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

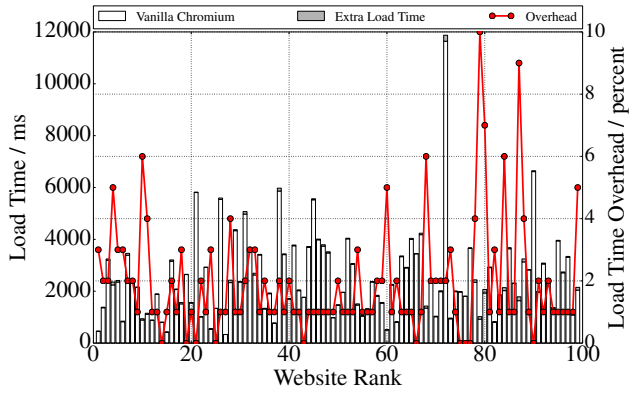

Figure 5: Page load time when visiting each of Alexa top 100 US websites under Content Analysis Only Configuration (CAOC), i.e., when `TrackMeOrNot` does not switch the browsing context. Overall, `TrackMeOrNot` imposed 39.56 ms (1.93%) extra load time on average compared to vanilla Chromium.

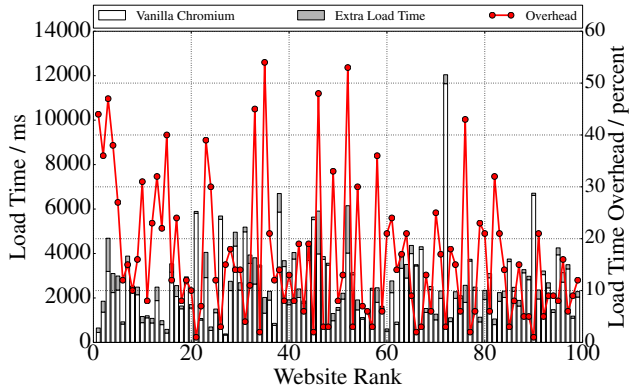

Figure 6: Page load time when visiting each of Alexa top 100 US websites under Browsing Context Switching Configuration (BCSC), i.e., when `TrackMeOrNot` performs switching to the persistent browsing context. Overall, `TrackMeOrNot` imposed 340.33 ms (16.40%) extra load time on average. Although this 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%

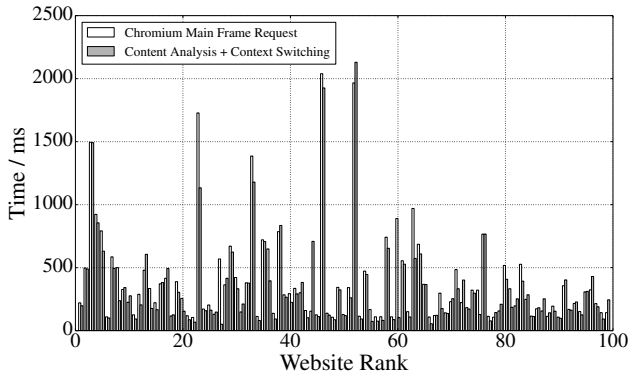

Figure 7: Main frame HTML source load time (marked as white boxes) *v.s.* extra page load time in `TrackMeOrNot` (marked as black boxes) when visiting each of Alexa top 100 US websites.

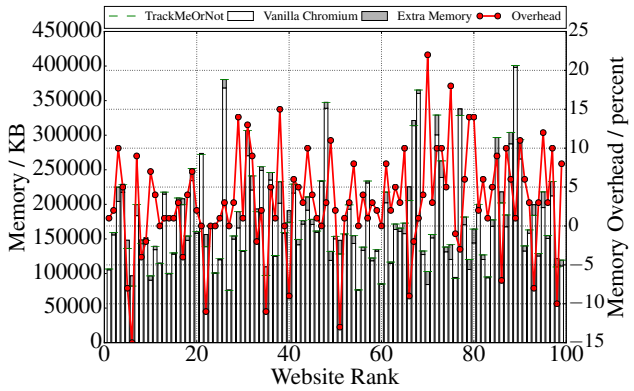

Figure 8: Peak memory usage when visiting each of Alexa top 100 US websites under Content Analysis Only Configuration (CAOC). `TrackMeOrNot` imposed 3.06% overhead on average.

(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

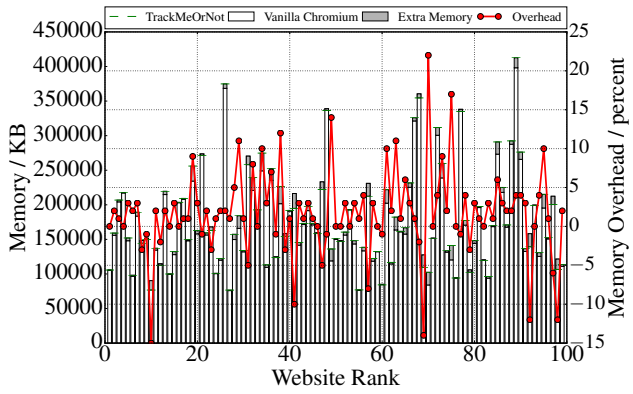

Figure 9: Peak memory usage when visiting each of Alexa top 100 US websites under Browsing Context Switching Configuration (BCSC). `TrackMeOrNot` imposed 1.68% overhead on average.

