Harlan Yu[1]
Department of Computer Science
Center for Information Technology Policy
Princeton University

W3C Workshop on Web Tracking and User Privacy
April 28-29, 2011

Accurately Communicating the Do Not Track User Preference

One advantage of the *user preference* approach to Do Not Track is that users don't need to know in advance whether servers engage in tracking activities. The user simply needs to communicate her preference to the server, and the burden will then be on the server to refrain from any tracking. A simple and elegant way to communicate the user preference is using an HTTP header. Most users can choose a *blanket* tracking preference—to always send an "enabled" header to all sites or to never send the header to any site.

However, some users may decide to make more fine-grained tracking choices. For instance, a user could signal a preference to not be tracked only to third party domains and not to first party domains. Or, the user could consent to tracking by some third party domains and not others. Or further, the user could consent to tracking by some third party domains only when they're present on certain first party websites—while signaling to all other first and third parties a preference not to be tracked. The Abine Firefox extension has already made some of these finer-grained header preference options available to the user.[2]

In nearly all cases, the header should be sufficient to convey the user's tracking preferences. But, situations exist where the header may fail to accurately communicate the user preference, such as if a network intermediary unexpectedly strips the header out of the request. In other scenarios, the server may simply prefer to use an alternate technical mechanism to check the user preference. For example, a site using a complicated hosting infrastructure may find it easier to detect the user's preference using client-side code, rather than at the server that initially receives the HTTP request.[3]

It may be useful for browsers to include a client-side hook, accessible via Javascript, which conveys the same user preference as the header. The W3C submission from

---

[1] Email: harlanyu@cs.princeton.edu
[2] "To Track or Not to Track? Introducing DNT+." Abine Privacy Blog, March 15, 2011. http://abine.com/wordpress/http:/abine.com/wordpress/2011/to-track-or-not-to-track-introducing-dnt/
[3] Stamm, Sid. Comment on "DOM Flag" on the Do Not Track mailing list, March 14, 2011. http://groups.google.com/group/do-not-track/msg/31df310ceb01c582

Microsoft proposes the use of a DOM property for this purpose.[4] As currently proposed, the property is a global binary variable that is set uniformly for all domains. This is sufficient when the user has chosen a blanket tracking preference, but once a user decides to fine-tune her tracking preferences, the global DOM property will no longer accurately reflect the user's choice in every case.

One requirement of such a client-side mechanism should be that it *accurately mirrors* the user's original header preference. This means that the mechanism must support the same level of granularity as the header preference allows. An undesirable scenario is one where the HTTP header signifies an opt-out preference, but the DOM property misreports either opt-in or no stated preference. The server will have received a *conflicting* user preference, and the server may well proceed to track the user despite the header opt-out.

Some users will inevitably set more granular header choices. This is bound to happen, whether through functionality implemented directly in browsers or through extensions like Abine. It's not clear that DOM properties will be able to easily and accurately mirror the more fine-grained header choices.

To implement DOM access to user tracking preferences, a single DOM attribute such as `document.doNotTrack` will likely be insufficient. A better implementation would be an access method such as:

```
document.getTrackingPreference(in DOMString domain)
```

to look up the user's tracking preference for a domain from this document.

There is one significant implementation hurdle: *access control*. The problem is that when a first party site *includes* code from a third party, whether locally or remotely, the code will run on behalf of the first party, within the first party's protection domain. Thus, when client-side code calls the access method, the browser cannot tell which entity—the first party or a third party—is trying to access the information. This means, for example, that the New York Times (the first party site) could learn that the user consents to tracking by DoubleClick but not by Quantcast on its website. Moreover, even Doubleclick could potentially learn that the user does not consent to tracking by Quantcast from the New York Times site.

Resolving the access control issue seems quite challenging to overcome in today's browsers. Browsers don't currently *tag* the origin of client-side code. Even if it did, there would be no way for browsers to distinguish actual first party code from "third party code" that is added locally on the first party site. Poorly implemented access control could lead to a form of history-stealing attacks or make it even easier to fingerprint the browser. Additionally, if it is desirable that the user tracking

[4] "Web Tracking Protection." W3C Member Submission, February 24, 2011. http://www.w3.org/Submission/2011/SUBM-web-tracking-protection-20110224/

preference is *read-write* rather than read-only (as discussed below) these access control problems become even more pronounced.

Opting-back-in and Maintaining Tracking Transparency

Another reason to potentially consider a client-side DNT mechanism is that servers may want to request that a user *opt-back-in* to tracking, inline in its Web application. A site may want to offer a special deal or premium services to an opted-out user, if the user is willing to opt-back-in to tracking. This could give sites a more flexible commercial framework to negotiate access to content or services, in exchange for tracking capabilities. Of course, any opt-back-in mechanism should carefully consider how to provide sufficient notice and obtain meaningful user consent.

But, regardless of whether a client-side mechanism is feasible, some sites may attempt to gain opt-back-in consent from users by storing the user preference server-side. Another undesirable scenario is where the user has selected the blanket preference to not be tracked, but certain entities continue to engage in tracking because the user—whether knowingly or not—has opted-back-in. Browsers would not be able to show in its interface which entities are still tracking the user.

As much as possible, tracking activities by servers should be transparent to the user. One potential remedy would be to implement a DNT *ack* header with the server's HTTP response. The ack would contain two parts. The first part just mirrors the DNT header from the HTTP request, so the user can verify that the preference was accurately received. The second part allows the server to report the user's tracking status.

For example, a DNT *ack* of "`10`" signifies two things. The "`1`" signifies that the server received a DNT:1 header in the user request. The "`0`" means that the server is still tracking the user, perhaps because the user has opted-back-in to tracking. Including an ack allows browsers to verify that DNT preferences are accurately received (and to notify the user when they are not) and to report in its interface how the user is being tracked, to the extent possible.

Separating the W3C submission on Web Tracking Protection

On a separate note, the Microsoft W3C submission on Web Tracking Protection proposes two distinct technical concepts to deal with the same issue. The first approach uses filter lists to block certain unwanted user agent requests. The second approach describes a user preference for tracking to communicate user tracking preferences to Web servers.

While both approaches strive toward a similar goal, there's no reason why they need to be considered together from a technical perspective. It may be that users will find it most beneficial to adopt both technologies simultaneously, but it need

not be this way. Each approach has its distinct strengths and weaknesses, as well as separate technical and policy challenges. Indeed, browser vendors may decide to implement one approach but not the other.

I believe it would be beneficial to the general discussion around web tracking to separate these two approaches.

Acknowledgements

Thanks to Ari Feldman at Princeton for helpful comments on this paper.