TrackMeOrNot: Enabling Flexible Control on Web Tracking

Wei Meng
Georgia Institute of Technology
wei@gatech.edu

Byoungyoung Lee
Georgia Institute of Technology
blee@gatech.edu

Xinyu Xing
Pennsylvania State University
xxing@ist.psu.edu

Wenke Lee
Georgia Institute of Technology
wenke@cc.gatech.edu

ABSTRACT

Recent advance in web tracking technologies has raised many privacy concerns. To combat users' fear of privacy invasion, online vendors have taken measures such as being more transparent with users about their data use and providing options for users to manage their online activities. Such efforts gain users' trust in online vendors and improve their willingness to share their digital footprints. However, there are still a significant amount of users who actively limit involuntarily sharing of data because vendor provided management tools only restrict the use of collected data and users worry vendors do not have enough measures in place to protect their privacy sensitive information.

In this paper, we propose `TrackMeOrNot`, a new anti-tracking mechanism. It allows users to selectively share their online footprints with vendors. With `TrackMeOrNot`, users are no longer concerned with privacy. Using it, users can specify their privacy sensitive activities and selectively disclose their activities to vendors based on their specified privacy demands. We implemented `TrackMeOrNot` on Chromium browser and systematically evaluated its performance using a large set of test cases. We show that `TrackMeOrNot` can efficiently and effectively shield privacy sensitive browsing activities.

1. INTRODUCTION

Recent advance in online tracking technologies are putting an unprecedented amount of user information into online vendors' hands. These information are surprisingly vast, from search queries to web browsing behavior to purchase history and more – and all together, they can be used to predict user preference. As such, online tracking brings many privacy concerns [20, 27].

Unarguably, online vendors are generous investors and untiring advocates of tracking techniques. Using tracking techniques to acquire more information about users, online vendors can improve their conversion rates and enable their business partners to be more economical with their advertising and marketing budgets. However, investments in tracking techniques can be severely undermined if users disable tracking and refuse to share any of their online footprints with vendors due to privacy concerns. Indeed, we have already seen a significant growth in the adoption of anti-tracking tools and software [23].

To combat users' fear of privacy invasion, recent marketing research [21, 26] provides online vendors with many suggestions, such as being transparent and simple about privacy policies, building users' trust in personal information use, and giving users options to manage the use of their shared information. Presumably, in response to these suggestions, online vendors take active measures. For example, Uber summarizes their privacy policies in an easy-to-read bulleted format that does not make users' eyes glaze over [35]. Both Google and Yahoo provide tools that allow users to manage their privacy sensitive data [25, 37].

While vendors taking care of privacy concerns build users' trust and increase users' willingness to share information, there are still a significant amount of users who actively limit involuntarily sharing of data. The reason is vendor-provided tools do not prevent them from logging a user's privacy sensitive visits but rather restricts how they use these data, meaning that the protection of such user's privacy is completely relying on vendors. Therefore, many users concern if vendors truly have proper mechanisms in place to protect their privacy sensitive information [19, 22, 24].

An intuitive solution to this conundrum is to provide users with a client side tool that shares the same functions with the tool provided by online vendors. Different from the vendor's tool, however, the client side solution provides users with a power, that is, only to share information that they are comfortable with but not disclose their privacy sensitive visits. In achieving this, this paper proposes `TrackMeOrNot`, a novel anti-tracking mechanism that prevents online vendors from tracking privacy sensitive visits specified by users. More specifically, we augment conventional web browsers with the ability to take a user's privacy demand, and shield her browsing activities based on her need.

Existing client side anti-tracking mechanisms completely impede vendors' tracking and vendors cannot obtain any information from users. Considering that online vendors also use user data to offer personalized online experience, these solutions can severely disrupt user experience. To this end, the goal of `TrackMeOrNot` is to allow users to trade the information that they are comfortable to share for a better user experience. In order to achieve such a goal, `TrackMeOrNot` has to address following two unprecedented challenges. First, `TrackMeOrNot` needs to correctly understand the semantic meaning of a web page to determine if the visit violates user's privacy demand (positive). `TrackMeOrNot` may leak many privacy sensitive page visits to vendors with a high false negative rate. On the other side, `TrackMeOrNot` may negatively affect user's

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883034.

browsing experience if lots of false positives are triggered. Our first challenge is how to build an efficient scheme that accurately determines if disclosing a visit to vendors' trackers violates a user's privacy demand. Second, to avoid exposing privacy sensitive page visits to vendors' trackers, TrackMeOrNot needs to examine if a page visit violates the user's privacy demand in advance of loading the page into a web browser. Browser interception for protecting privacy can introduce unexpected overhead, and sluggish examination can decelerate page loading and jeopardize user experience. As a result, our second challenge is how to build a lightweight interception scheme for privacy protection.

In this paper, we address the aforementioned challenges as follows. First, we introduce an efficient approach to examine a user's visit and her privacy demand. This approach decouples web page content from that of tracking parties, and excludes the unnecessary page rendering. Second, we introduce a lightweight browser interception scheme. It reduces memory consumption by maintaining only one additional browsing context for each browser tab. These two contexts allow a web browser to quickly and smoothly cloak a user's browsing session accordingly. In summary, this paper makes the following contributions.

- We propose a novel anti-tracking mechanism, TrackMeOrNot, that allows users to opt out of online vendors' trackers based on their demand and cloak their privacy sensitive browsing activities.
- We implement the prototype of TrackMeOrNot on open-source Chromium browser. TrackMeOrNot transparently supports key features of modern web browsers, including the same-tab browsing context switch and content preloading, both of which enables TrackMeOrNot to efficiently achieve the anti-tracking capability at the same time preserving user experiences.
- We thoroughly evaluate the various aspects of TrackMeOrNot. We demonstrate its efficiency and effectiveness in shielding privacy sensitive browsing activities. On average, TrackMeOrNot imposed 16.40% and 3.06% page load time and memory use overhead, respectively. In addition, TrackMeOrNot showed 0.86 accuracy in correctly satisfying user's privacy shielding needs.

The rest of the paper is organized as follows. §2 describes the background and motivation. §3 and §4 describe the design and implementation of TrackMeOrNot, respectively. §5 and §6 evaluate TrackMeOrNot's system performance and classification results. §7 discusses the related work, and §8 concludes the paper.

2. BACKGROUND AND MOTIVATION

An online vendor typically partners with many websites. Using a JavaScript snippet embedded on the partner sites, it places a persistent identifier (*e.g.*, tracking cookie) on a user's browser. Every time a user visits a vendor's partner site, the JavaScript snippet reports the visit to the vendor along with the persistent identifier associated with the user's browser. As such, the vendor can link a user's browsing activities across multiple sites, build the user profile and tailor his or her browsing experience. Figure 1 illustrates this tracking process.

Considering users may not be comfortable with disclosing all her footprints to vendors, many vendors provide web portals that allow users to opt out their unwanted footprints. The opt-out option shows vendors' respect on user privacy and have been accepted by many users. However, there are still a significant number of users who are concerned that vendors may not be able to properly protect

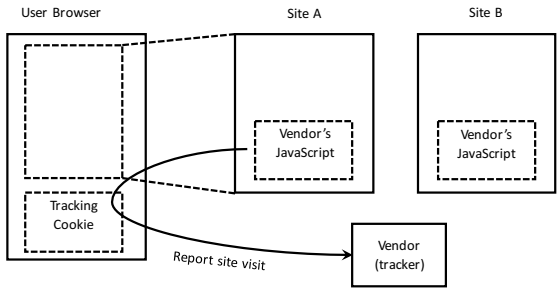

Figure 1: Online tracking workflow. When a user visits the first party website (i.e., either Site A or Site B), its content is delivered to the user's browser. To enable online tracking, this content from the first party embeds the JavaScript code uploaded by the third party (i.e., vendor). Once the JavaScript code is executed by the browser, the persistent tracking cookie will be stored in the user's browser, thereby allowing the third party to keep track of user's browsing activity.

their privacy footprints because the opt-out only restricts vendors to use unwanted data rather than completely preventing vendors from collecting unwanted data.

