitter, YouTube, and others.

We also measured the effectiveness of disabling JavaScript, the most blunt defense against tracking. We find that it is extremely effective at preventing tracking behaviors that require API access to cookies to leak them, as is the case for Behavior A (Analytics) and Behavior D (Referred). However, trackers can still set cookies via HTTP headers and Behavior C trackers can use HTML redirects. Any tracking that requires only that content be requested from a tracker is not impacted — thus, while the scripts of Behavior E and other trackers cannot run (e.g., to render a social widget), they can be requested, thereby enabling tracking. Some trackers simply use `<noscript>` tags to fetch single-pixel images ("beacons") when more complex scripting techniques are not available. Despite being the most effective single defense, disabling JavaScript renders much of today's web unusable, making it an unworkable option for most users.

The Do Not Track header does not yet appear to have a significant effect on tracking, as evidenced by the sustained prevalence of most trackers in Table 3. Note that, as in all the results we report, we did exclude any trackers that set only non-unique and/or session cookies, as some trackers may respond to the DNT header by setting an opt-out cookie. We did notice that a few fairly prevalent trackers appeared to respond to the header, including `gemius.pl`, `serving-sys.com`, `media6degrees.com` and `bluekai.com`. These results are consistent with a recent set of case studies of DNT compliance [17].

Finally, we note that we did not observe any trackers actively changing behavior in an attempt to circumvent the tested defenses — that is, we did not observe more LocalStorage or more Flash cookies. Though we have not verified whether or not these trackers instead use more exotic storage mechanisms like cache Etags, we hypothesize that enough users do not enable these defenses to mobilize trackers to substantially change their behavior, and hence fall outside our common case explorations.

5.3 A New Defense: ShareMeNot

From these measurements, we conclude that a combination of defenses can be employed to protect against a large set of trackers. However, Behavior E trackers like

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 4: *Effectiveness of ShareMeNot.* ShareMeNot drastically reduces the occurrences of tracking behavior by the supported set of Behavior E trackers.

Facebook, Google, and Twitter remain largely unaffected. Recall that these trackers can track logged-in users non-anonymously across any sites that embed so-called "social widgets" exposed by the tracker. For example, Facebook allows other websites to embed a "Like" button, Google exposes a "+1" button, and so on (see Figure 10 for a number of examples). These buttons present a privacy risk for users because they track users even when they choose not to click on any of the buttons.

When users can protect themselves from tracking in this fashion, it comes at the expense of the functionality of the button. In Firefox, the stricter third-party cookie blocking policy renders the buttons unusable. Other existing defenses, including the popular Disconnect browser extension[6], work by simply blocking the tracker's scripts and their associated buttons from being loaded by the browser at all, thereby effectively removing the buttons from the user's web experience entirely.

We introduce ShareMeNot, a Firefox add-on that aims to find a middle ground between allowing the buttons to track users wherever they appear and retaining the functionality of the buttons when users explicitly choose to interact with them. It does this by stripping cookies from third-party requests to any of the supported Behavior E trackers under ordinary circumstances (as well as from any other blacklisted requests that are made in the context of loading such a button); when it detects that a user has clicked on a button, it allows the cookies to be included with the request, thereby allowing the button click to function as normal, transparently to the user.

The use of ShareMeNot shrinks the profile that the supported Behavior E 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 and the identity of the site on which the button was found in order to link the "like" or the "+1" action to the user's profile. No other existing approach can both shrink the profile a Behavior E tracker can create while also retaining the functionality of the buttons, though concurrent work on the Priv3 Firefox add-on [3] adopts the same basic approach; as of February 2012, Priv3 supports fewer widgets and, to our knowledge, was not iteratively refined through measurement.

We experimentally verified the effectiveness of ShareMeNot. As summarized in Table 4, ShareMeNot dramatically reduces the presence of the Behavior E trackers it supports to date. 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 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 information) not blacklisted by ShareMeNot; the Google cases appear mostly on other Google domains (e.g., `google.ca`).

The currently released ShareMeNot add-on does not fully block requests to the trackers, thus exposing the user's IP address and other fingerprinting information, nor does it block programmatic access to `document.cookie`. A new version of ShareMeNot is under development that aims to address these issues by replacing widgets entirely with client-side buttons, making no requests to trackers until these replacement buttons are clicked.

As of February 2012, we have seen over 20,000 downloads from our own servers[7], in addition to over 7000 daily users as reported by the official Mozilla add-on site[8].

6 Related Work

We expand our discussions of several related works and consider additional related works not discussed above.

A number of studies have empirically examined tracking on the web, most notably [14]. In that paper, the authors present the results of a longitudinal measurement study of web tracking, examining the prevalence of third-party trackers on the web. The authors do not distinguish between different types of trackers, grouping together, for example, Google Analytics (a within-site tracker) and Doubleclick (a cross-site tracker), though they touch on aspects in prior work [15]. As discussed, we believe that this distinction is fundamentally important for understanding and responding to web tracking.

In their five-year study of modern web traffic, Ihm and Pai [8] find that ad network traffic accounts for a growing percentage of total requests (12% in 2010). They find Google Analytics on up to 40% of the pages reflected in their data, a number that has increased to over 50% in our data. Another measurement study of web tracking appeared in [12], in which the authors examined the prevalence of cookie usage and P3P policies.

