REPRIV: Re-Envisioning In-Browser Privacy

***Abstract*—In this paper, we present REPRIV, a system that combines the goals of privacy and content personalization in the browser. REPRIV discovers user interests and shares them with third-parties, but only with an explicit permission of the user. We demonstrate how always-on user interest mining can effectively infer user interests in a real browser. We go on to discuss an extension framework that allows third-party code to extract and disseminate more detailed information, as well as language-based techniques for verifying the absence of privacy leaks in this untrusted code. To demonstrate the effectiveness of our model, we present REPRIV extensions that perform personalization for Netflix, Twitter, Bing, and GetGlue.**

This paper evaluates important aspects of REPRIV in realistic scenarios. We show that REPRIV's default in-browser mining can be done with no noticeable overhead to normal browsing, and that the results it produces converge quickly. We demonstrate that REPRIV personalization yields higher quality results than those that may be obtained about the user from public sources. We then go on to show similar results for each of our case studies: that REPRIV enables high-quality personalization, as shown by cases studies in news and search result personalization we evaluated on thousands of instances, and that the performance impact each case has on the browser is minimal. We conclude that personalized content and individual privacy on the web are not mutually exclusive.

I. INTRODUCTION

The motivation of this work comes from the observation that in today's web there are two distinct groups, *users* and *service providers* such as Amazon, Google, Microsoft, Facebook and the like. Service providers are interested in learning as much about their users as they can so that they can better target their ads or provide content personalization. Users might welcome content, ad, and site personalization as long as it does not compromise their privacy.

In today's web, for service providers, personalization opportunities are limited. Even if sites like Amazon and Facebook allow or sometimes require authentication, service providers only know as much about the user as can be gathered through interaction with the site. A user might only spend a few minutes a day on Amazon.com, for example. This is minuscule compared to the amount of time the same user spends in the browser. This suggests a simple lesson: the browser knows much more about you than any particular site you visit. Based on this observation, we suggest the following strategy, which forms the basis for REPRIV:

1) Let the browser infer information about the user's interests based on her browsing behavior, the sites he visits, her prior history, and detailed interactions on web sites of interest to form a *user interest profile*.
2) Let the browser control the release of this information. For instance, upon the request of a site such as Amazon.com or BarnesAndNoble.com, the user will be asked for a permission to send her high-level interests to the site. This is similar to prompting for the permission to obtain the geo-location in today's mobile browsers. By default, more explicit information than this, such as the history of visited URLs would not be exposed to the requesting site. It is an important design principle of REPRIV that the user stay in control of what information is released by the browser.
3) In addition to default user interest mining, REPRIV allows service providers to register extensions that would perform information extraction within the browser. For instance, a Netflix extension (or *miner*) may extract information pertinent to what movies the user is interested in. The miner may use the history of visiting Fandango.com to see what movies you saw in theaters in the past. REPRIV miners are statically verified at the time of submission to disallow undesirable privacy leaks.

This approach is attractive for web service providers because they get access to user's preferences without the need for complex data mining machinery and is in any case based on limited information. It is also attractive for the user because of better ad targeting and content personalization opportunities. Moreover, this approach opens up an interesting new business model: service providers can incentivize users to release their preferences in exchange for store credit, ad-free browsing, or access to premium content. Compared to prior research [9, 28], the appeal of REPRIV is considerably more extensive as it enables the following broad applications:

1) **Personalized search.** Search results from a variety of search engines can be re-ranked to match user's preferences as well as their browsing history (Section VI-A).
2) **Site personalization.** Sites such as Google News, CNN.com, or Overstock.com can be easily adopted within the browser to match user's news or shopping preferences (Section VI-B).
3) **Ad targeting.** Although we do not explicitly focus on ad personalization in this apper, REPRIV enables browser-based ad targeting as suggested by Adnostic [28] and Privad [9].

Note that REPRIV is largely orthogonal to in-private browsing modes supported by modern browsers. While it is still possible for a determined service provider to perform user tracking unless the user combines REPRIV with a browser privacy mode such as InPrivate Browsing in Internet Explorer, it is our hope that, going forward, the service provider will opt for explicitly requesting user preferences through the REPRIV protocol rather than using a back door.

A. Contributions

Our paper makes the following contributions:

- RePriv. We present RePriv, a system for controlling the release of private information within the browser. We demonstrate how built-in data mining of user interests can work within an experimental HTML5 platform called C3 [14].
- **Extension Framework.** We developed a browser extension framework for allowing untrusted third-party code to make use of RePriv's data. We discuss the API and type system based on the Fine programming language [25] that ensures these extensions do not introduce privacy leaks. We developed six realistic miner examples that demonstrate the utility of this framework.
- **Evaluation.** We demonstrate that RePriv mining can be done with minimal overhead to the end-user latency. We also show the efficacy of RePriv mining on real-life browsing sessions and conclude that RePriv is able to learn user preferences quickly and effectively. We demonstrate the utility of RePriv by performing two large-scale case studies, one targeting news personalization, and the other focusing on search result reordering, both evaluated on real user data.