Existing client side anti-tracking mechanisms (*e.g.*, disabling third-party cookie or blocking tracking traffic) could completely impede vendors' tracking such that the trackers cannot collect any information from users. While providing strong protection for users' privacy, such mechanisms jeopardize user experience because – in addition to yielding profits – vendors cannot tailor user's online experiences anymore using his or her footprints in the past.

To the best of our knowledge, none of the existing controls on web tracking could protect users' privacy while preserving usability. We argue that privacy and usability do not have to be mutually exclusive and observe the need for new anti-tracking mechanisms that could balance between users' demands for privacy and usability. Such anti-tracking mechanisms should meet the following requirements: 1) user's privacy sensitive browsing activities should not be known to any vendors; 2) vendors should be able to collect data about browsing activities that have been explicitly granted by the user. In addition, any anti-tracking mechanisms should not negatively affect user's browsing experience (*e.g.*, introducing acceptable overheads such as latency, computation and memory usage).

In this work, we propose a new anti-tracking mechanism that allows users to selectively share their browsing activities with online trackers for better user experience while shielding their sensitive browsing activities. We emphasize that our goal is not only to develop algorithms that determine if a visit violates a user-specified privacy need, but also to develop a system that can effectively and efficiently decouple tracking identifiers from his or her privacy sensitive visits. Our proposed anti-tracking mechanism is mainly used for defending against stateful tracking techniques [28], such as HTTP cookies and supercookies. But, it can also be further extended with stateless tracking defense techniques, e.g., avoiding browser fingerprinting [5].

3. DESIGN

In this section, we present our design of TrackMeOrNot to support fine-grained content aware control on first-party and third-party web tracking.

3.1 Overview

As is described in §2, users have different and unique tracking privacy needs to online web tracking. To this end, we design TrackMeOrNot and depict its overall workflow in Figure 2. In order

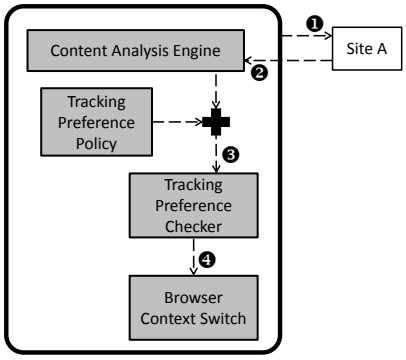

Figure 2: The overall workflow of TrackMeOrNot, where each component of TrackMeOrNot is represented as a gray box: ❶ a user tries to visit a website, Site A; ❷ Site A sends a response to a user, and TrackMeOrNot intercepts this response and forwards it to the content analysis engine; ❸ the result of content analysis engine (i.e., the content semantics) and tracking preference policy are forwarded to the tracking preference checker; ❹ TrackMeOrNot switches the browsing context only if required.

to balance between such privacy needs and usability, TrackMeOrNot employs a tracking preference policy, in which users can easily specify their own privacy needs regarding online tracking (§3.2). TrackMeOrNot separates user's sensitive browsing activities from user's normal browsing activities by leveraging isolated browsing context, which consists of all local states (*e.g.*, cache, cookie jar, local storage, *etc.*) associated with one browser instance. All navigation in TrackMeOrNot starts with an anonymous and transient browsing context. During the web navigation, TrackMeOrNot analyzes the semantic meaning of the target web contents (§3.3) to interpolate its analysis results with a privacy policy. Then TrackMeOrNot leverages such learned information to determine which browsing context (persistent or anonymous) it should be running to meet the user's tracking preference policy (§3.4). Finally, TrackMeOrNot carries out seamless browsing context switch based on the switching decision (§3.5).

3.2 Tracking Preference Policy

Following the general and well-known beliefs in user's privacy protection, TrackMeOrNot allows users to specify their privacy needs on their own. These privacy needs in general can be expressed with the well known access control concepts, blacklists and whitelists. Leveraging these basic schemes, TrackMeOrNot introduces two different types of protection entities, web content category and domain name. Web content category offers the capability to specify user's privacy needs from a bag of category words. For example, a user can enlist the category *drug* into the blacklist of the web content category, if the user wants to hide the fact that she or he has browsed a web page related to drugs. On the other hand, *drug* can be whitelisted if the user does not feel those contents are privacy sensitive and thus wants to receive useful services based on those contents (e.g., targeted advertisements from third parties). Moreover, the other protection entity, domain name, allows users to easily specify their privacy needs using the domain name itself. For example, a user can whitelist *cnn.com*, if she/he believes the contents offered by *cnn.com* are not privacy sensitive and thus allow all third-party trackers loaded on *cnn.com* to trace her/his visits.

Similar to access control systems, TrackMeOrNot also defines override rules on privacy policies. A user could assign a high priority to a whitelist rule so that TrackMeOrNot disregards blacklist rules with lower priorities if the whitelist rule is ever matched. If two rules share the same priority, a blacklist rule always precedes a whitelist rule. TrackMeOrNot supports the following three custom

```
{
    "category-blacklist": {
        "drugs": "high"
    },
    "category-whitelist": {
        "sports": "low"
    },
    "domain-blacklist": {
        "aaa.com": "medium"
    },
    "domain-whitelist": {
        "bbb.com": "medium"
    },
    "fallback browsing mode": {
        "anonymous"
    }
}
```

Figure 3: tracking preference policy for TrackMeOrNot. TrackMeOrNot uses universal and simple JSON format so that it can be supported from many different platforms.

priorities for each policy rule that users can specify: *high*, *medium*, and *low*, where *high* and *low* indicate the highest and lowest priority respectively. For example, suppose the user specified *drug* as a category blacklist rule with a *medium* priority and *cnn.com* as a domain whitelist rule with *high* priority, TrackMeOrNot will first apply the policy on *cnn.com* and then disregard the policy on *drug* if the policy on *cnn.com* is matched. By default, all rules have the same priority (medium) if the user does not define explicitly.

Furthermore, users can specify a fallback browsing mode, which specifies which browsing context would be used in case none of the policies are matched. A privacy conscious user may select *anonymous* as her or his fallback browsing mode and specify only a few web content categories and domains as whitelist rules. On the other side, a user who is more concerned with usability (*e.g.*, the quality of personalization service) may use *normal* as her or his fallback browsing mode and enlist those categories and domains that she or he determines as sensitive into blacklist. We note that how a user defines her or his tracking preference policy impacts the performance of TrackMeOrNot as we will discuss in §6.

Figure 3 depicts an example of the privacy preference policy that TrackMeOrNot accepts. Each protection entry (i.e., either category or domain) has both blacklist and whitelist policies. In each policy, the protection entity and priority are specified as a key/value pair. Moreover, a `fallback browsing mode` is specified as `anonymous` in this example, meaning that the anonymous browsing context would be used as the fallback mode.

3.3 Content Analysis Engine

In order to determine whether a target website that a user is visiting is against any blacklist policies, TrackMeOrNot intercepts web navigation processes to analyze web contents that the browser receives and renders. In particular, TrackMeOrNot first intercepts all the network responses from the first party, and then it starts the content analysis with the built-in machine learning classifiers to obtain the representative category of the responses.