[6]`http://disconnect.me`

[7]`http://sharemenot.cs.washington.edu/`

[8]`https://addons.mozilla.org/firefox/addon/sharemenot/`

From a slightly different threat model, the authors of [11] examined privacy-violating information flows on the web, though they don't distinguish third-party trackers from visited sites themselves. As in our study, they found a number of instances of cookie leaking, as well as on-site behavioral tracking and other privacy violations. In [13] and [16], the authors examine the direct leakage of private data from first-party websites to data aggregators, including the potential linkage of user accounts on separate sites.

In [9], the authors classify trackers based on cooperation between the embedding site and the trackers, which in some ways overlaps with our classification. They do not measure the prevalence of these tracker classes, and miss Behavior E (Personal), which has only emerged in popularity since the publication of that paper.

Further afield, a number of researchers [5, 7, 23] have tackled the problem of privacy-preserving targeted advertising and other personalized content, attempting to find a middle ground that balances the values of users, websites, and advertisers or other content providers.

Additionally, there have been significant online discussions about tracking, e.g., [17]. Finally, entire workshops on tracking have emerged, e.g., the 2011 Workshop on Internet Tracking, Advertising, and Privacy and the 2011 W3C Workshop on Web Tracking and User Privacy.

7 Conclusion

In this paper we presented an in-depth empirical investigation of third-party web tracking. Our empirical investigation builds on the introduction of what we believe to be the first comprehensive classification framework for web tracking based on client-side observable behaviors. We believe that this framework can serve as a foundation for future technical and policy initiatives. We additionally evaluated a set of common defenses on a large scale and observed a gap — the ability to defend against Behavior E tracking with social media widgets, like the Facebook "Like" button, while still allowing those widgets to be useful. In response, we developed and evaluated ShareMeNot, which is designed to thwart such tracking while still allowing the widgets to be used.

Acknowledgements

We thank our shepherd, Jon Crowcroft, and the anonymous NSDI reviewers for their feedback. We thank Daniel Halperin, Arvind Narayanan, and Charlie Reis for feedback on earlier drafts; Chris Rovillos for volunteering to help maintain and extend ShareMeNot; and Brandon Lucia for naming it. 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] G. Aggrawal, E. Bursztein, C. Jackson, and D. Boneh. An analysis of private browsing modes in modern browsers. In *Usenix Security Symposium*, 2010.

[2] M. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoffnagle. Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. *Social Science Research Network Working Paper Series*, 2011.

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

[4] P. Eckersley. How unique is your web browser? In *International Conference on Privacy Enhancing Technologies*, 2010.

[5] M. Fredrikson and B. Livshits. RePriv: Re-Envisioning In-Browser Privacy. In *IEEE Symp. on Security and Privacy*, 2011.

[6] A. Goldfarb and C. E. Tucker. Privacy Regulation and Online Advertising. *Management Science*, 57(1), Jan. 2011.

[7] S. Guha, B. Cheng, and P. Francis. Privad: Practical Privacy in Online Advertising. In *NSDI*, 2011.

[8] S. Ihm and V. Pai. Towards understanding modern web trafÞc. In *IMC*, 2011.

[9] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell. Protecting browser state from web privacy attacks. In *WWW*, 2006.

[10] A. Janc and L. Olejnik. Feasibility and real-world implications of web browser history detection. In *W2SP*, 2010.

[11] D. Jang, R. Jhala, S. Lerner, and H. Shacham. An empirical study of privacy-violating information flows in JavaScript web applications. In *CCS*, 2010.

[12] C. Jensen, C. Sarkar, C. Jensen, and C. Potts. Tracking website data-collection and privacy practices with the iWatch web crawler. In *SOUPS*, 2007.

[13] B. Krishnamurthy and C. Wills. On the leakage of personally identifiable information via online social networks. In *WOSN*, 2009.

[14] B. Krishnamurthy and C. Wills. Privacy diffusion on the web: a longitudinal perspective. In *WWW*, 2009.

[15] B. Krishnamurthy and C. E. Wills. Generating a privacy footprint on the internet. In *IMC*, 2006.

[16] B. Krishnamurthy, K. Naryshkin, and C. Wills. Privacy leakage vs. protection measures: the growing disconnect. In *W2SP*, 2011.

[17] J. Mayer. Tracking the Trackers: Early Results, 2011. `http://cyberlaw.stanford.edu/node/6694`.

[18] J. Mayer. Safari tracking, Jan. 2012. `http://webpolicy.org/2012/02/17/safari-trackers/`.

[19] Mozilla. Bug 417800 — Revert to not blocking third-party cookies, 2008. `https://bugzilla.mozilla.org/show_bug.cgi?id=417800`.

[20] G. Pass, A. Chowdhury, and C. Torgeson. A Picture of Search. In *Conf. on Scalable Information Systems*, 2006.

[21] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle. Flash Cookies and Privacy. *Social Science Research Network Working Paper Series*, Aug. 2009.

[22] ThreatMetrix. Tech. overview. `http://threatmetrix.com/technology/technology-overview/`.

[23] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy Preserving Targeted Advertising. In *NDSS*, 2010.

[24] R. Vamosi. Device Fingerprinting Aims To Stop Online Fraud. PCWorld, Mar. 2009. `http://www.pcworld.com/businesscenter/article/161036/`.

[25] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi. Host fingerprinting and tracking on the web: Privacy and security implications. In *NDSS*, 2012.

[26] H. Yu. Do Not Track: Not as Simple as it Sounds, Aug. 2010. `https://freedom-to-tinker.com/blog/harlanyu/do-not-track-not-simple-it-sounds`.