B. Paper Organization

The rest of the paper is organized as follows. Section II provides some background on web privacy and personalization and motivates the problem RePriv attempts to solve. Section III talks about RePriv implementation and resulting technical issues. Section IV discusses custom RePriv miners and their verification. Section V describes our experimental evaluation. Section VI describes two detailed case studies, one focusing on news and the other on search personalization. Section VII discusses the topics of incentives for RePriv use, usability, deployment, etc. Finally, Sections VIII and Section IX describe related work and conclude.

II. Overview

We begin with a high-level discussion in Section II-A of existing efforts to preserve privacy on the web, and how RePriv fits into this context. Section II-B talks about site personalization and Section II-C motivates third-party personalization extensions that we call "miners".

A. Background

One definition of privacy common in popular thought and law is summarized as follows: individual privacy is a person's right to control information about one's self, both in terms of how much information others have access to, and the manner in which others may use it. The web as it currently stands is different from how it was initially conceived; it has transformed from a passive medium to an active one where users take part in shaping the content they receive. One popular form of active content on the web is *personalized* content, wherein a provider uses certain characteristics of a particular user, such as their demographic or previous behaviors, to filter, select, or otherwise modify the content that it ultimately presents. This transition in content raises serious concerns about privacy, as arbitrary personal information may be required to enable personalized content, and a confluence of factors has made it difficult for users to control where this information ends up, and how it is used.

Because personalized content presents profit opportunity, businesses have incentive to adopt it quickly, oftentimes without user consent. This creates situations that many users perceive as a violation of privacy. A prevalent example of this is already seen with online targeted advertising, such as that offered by Google AdSense [7]. By default, this system tracks users who enable browser cookies across all web sites that choose to partner with it. This tracking can be arbitrarily invasive as it pertains to the user's behavior at partner sites, and in most cases the user is not explicitly notified that the content they choose to view also actively tracks their actions, and transmits it to a third party (Google). While most services of this type have an opt-out mechanism that any user can invoke, many users are not even aware that a privacy risk exists, much less that they have the option of mitigating it.

As a response to concerns about individual privacy on the web, developers and researchers continue to release solutions that return various degrees of privacy to the user. One well-known example is the *private browsing modes* available in most modern browsers, which attempt to conceal the user's identity across sessions by blocking access to various types of persistent state in the browser [1]. However, a recent study [1] demonstrated that none of the major browsers implement this mode correctly, leading to alarming inconsistencies between user expectations and the features offered by the browser. Even if private browsing mode were implemented correctly, it inherently poses significant problems for personalized content on the web, as sites are not given access to the information needed to perform personalization.

Others have attempted to build schemes that preserve the privacy of the user while maintaining the ability to personalize content. Most examples [6, 9, 13, 28] concern targeted advertising, given its prevalence and well-known privacy implications. For example, both PrivAd [9] and Adnostic [28] are end-to-end systems that preserve privacy by performing all behavior tracking on the client, downloading *all* potential advertisements from the advertisor's servers, and selecting the appropriate ad to display locally on the client. Although these systems differ in details regarding accounting and architecture, they share a basic strategy for maintaining user privacy: keep sensitive information local to the user, to simplify the matter of control.

The goal of RePriv is to enable general personalized content on the web in a privacy-conscious manner. Like PrivAd and Adnostic, RePriv does this by keeping all of the sensitive information necessary to perform personalization close to the user, within the browser. However, RePriv differs from these systems both technically and in the notion of privacy it considers. Because RePriv does not target a specific application, it does not attempt to completely hide all personal information from the party responsible for providing personalized content. Aside from the improbable technical advances needed to make such a system practical, it is not clear that content providers would take part in such a scheme, as they would loose access to the valuable user data that they currently use to improve their products and increase efficiency. Rather, REPRIV leaves it to the user to decide which parties may access the various types of data stored inside the browser, and manages dissemination accordingly in a secure manner.

We posit that expecting the user to make this decision is not only reasonable, but necessary given the constraints discussed above. The basis of this decision must be two-fold, depending both on the trust the user has in the content provider, as well as the incentive the content provider gives the user for access to his data. However, this type of decision is ultimately similar to the type of decision a user makes when signing up for an account at Amazon.com or Netflix.com: if she agrees to the terms in the privacy policy, then he has deemed the benefit offered by that site worth the reduction in personal privacy needed to obtain it. This is the same negotiation that REPRIV relies on to protect user privacy while still enabling a diverse set of personalized applications. Thus, the challenge of REPRIV is to facilitate the collection of personal information from the browser in a manner flexible enough to enable existing and future personalized applications, while maintaining explicit user control over how that information is used and disseminated to third parties on the web.