TrackMeOrNot collects all the network responses by intercepting the browser's event on completing the fetching of a resource. TrackMeOrNot particularly focuses on collecting the responses only from the first party for the following reasons. First of all, most of the content semantics are delivered by the first party (i.e., the third party mostly delivers external data including images, tracking scripts, or advertisements), and these semantics are what users are concerned about from the privacy point of view. Furthermore, it would require too much analysis time if TrackMeOrNot also considers the resources from the third party as well because the loading of third party resources can take quite long time. Again, the resources from

the third party are mostly related to external data, which usually have a significantly bigger size than those from the first party.

Once such an event is fired, TrackMeOrNot forwards the corresponding response contents to a content analysis engine. The analysis engine first parses responses using its own HTML parser. Since the HTML parser provided by the underlying browser aims at rendering the complete web page from the responses, it is not designed to parse incomplete responses. Thus, TrackMeOrNot implements a custom HTML parser, which is capable of handling such incomplete responses and focuses on extracting text semantics. Specifically, this parser constructs the DOM tree without sending new network requests nor evaluating JavaScript and Cascading Style Sheet, each of which is not related to the content semantics. Next, TrackMeOrNot scans the DOM tree and extracts texts from it. Non-content texts (i.e., navigation links and advertisements) are excluded from the final result as those are not related to the real content semantics.

TrackMeOrNot then utilizes pre-built machine learning classifiers to summarize the extracted text contents. For example, a classifier may predict the likelihood that the current web page contains pornography content. Another classifier may predict whether the web page is about education. The summarized classification results (probabilities) are then sent to the central controller for checking with user tracking preference, which we describe in the next subsection. If the built-in machine learning classifiers are not satisfactory to some users, they could also select some free online classification services for processing their navigation request before the navigation request is sent in TrackMeOrNot. The correct browsing context could be directly decided after checking the returned results with user privacy preference. However, the users have to build their privacy protection on trust of the third party service providers and may suffer from unpredictable browsing latency.

While the content analysis engine is processing the content semantics, the page loading process is not blocked so that TrackMeOrNot introduces minimum overhead in page load time if TrackMeOrNot does not switch the browsing context.

3.4 Tracking Preference Checker

Once the content analysis is finished, TrackMeOrNot checks the privacy sensitivity of the target website. In other words, using the tracking preference policies and the target website's content semantics, TrackMeOrNot determines whether it needs to keep using anonymous browsing context to prevent tracking or to switch to persistent browsing context to augment usability.

As described in §3.2, TrackMeOrNot checks each policy rule in the order of associated priority. For example, suppose a user specified *drugs* in category blacklist as depicted in Figure 3. And further suppose that the content analysis engine returned the target website has the semantics, drugs. In this case, TrackMeOrNot determines that the browsing context does not need to be changed. However, if drugs were specified in category whitelist, TrackMeOrNot will switch to the persistent browsing context, which we describe in detail in the next section.

3.5 Browsing Context Switch

Based on the current browsing context and the decision made after checking user's tracking preference, TrackMeOrNot may switch to the persistent browsing context for the current navigation request. If a browsing context switch is not needed, TrackMeOrNot simply does nothing, meaning that the browsing context would be stayed in the anonymous one. Otherwise, TrackMeOrNot terminates the current pending navigation originated from the initial anonymous browsing context, and restarts the navigation with the persistent browsing context.

TrackMeOrNot will also mark the new navigation request so that it will not be intercepted again. Since the machine learning classifiers cannot be 100% accurate, sometimes TrackMeOrNot may switch to a browsing context that the user does not want. In this case, the user could manually switch back to the correct browsing context.

3.6 Discussion

By isolating privacy sensitive browsing activities in anonymous browsing contexts, TrackMeOrNot effectively prevents online vendors from linking those activities with a user's persistent profile. However, a user may still feel unsafe because other non-local state features may be used to track the user's browsing activities as well. For example, stateless fingerprinting does not use any local states of a browsing context to identify a browser instance. Such stateless tracking threat has already been addressed in [29]. The solutions could be integrated with TrackMeOrNot by generating a temporary fingerprint when using the temporary browsing context. IP anonymity technologies [16] could similarly be integrated with TrackMeOrNot to defend against IP address based tracking.

Another limitation of TrackMeOrNot is that in an anonymous browsing context TrackMeOrNot is not able to extract the content of a few websites that only provide services after the user has logged in. For example, Facebook does not provide its service if the user does not sign in with her or his account. As a result, the user has to explicitly specify a whitelist rule for Facebook if the user wants to use its service, or the user needs to log in her/his account of Facebook in the anonymous browsing context as well. The current design of TrackMeOrNot may also allow user's browsing activities on websites like Facebook to be tracked by other third-party trackers, if Facebook and other websites explicitly embed tracking scripts of other vendors into their own websites. However, such behavior is essentially the same as directly sharing first party data with third party trackers, which cannot be prevented if the user has agreed with the terms and conditions of these websites that explicitly indicate that user data will be shared with third-parties.

4. IMPLEMENTATION

To demonstrate the feasibility and effectiveness of TrackMeOrNot, we implemented a prototype of TrackMeOrNot based on one of the most popular modern browsers, Chromium (version 45.0.2426.3). Although we only implemented a prototype on Chromium, the design of TrackMeOrNot is generic and can be easily extended to other browser platforms. In terms of implementation complexity, the Chromium version of TrackMeOrNot introduced 1,400 new lines of code (200 LoC for navigation interception, 700 LoC for content analysis engine, 500 LoC for tracking preference and browsing context switching) to Chromium.

In the rest of this section, we first introduce the general architectural design of the Chromium browser, and then describe how each component of TrackMeOrNot is implemented along with Chromium's design.

4.1 Chromium Browser's Architecture

Chromium uses a multi-process architecture [11] to utilize process-level isolation between the browser process and the renderer process. The browser process of Chromium is the main process that manages UI, I/O, tabs, configuration, etc. The renderer process renders the web page using the WebKit layout engine, and multiple renderer processes are associated with and controlled by the browser process. At the time of creation, the browser process and renderer processes

are strictly bound with a certain browsing context (either persistent or temporary browsing context), and the current design does not allow to switch between them after being created.

4.2 Tracking Preference Policy

`TrackMeOrNot`'s tracking preference policy leverages existing preference system in Chromium [13], which is managed by the browser process. While `TrackMeOrNot` internally reuses Chromium's preference system, it also provides an interface for users to easily configure her/his tracking preference through Chromium's browser setting interface itself (*i.e.*, we added the `Tracking` section into the `Content settings` option of Chromium's `Privacy` setting). This tracking preference is loaded into a browser process when the Chromium browser is launched, and will be used later to enforce tracking policy based on the content analysis results.

4.3 Content Analysis Engine

The content analysis engine is implemented inside WebKit in the renderer process. We intercept each navigation request inside the WebKit layout engine. In our current design, we only intercept network responses of first-party contents in the ResourceFetcher, as is discussed in §3.3. The network requests for third-party contents are temporarily held inside the DOM parser of WebKit when a new navigation starts. The hold is released when the renderer receives a notification from the browser process, or is terminated with the navigation request if the browser process switches to a different renderer process with a different browsing context. If a new renderer process is selected for restarting the navigation, the browser process will set a flag in that renderer process so that the new navigation request will not be intercepted again.