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	58	34	29	16
	5.52%	40.00%	23.45%	20.00%	11.03%

Gender	Male	Female
	60	85
	41.38%	58.62%

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

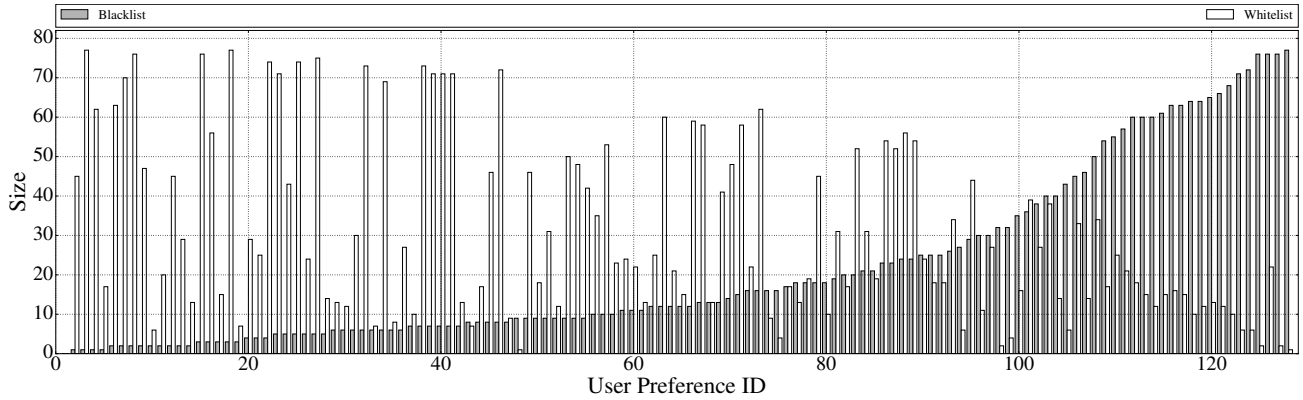

Figure 10: The number of whitelist and blacklist rules across 129 distinct privacy needs.

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

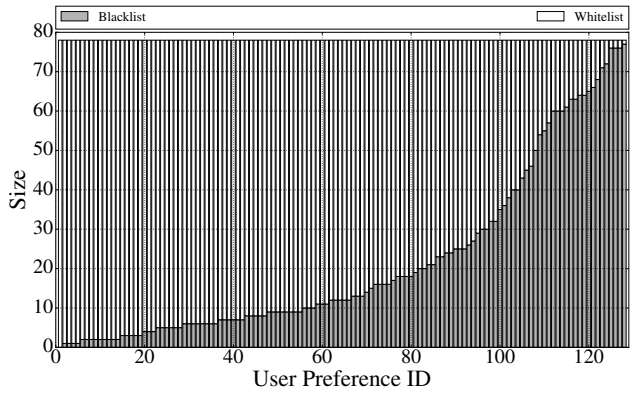

Figure 11: Tracking preferences using persistent fallback browsing context.

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.

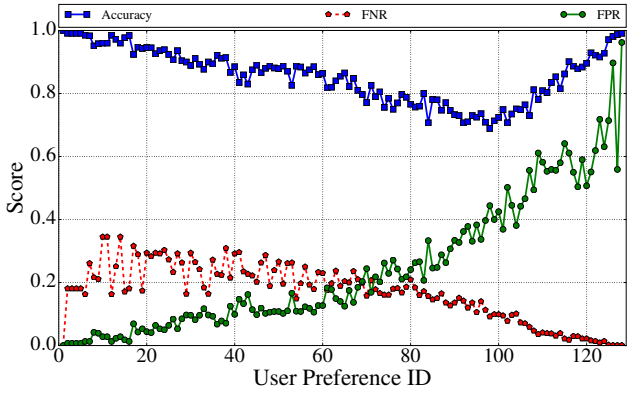

Figure 12: Evaluation results on 129 tracking preferences with persistent fallback browsing context.

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

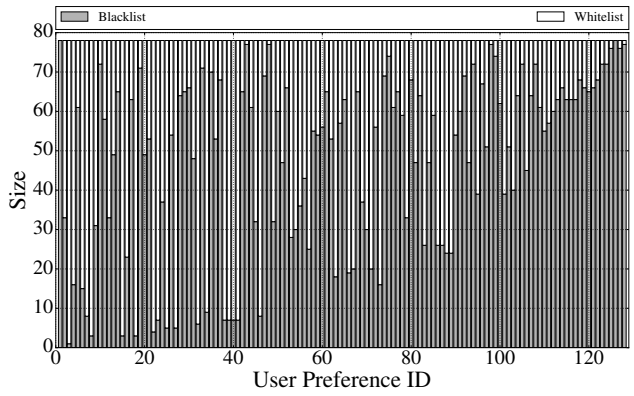

Figure 13: Tracking preferences using anonymous fallback browsing context.

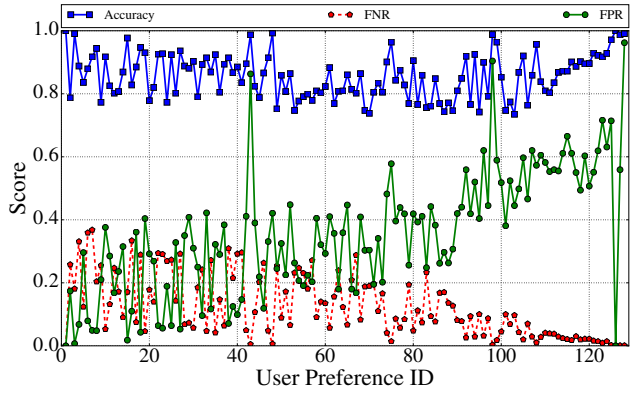

Figure 14: Evaluation results on 129 tracking preferences with anonymous fallback browsing context.

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/.