In order to understand the semantic meaning of a web page, `TrackMeOrNot` needs to parse the corresponding HTML source and extract the content from it. However, the DOM parser of WebKit might be blocked by network requests that are held [36], so that we are not able to extract all the text contents with the DOM parser. To overcome this problem, we implemented a lightweight DOM parser with the *libxml2* library [17]. Our DOM parser creates the DOM tree from the HTML source without loading new resources and evaluating JavaScript and CSS. After the DOM tree is built, we extract all the text on the web page. However, the extracted text may contain many non-content text, such as navigation links, copyright disclaimer, or advertisements. Such non-content text may introduce noise in the classification procedure. We implemented the CETD algorithm [34] inside WebKit to further remove non-content text.

A key feature of `TrackMeOrNot` is to use machine learning algorithms for content analysis. While there are many online classification web services, we decided to implement local classification models for privacy and performance concerns that have been discussed in §3. Due to the diversity of web pages on the Internet today, we had to train the machine learning models with a large set of diverse web pages. However, to the best of our knowledge none of the widely used web page classification benchmark data sets [32] could meet our requirement because they either contain very few documents that are not diverse enough or use coarse-grained labels. We eventually selected the well known AOL query logs [31] for training our classification models. The AOL query logs include 21 million search queries from 650k real users. The logs also contain over 19 million user click-through events of 1.6 million unique web pages, which had been visited by real users and represent diverse human interests. Some sensitive contents that people generally do not want to be tracked (*e.g.*, pornography) are also included in the data set.

Although the AOL data set is a good fit for our study, unlike other web classification data sets the AOL data set is not labeled. It is impractical to manually label all the URLs in the data set. We used the natural language processing API of AlchemyAPI [14] to label the English web pages that were still accessible in September 2015. Each web page was pre-processed with the CETD algorithm to remove non-content text before calling the API. AlchemyAPI classifies each web page into topic category up to five levels deep [15]. In total, we were able to label 369,179 web pages with the support from AlchemyAPI [1].

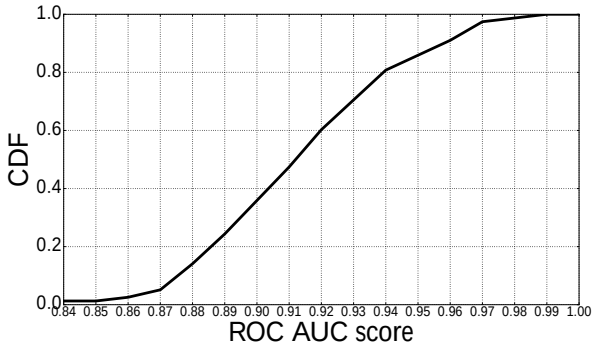

Figure 4: The ROC AUC score distribution of binary classifiers.

Since some categories have much more web pages compared with other categories, we further broke down web pages in these popular categories into their second level category. For example, "art and entertainment" web pages were further broken into "books and literature", "movies and TV", "music", "shows and events", "visual art and design" and "arts/other". We also manually selected all (second level) categories that are generally considered as sensitive, *i.e.*, "finance", "health and fitness/{disease, disorders, drugs}", "law, govt and politics/{armed forces, government, law enforcement, legal issues}", "society/{crime, dating, sex}". Eventually, we got 78 categories in our data set. This labeled AOL data set was also used in our evaluation of `TrackMeOrNot` in §6.

For each of the 78 categories, we built a binary classification model using Linear SVC. The term frequency-inverse document frequency (tf-idf) of each term in the extracted text is used as feature. All the web pages in a given category are positive samples. Equivalent number of randomly selected web pages of other categories are negative samples. The positive samples and negative samples were randomly and evenly split into a training set and a test set. We used 10-fold cross validation in the training set to learn the best parameters for each category, and used the test set for evaluation. The distribution of the ROC AUC scores of the 78 categories are shown in Figure 4. The minimum, mean, maximum, and standard deviation of ROC AUC scores are 0.84, 0.92, 0.99, 0.03, respectively.

The prediction methods of the trained models are implemented inside WebKit. For each navigation request, the extracted text is evaluated with all the classification models. The prediction confidence (decision function in the case of LinearSVC) of each binary classifier is gathered to select the best label(s) for the web page. In our implementation, the label of the classification model that predicts positive class with highest confidence is assigned to the web page. The final prediction result is then sent back to the browser process for tracking preference check. Since the topic of web pages is very subjective, we plan to enable the users to use their customized classifiers for their own need in the future.

[1]We would like to thank AlchemyAPI for granting us special academic license in this study.

4.4 Tracking Preference Checker

Given the content analysis result returned from the renderer process and the user specified tracking preference, this phase determines whether the current navigation leads to content that is privacy sensitive to the user. Based on the result, TrackMeOrNot decides whether to switch the browsing context or not. This decision procedural follows the algorithm as illustrated in §3.2. If TrackMeOrNot determines that the browsing context should not be switched (i.e., keep using the anonymous browsing context), TrackMeOrNot notifies the corresponding renderer process to continue rendering the current web page and does not perform any additional operations because the renderer process is already handling the navigation using the anonymous browsing context. On the other hand, if it has to switch to the persistent browsing context, then TrackMeOrNot will terminate the current navigation request that is being processed in the renderer process and then switch the browsing context as we describe more details in the next subsection.

4.5 Seamless Browsing Context Switch

In Chromium, one browser process is bound with one fixed browsing context. As a result, tabs of different browsing contexts (users) cannot be merged within one browser window. A naive implementation of TrackMeOrNot will pop-up a new browser window each time the browsing context is switched, which is very annoying to users. To preserve good user experience, we break the binding between one browsing context and one browser process in Chromium. TrackMeOrNot allows one browser process to be associated with multiple browsing contexts so that the browser process could create and manage renderer processes with different browsing contexts. All browsing context switching in our implementation are seamlessly done in the same browser tab without annoying the users. To make the browsing context switching transparent to the users, we also switch the theme of the browser UI when switching the browsing context so that the user could easily tell which browsing context she/he is browsing with. If the user ever wants to use a different browsing context for a navigation, she/he could click a "switch" button on the navigation bar to manually switch the browsing context.

5. SYSTEM PERFORMANCE EVALUATION

In this section, we present the system performance evaluation of TrackMeOrNot. A vanilla build of Chromium (version 45.0.2426.3) was used as the baseline system for comparison. We measured the web page load time and peak memory usage of the two browsers to show the impact of TrackMeOrNot on browser performance and user experience. All experiments were run on Debian Jessie (Linux Kernel 3.16) with a quad-core 3.20 GHz CPU (Intel Xeon W3565) and 24 GB RAM.

We selected the Alexa top 100 US websites as the data set for system performance evaluation. Specifically, the two browsers (i.e., the vanilla Chromium and TrackMeOrNot) sequentially visited the main page of the top 100 websites three times. Note that some top websites (*e.g.*, googleusercontent.com) can not be directly visited from the browser. Thus, we selected the next top websites to fill the places of such websites. We report the average of the three measurements in all the following results.

Depending on the user tracking preference and the web page the user is navigating to, TrackMeOrNot can exhibit two different browsing behaviors: 1) staying in anonymous browsing context; and 2) switching to persistent browsing context. Intuitively, switching to persistent browsing context in the navigation would impose more runtime overheads in terms of both page load time and memory usage. To clearly understand how much more overheads are imposed in TrackMeOrNot, we configured two user tracking preferences for each of the web page to instruct TrackMeOrNot to either stay in the current browsing context (*Content Analysis Only Configuration*, or *CAOC*) or switch to a different browsing context (*Browsing Context Switching Configuration*, or *BCSC*) in the three visits.

5.1 Page load time

Regardless of user tracking preferences, TrackMeOrNot has to always perform content analysis and tracking preference check, which would result in navigation latency. Additionally, TrackMeOrNot may reload the web page using a different browsing context based on the result of tracking preference check that further extends the page load time. In order to precisely measure such extra latency, we first implemented internal hooks in various critical event handlers in Chromium (*e.g.*, page load completion event), each of which measures a clock in nano-second precision using `clock_gettime()`.

Figure 5 and Figure 6 present the result of main page navigation in case of CAOC and BCSC, respectively. The page load time overhead is the ratio of the extra load time introduced by TrackMeOrNot to the complete page load time using the vanilla Chromium.

When configured with CAOC, TrackMeOrNot introduced negligible extra latency - 0% to 10% overhead in the page load time with 1.93% as mean and 1.81% as standard deviation. We believe the content analysis engine of TrackMeOrNot is very efficient and would not disrupt the user's navigation experience in practice for the CAOC case, because minimum, average, maximum and standard deviation of extra processing time were only 1.00 ms, 39.56 ms, 232.00 ms and 37.40 ms, respectively.

When configured with BCSC, TrackMeOrNot needs to restart the navigation with the persistent browsing context, thereby incurring more latency to page load time. From our evaluation, the overhead varied significantly: the minimum, average, median, maximum and standard deviation of load time overhead were 1.00%, 16.40%, 13.00%, 54.00% and 12.92%, respectively. The extra load time (minimum: 51.00 ms, mean: 340.33 ms, median: 228.00 ms, maximum: 2130.00 ms, standard deviation: 352.60 ms) is inconsistent with the overhead, because the raw page load time itself using the vanilla Chromium varied a lot. For example, the vanilla browser loaded www.google.com using 451 ms where TrackMeOrNot spent 198 ms in resending the request to www.google.com (including content analysis), imposing 44% (198/451) overhead in page load time. On the other hand, TrackMeOrNot only took 148 ms to switch browsing context for www.nytimes.com where the complete page of www.nytimes.com needed 5,539 ms to load using vanilla browser, resulting in only 3% (148/5539) overhead in page load time. Considering that half of the main pages of US top 100 websites needed more than 2,145 ms (median) to load using vanilla Chromium, we believe the delay in page loading (median: 228 ms) caused by TrackMeOrNot would not be observable to normal users.

To better understand the latency introduced when using BCSC, we also recorded the time to load the main frame HTML source using the vanilla Chromium for each web page. We present the side-by-side comparison of main frame HTML source load time of vanilla Chromium and extra page load time of TrackMeOrNot in Figure 7. As is evident from the figure, the extra page load time closely matches with the time needed for loading the HTML source of main frame. If the full page load time is not significantly higher than the main frame HTML source load time, then TrackMeOrNot will lead to very high overhead in page load time. However, TrackMeOrNot could be enhanced by caching the main frame HTML source that is loaded in the previous browsing context and sharing the cached HTML source with the new renderer process. Thus no extra request needs to be sent, which could significantly

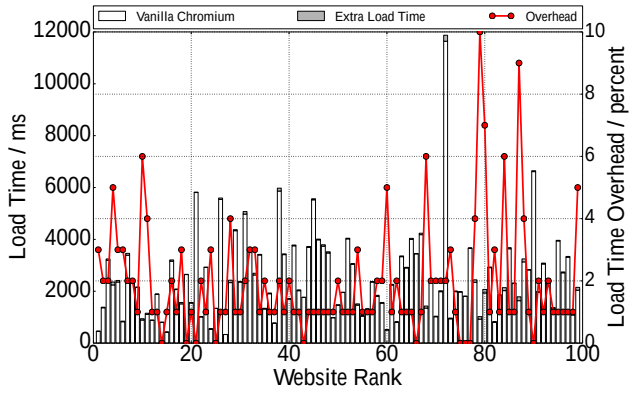

Figure 5: Page load time when visiting each of Alexa top 100 US websites under Content Analysis Only Configuration (CAOC), i.e., when TrackMeOrNot does not switch the browsing context. Overall, TrackMeOrNot imposed 39.56 ms (1.93%) extra load time on average compared to vanilla Chromium.

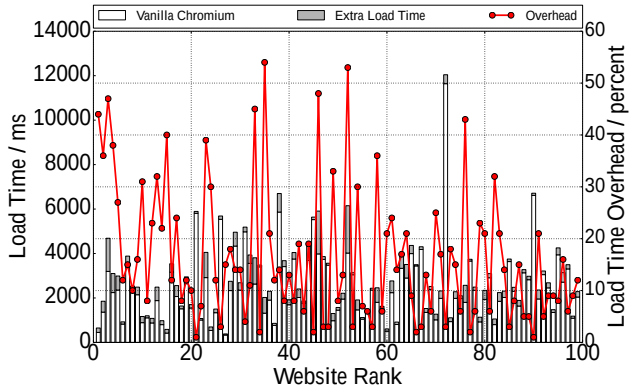

Figure 6: Page load time when visiting each of Alexa top 100 US websites under Browsing Context Switching Configuration (BCSC), i.e., when TrackMeOrNot performs switching to the persistent browsing context. Overall, TrackMeOrNot imposed 340.33 ms (16.40%) extra load time on average. Although this extra overhead may seem significant, this would not be easily observable to users in practice due to the natural inconsistent network latency.

reduce overhead in page load time introduced by TrackMeOrNot. We leave this implementation optimization as our future work.

5.2 Memory

TrackMeOrNot may require more memory in runtime because it includes 78 classification models (including a large vocabulary of features) in our implementation. In addition, if a browsing context switch is requested, TrackMeOrNot needs to create new renderer process which may also increase its memory usage. For these reasons, we measured the peak memory usage of TrackMeOrNot and vanilla chromium for 15 seconds in each of the 3 visits to a web page.

Figure 8 and Figure 9 show the peak memory usage of vanilla Chromium and the extra peak memory usage of TrackMeOrNot when visiting the main pages using Content Analysis Only Configuration (CAOC) and Browsing Context Switching Configuration (BCSC), respectively. The memory overhead is the ratio of extra peak memory usage of TrackMeOrNot to the peak memory usage of vanilla Chromium. For both configurations, the memory overheads are between -15% and 22%. The means are 3.06%

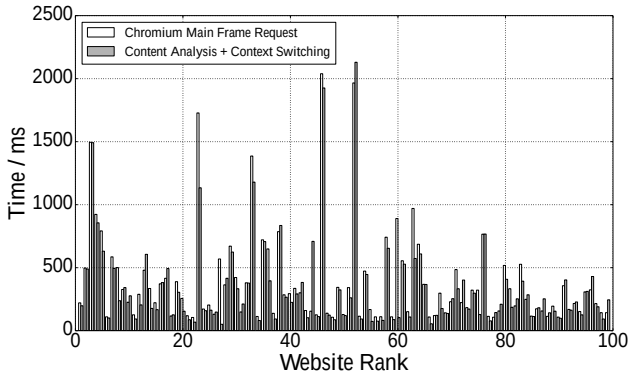

Figure 7: Main frame HTML source load time (marked as white boxes) *v.s.* extra page load time in TrackMeOrNot (marked as black boxes) when visiting each of Alexa top 100 US websites.

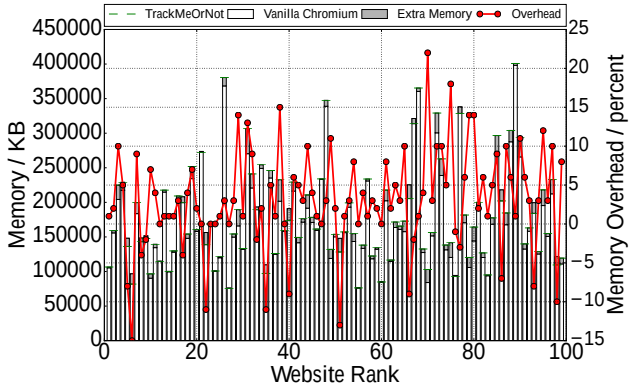

Figure 8: Peak memory usage when visiting each of Alexa top 100 US websites under Content Analysis Only Configuration (CAOC). TrackMeOrNot imposed 3.06% overhead on average.

(CAOC) and 1.68% (BCSC), respectively. We observe that sometimes TrackMeOrNot consumed less memory than vanilla Chromium, which might be attributed to that smaller dynamic contents were loaded when using TrackMeOrNot to visit those web pages. Similarly, the memory over consumed by TrackMeOrNot may also due to the dynamics of web contents. As a result, TrackMeOrNot did not significantly require more memory than the vanilla build.

6. ANTI-TRACKING EVALUATION

As is mentioned in §3, TrackMeOrNot allows users to specify unwanted page visits based on web content category. Our implementation of TrackMeOrNot includes 78 classifiers for categorizing the web pages that users visit. TrackMeOrNot compares the output of the classification models with user specified needs and switch a browsing session between different browsing contexts. In this section, we demonstrate how effectively TrackMeOrNot uses the classifiers to conceal users' privacy sensitive visits. In particular, we first describe our experimental design. Then, we evaluate how effectively the classifier satisfies user's privacy needs.

6.1 Experimental Design

To evaluate the effectiveness of TrackMeOrNot on hiding sensitive web browsing activities – such as browsing pornographic web pages and health related forums – we need to simulate a series of web page visits and then examine if TrackMeOrNot can accurately

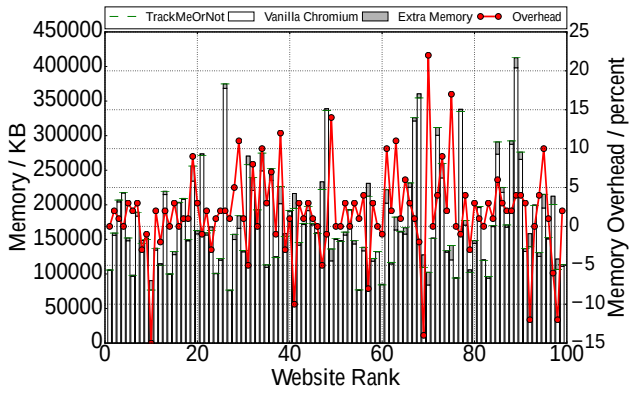

Figure 9: Peak memory usage when visiting each of Alexa top 100 US websites under Browsing Context Switching Configuration (BCSC). `TrackMeOrNot` imposed 1.68% overhead on average.

conceal user specified privacy sensitive visits when visiting these web pages. As a user may specify any category of visits as privacy sensitive browsing activities, we need a sequence of page visits that covers all spectrums of categories. We selected web pages that had not been used for training any classifier from the AOL data set discussed in §4 as our evaluation web page corpus, which contains 43,807 English web pages in 78 unique categories.

To examine the effectiveness of `TrackMeOrNot` on hiding privacy sensitive visits, we also need to know users' needs to browsing privacy. In other words, we need to know the page categories in which a user is (or not) willing to disclose his or her visits to vendors. To obtain the users' needs to browsing privacy, we conducted an online survey through Amazon Mechanical Turk. The survey was approved by the IRB of our institute. We presented all 78 categories to participants and asked them to choose the page categories in which they are not comfortable to disclose their visits. In addition, participants were asked to choose the page categories in which they are comfortable to share their visits. Ultimately, we collected 145 valid responses. The demographic distribution of the 145 subjects is shown in Table 1. Table 2 and Table 3 show the top categories of web pages that users specified as blacklist rules or whitelist rules, respectively. In other words, they represent the page categories on which users want to hide or disclose their browsing activities from or with online vendors. From our collected questionnaires, we found 14 participants who expressed strong privacy concern and did not want to share any of their visits with vendors. We also observed 2 participants who stated that they were not concerned with privacy and wanted to offer all of their footprints to vendors. Overall, we obtained 129 unique privacy needs from all the participants.

For each unique privacy need, we encode it by converting it into blacklist and whitelist rules as discussed in §3.2. We illustrate the number of blacklist and whitelist rules across 129 unique privacy needs in Figure 10. Note that, for those page categories that a user does not specify as *"comfortable to reveal"* or *"reluctant to disclose"*, we convert them into either whitelist rules or blacklist rules by assuming the users had specified *persistent* or *anonymous* as their fallback browsing context, respectively. Thus, we have two different rule sets across 129 unique privacy needs (see Figure 11 and Figure 13).

Including two special whitelist and blacklist rule pairs that indicate *"do not disclose any visits" and "reveal all visits"*, we configure `TrackMeOrNot` using the 256 whitelist and blacklist rule pairs shown in Figure 11 and Figure 13. Then, we simulate the visits to the aforementioned 43,807 web pages using `TrackMeOrNot`. We examine the accuracy of hiding blacklist visits and disclosing whitelist visits. In addition, we study the False Negative Rate (FNR) and False Positive Rate (FPR) of our `TrackMeOrNot`. Specifically, a page that needs to be browsed in an anonymous browsing context is a positive sample in our evaluation. The false negative rate is the ratio of number of false negatives (incorrectly predicted as negative) to the number of positives (including false negatives and true positives). The false positive rate is the ratio of number of false positives (incorrectly predicted as positive) to the number of negatives (including false positives and true negatives). The system would mistakenly disclose many privacy sensitive visits to vendors when the false negative rate is high. Similarly, the system may hide many visits that the user feels OK to share with vendors when the false positive rate is high. We present the evaluation results in the next subsection.

Table 1: Distribution of demographics of survey subjects.

Age	18-24	25-34	35-44	45-54	55+
	8 5.52%	58 40.00%	34 23.45%	29 20.00%	16 11.03%

Gender	Male	Female
	60 41.38%	85 58.62%

Education	High school or less	Associates	Bachelor or higher
	32 22.07%	25 17.24%	88 60.69%

Table 2: The top-10 page categories in which users do not want to disclose their visits to vendors.

Category	% of votes
society/sex	78.67
finance	58.67
society/dating	58.67
health and fitness/disorders	52.67
society/crime	51.33
law, govt and politics/armed forces	50.00
law, govt and politics/legal issues	50.00
religion and spirituality	47.33
law, govt and politics/government	47.33
health and fitness/disease	47.33
law, govt and politics/law enforcement	46.00

6.2 Evaluation Result

Figure 12 and Figure 14 show the performance of `TrackMeOrNot` in terms of accuracy, false positive rates and false negative rates for persistent and anonymous fallback tracking preferences, respectively. We observed that `TrackMeOrNot` achieved 0.86 accuracy in hiding and disclosing user visits on average for both persistent and anonymous fallback tracking preferences. The minimum accuracies were 0.69 and 0.74 for the two different settings, respectively, where the maximum accuracy in both settings was 1.00. The results indicate `TrackMeOrNot` can effectively cloak user specified privacy sensitive visits and disclose a certain amount of footprints regardless of the fallback browsing context.

We observed some interesting patterns of false positive rates and false negative rates in regards to the number of blacklist rules in the tracking preference. As is evident in Figure 12 and Figure 14, the false positive rates increased when a user specified more page categories as his or her blacklist rules. On the other side, we also observed the decrease in false negative rates as more categories were specified in blacklist. The reason behind these patterns are that with

Figure 10: The number of whitelist and blacklist rules across 129 distinct privacy needs.

Table 3: The top-10 page categories in which users are comfortable to share their visits with vendors.

Category	% of votes
arts and entertainment	64.00
food and drink	54.00
hobbies and interests	50.67
sports	45.33
pets	45.33
shopping	44.00
travel	42.67
home and garden	42.67
news	42.00
education	39.33

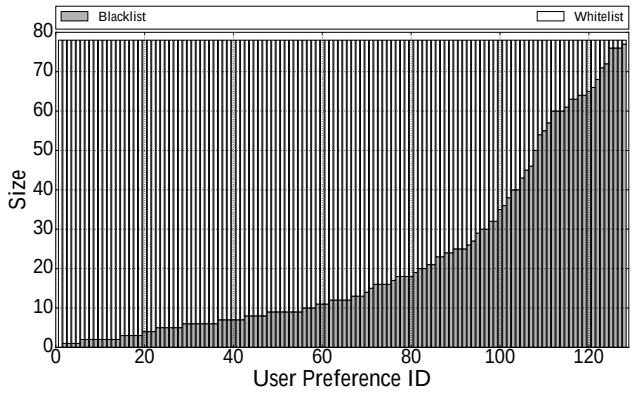

Figure 11: Tracking preferences using persistent fallback browsing context.

more categories that are specified as blacklist rules, `TrackMeOrNot` relies on more binary classifiers to examine blacklist visits, and consequently a web page is more likely to be classified as positive.

The average false negative rates of `TrackMeOrNot` were about 0.17 and 0.12 and average false positive rates were about 0.24 and 0.36 when using persistent and anonymous fallback browsing context, respectively. As is shown in Figure 12 and Figure 14, the false negative and false positive rates are complementary. For users concerned more with privacy than personalized user experience, they may configure `TrackMeOrNot` to achieve a low false negative rate and a high false positive rate by specifying more blacklist rules. In contrast, users may configure `TrackMeOrNot` with a high false negative rate but a low false positive rate by trading her or his need for privacy for better usability.

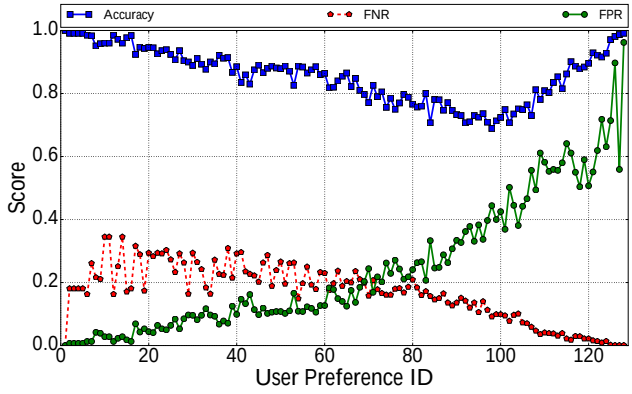

Figure 12: Evaluation results on 129 tracking preferences with persistent fallback browsing context.

7. RELATED WORK

In this section, we summarize various tracking technologies and analyze existing anti-tracking mechanisms.

Defense against HTTP cookie tracking. HTTP cookie tracking is a stateful tracking technology. It stores in a user's browser a piece of data (including a unique identifier) set by an online vendor while the user is browsing a website that contains the vendor's content. The cookie is automatically sent to the vendor whenever the user visits a website that contains the vendor's content in the future. It can be used to analyze the user's web browsing behavior. Tracking cookies are widely adopted by advertising networks for the purpose of serving up "interest-based" or "behaviorally targeted" ads. To stop them from tracking a person's surfing habits, companies and non-profit organizations (*e.g.*abine [1], NAI [12] and DAA [7]) implement a cookie opt-out mechanism which enables users to block and prevent the advertising network from installing future tracking cookies. A recent study demonstrates many drawbacks of this approach including poor usability and unreliability [28]. As an alternative approach, user self-help anti-tracking tools are developed and implemented [2, 8, 33]. They defend trackers by blocking HTTP requests to corresponding vendors. For example, Adblock plus [2] and Ghostery [8] impede HTTP requests to advertising networks, and thus browsers cannot report user footprints to the advertising networks. Both cookie opt-out and self-help anti-tracking tools are designed for defending HTTP cookie tracking. Considering a number of online vendors have been discovered using advanced technologies to track users [4], the effectiveness of these approaches wanes. In this paper, our proposed anti-tracking mechanism not

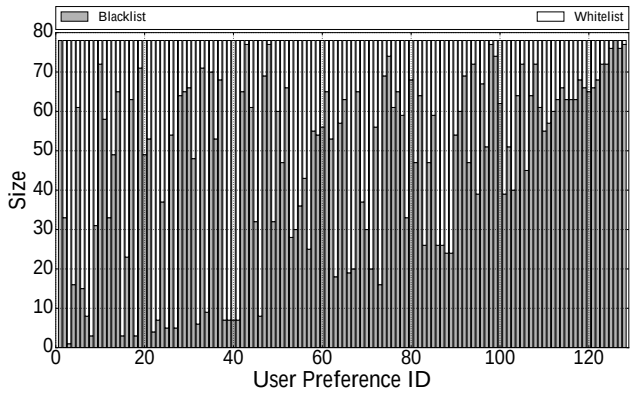

Figure 13: Tracking preferences using anonymous fallback browsing context.

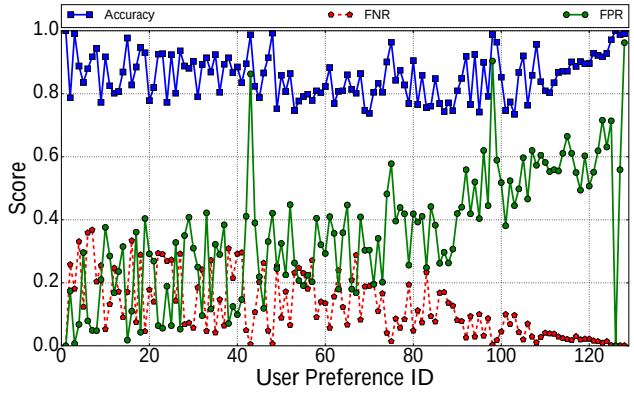

Figure 14: Evaluation results on 129 tracking preferences with anonymous fallback browsing context.

only can impede HTTP cookie tracking, but also defend many other previously known stateful tracking technologies, e.g., supercookies.

Defense against Supercookies. Apart from HTTP cookie tracking, another stateful tracking technology is supercookie tracking. Using this technology, online vendors can encode a globally unique identifier – supercookies – into a web browser. For example, vendors can abuse HTML5 local storage feature [9] or Flash Local Share Object [10] to store a unique identifier on a user's hard drive for tracking the user's digital footprints later on. To the best of our knowledge, there are only two approaches that can be adopted to impede such advanced tracking technologies. Private browsing [3] is one of these approaches, which does not store any local data that could be retrieved at a later date. Using private browsing, a user can therefore prevent vendors from using supercookies to track her surfing habits. Another defense against supercookies is TrackingFree [30], an anti-tracking browser that can impede supercookie tracking practice by partitioning a user's visits into multiple isolation units based on URL. With this isolation, online vendors can still store supercookies but not correlate a user's browsing activities across websites. One problem of this approach is that the system overhead linearly increases when a user browses more websites because TrackingFree maintains an isolated browser state for each unique website and OS needs to allocate a new memory space for each isolated browser state. Considering unexpected memory consumption can potentially jeopardize user experience, our proposed anti-tracking mechanism follows a lightweight design principle which constructs in-memory isolation units based on browser tabs. In addition, our proposed anti-tracking mechanism goes beyond the aforementioned two approaches by allowing users to instruct web browser to selectively block tracking activities. In addition to boosting profits, online vendors use information collected to personalize user experience. From the usability perspective, our anti-tracking mechanism therefore allows users to enjoy customized browsing experience without worrying about privacy invasion, while existing defense restrict data sharing completely and users cannot obtain any benefits from personalization.

Defense against Stateless Fingerprinting. Different from the stateful tracking technologies discussed above, a new class of web tracking technologies can use stateless information to identify users and report their surfing habits. This new class of tracking technologies neither stores nor retrieves data on user's hard drive. Instead, it tracks a user by learning properties of her web browser and forming a unique or nearly unique identifier (*i.e.*, fingerprinting) [5]. To counteract such a tracking mechanism, anti-tracking technologies focus on making browser fingerprints non-deterministic across multiple browsing sessions [6, 18, 29]. This makes vendors difficult to link a user's multiple visits. For example, Nikiforakis *et al.* designed and developed PriVaricator [29] that utilizes the power of browser fingerprint randomization to break vendors' ability to connect the same fingerprint across multiple visits. Our anti-tracking mechanism is orthogonal to defense against stateless tracking. In this paper, we focus on building an anti-tracking mechanism that impedes stateful tracking based on user demand.

8. CONCLUSION

In this paper, we present `TrackMeOrNot`, a new anti-tracking mechanism that allows users to selectively sharing their online footprints with vendors for better user experience while shielding privacy sensitive browsing activities from online trackers. `TrackMeOrNot` provides a user with two browsing contexts – anonymous and persistent context – and a web browser can switch a user's browsing session between the contexts based on user specified privacy needs. We demonstrate how a user can specify his or her privacy need and employ `TrackMeOrNot` to surf the web without disclosing privacy sensitive visits accordingly.

As `TrackMeOrNot` allows users to selectively disclose their online footprints, users can enjoy personalized online experience without worrying about privacy. From the perspective of online vendors, `TrackMeOrNot` may persuade users overly concerned with privacy to share footprints selectively for specific rewards, and vendors may use shared browsing habits to yield more profits.

Acknowledgments

The authors would like to thank the anonymous reviewers for their help and feedback. This research was supported by the NSF award CNS-1017265, CNS-0831300, CNS-1149051 and DGE-1500084, by the ONR under grant N000140911042 and N000141512162, by the DHS under contract N66001-12-C-0133, by the United States Air Force under contract FA8650-10-C-7025, by the DARPA Transparent Computing program under contract DARPA-15-15-TC-FP-006. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, ONR, DHS, United States Air Force or DARPA.

References

[1] abine. https://www.abine.com/index.html.

[2] Adblock Plus. http://adblockplus.org/.

[3] Browse in private with incognito mode. https://support.google.com/chrome/answer/95464?hl=en.

[4] Browser Fingerprints: A Big Privacy Threat. http://www.pcworld.com/article/192648/browser_fingerprints.html.

[5] Browser fingerprints, and why they are so hard to erase. http://www.networkworld.com/article/2884026/security0/browser-fingerprints-and-why-they-are-so-hard-to-erase.html.

[6] Cross-browser fingerprinting test 2.0. http://fingerprint.pet-portal.eu/?menu=6.

[7] Digital Advertising Alliance (DAA). http://www.aboutads.info/.

[8] Ghostery. http://www.ghostery.com/.

[9] Lawsuit: Ad Network Could Be Tracking You With HTML5. http://www.techhive.com/article/205950/ad_network_abuses_html5_privacy_advocates_cry_foul.html.

[10] Local Shared Objects – "Flash Cookies". https://epic.org/privacy/cookies/flash.html.

[11] Multi-process architecture in the chromium projects. https://www.chromium.org/developers/design-documents/multi-process-architecture.

[12] Network Advertising Initiative (NAI). https://www.networkadvertising.org/.

[13] Preferences in the chromium projects. https://www.chromium.org/developers/design-documents/preferences.

[14] Taxonomy api | alchemyapi. http://www.alchemyapi.com/products/alchemylanguage/taxonomy.

[15] Taxonomy api documentation | alchemyapi. http://www.alchemyapi.com/api/taxonomy.

[16] Tor browser. https://www.torproject.org/projects/torbrowser.html.en.

[17] The xml c parser and toolkit of gnome. http://www.xmlsoft.org/.

[18] K. Boda, A. M. Földes, G. G. Gulyás, and S. Imre. User tracking on the web via cross-browser fingerprinting. In *Proceedings of the 16th Nordic Conference on Information Security Technology for Applications*, pages 31–46, 2012.

[19] R. Calo. Does nai's opt out tool stop consumer tracking? http://cyberlaw.stanford.edu/blog/2009/04/does-nai%E2%80%99s-opt-out-tool-stop-consumer-tracking.

[20] C. Castelluccia, M.-A. Kaafar, and M.-D. Tran. Betrayed by your ads!: Reconstructing user profiles from targeted ads. In *Proceedings of the 12th International Conference on Privacy Enhancing Technologies*, pages 1–17, 2012.

[21] R. K. Chellappa and R. G. Sin. Personalization versus privacy: An empirical examination of the online consumer's dilemma. *Inf. Technol. and Management*, 6(2-3):181–202, Apr. 2005.

[22] P. Dixon. THE NETWORK ADVERTISING INITIATIVE: Failing at Consumer Protection and at Self-Regulation. http://www.worldprivacyforum.org/wp-content/uploads/2007/11/WPF_NAI_report_Nov2_2007fs.pdf.

[23] P. Eckersley. Stop sneaky online tracking with eff's privacy badger. https://www.eff.org/press/releases/stop-sneaky-online-tracking-effs-privacy-badger.

[24] S. Garfinkel. *Database Nation: The Death of Privacy in the 21st Century*. O'Reilly & Associates, Inc., 2000.

[25] Google. Google ads preferences manager. https://www.google.com/settings/ads/onweb.

[26] J. Jao. Perils of personalization vs. privacy. http://www.mediapost.com/publications/article/210880/perils-of-personalization-vs-privacy.html?edition=.

[27] A. Korolova. Privacy violations using microtargeted ads: A case study. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, pages 474–482, 2010.

[28] J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 413–427, 2012.

[29] N. Nikiforakis, W. Joosen, and B. Livshits. Privaricator: Deceiving fingerprinters with little white lies. In *Proceedings of the 24th International Conference on World Wide Web*, pages 820–830, 2015.

[30] X. Pan, Y. Cao, and Y. Chen. I do not know what you visited last summer: Protecting users from third-party web tracking with trackingfree browser. In *Proceedings of the 2015 Network and Distributed System Security Symposium*, 2015.

[31] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*, 2006.

[32] X. Qi and B. D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2):12:1–12:31, Feb. 2009.

[33] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, pages 155–168, 2012.

[34] F. Sun, D. Song, and L. Liao. Dom based content extraction via text density. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 245–254, 2011.

[35] Uber. User privacy statement. https://www.uber.com/legal/privacy/users/en.

[36] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystifying page load performance with wprof. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, pages 473–486, 2013.

[37] Yahoo. Ad interest manager - yahoo. https://info.yahoo.com/privacy/us/yahoo/opt_out/targeting/.