B. Motivating Personalization Scenarios

Several applications drove the development of REPRIV. We briefly discuss a sampling of them in this section.

Content Targeting: Commonplace on many online merchant web sites is content targeting: the inference and strategic placement of content likely to compel the user, based on previous behavior. Although popular sites such as Amazon.com and Netflix.com already support this functionality without issue, the amount of personal information collected and maintained by these sites have real implications for personal privacy that may surprise many users [20]. Additionally, the fact that the personal data needed to implement this functionality is vaulted on a particular site is an inconvenience for the user, who would ideally like to use their personal information to receive a better experience on a competitor's site. By keeping all of the information needed for this application in the browser, REPRIV can solve both problems. The content provider can ask the user's browser for data as it needs it, and the user can accept or decline requests either programatically or via a high-level policy.

Targeted Advertising: Advertising serves as one of the primary enablers of free content on the web, and *targeted advertising* allows merchants to maximize the efficiency of their efforts. REPRIV should facilitate this task in the most direct way possible by allowing advertisers to consult the user's personal information in a consentual manner. Advertisers have incentive to use the accurate data stored by REPRIV,

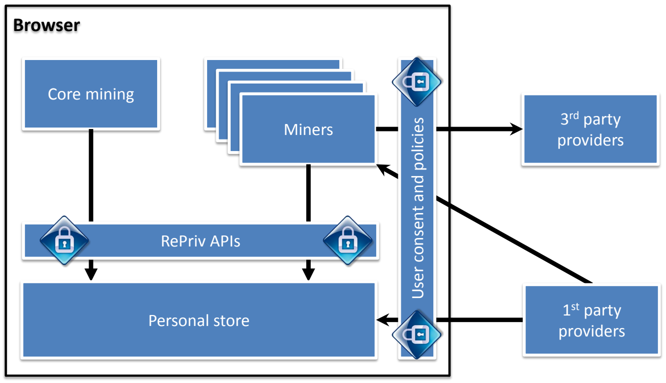


Fig. 1: REPRIV architecture.

rather than collecting their own data, as the browser-computed interests are more representative of the user's complete browsing behavior. Additionally, consumers are likely to select businesses who engage in practices that do not seem invasive or threatening.

C. Personalization Extensions

While the core mining mechanism in REPRIV is meant to be as general-purpose as possible, the pace at which new personalized web applications is appearing suggests that REPRIV will need an extra degree of flexibility to support up-and-coming apps. A large part of our work focuses on an extension platform that enables near-arbitrary programmatic interaction with the user's personal data, in a verifiably privacy-preserving manner.

Topic-Specific Functionality: Users may spend a large amount of time at particular types of sites, e.g. movie-related, science, or finance sites. Users will expect specific personalization on these sites that cannot be provided by a general-purpose behavior mining algorithm. To facilitate this, third-party developers should be able to write extensions that have site-specific understanding of user input, and are able to mediate REPRIV's stored personalization information accordingly. For example, a plugin should be able to track the user's interaction with Netflix.com, observe which movies he likes and dislikes, and update his interest profile to reflect these preferences.

Web Service Relay: Many web API's now provide services relevant to personalization. For example, Netflix now has an API that allows a third-party developer to programmatically access information about the user's account, including their movie preferences and purchase history. Other examples allow a third-party developer to submit portions of a user's overall preference profile or history to receive content recommendations or ratings; Getglue.com, Hunch.com, and Tastekid.com are all examples of this. REPRIV extensions should be able to act as intermediaries between the user's personal data and the services offered by these API's. For example, when a user navigates to Fandango.com, the site can query an extension that in turn consults the user's Netflix interactions and Amazon.com purchases, and returns useful derived information to

Fandango.com for personalized show times or film reviews.

Direct Personalization: In many cases, it is not reasonable to expect a web site to keep up with the user's personalization expectations. It is often simpler to write an extension that can access REPRIV's repository of user information, and modify the presentation of selected sites to reflect preferences. To facilitate this need, REPRIV extensions should be able to interact with and modify the DOM structure of selected web sites to reflect the contents of the user's personal information.

III. TECHNICAL ISSUES

This section is organized as follows. Section III-A discussed browser modifications we implemented to support REPRIV. Section III-B discusses support for REPRIV miners.

A. *Browser Modifications*

Our current research prototype is built on top of C3, an HTML5 experimental platform developed in .NET [14]. However, we believe that other browsers can be modified in a very similar manner. We modified C3 in the following ways to add support for REPRIV:

- Added a behavior mining algorithm that observes users' browsing behavior and automatically updates a profile of user interests (Section III-A).
- Implemented a communication protocol that sits on top of HTTP and allows web sites to utilize the information maintained by REPRIV in the browser (Section III-A).
- Implemented an extension framework that allows third-party extensions to utilize the information maintained by REPRIV, and interact programatically with web sites (Section III-B).

User Behavior Mining: The goal of our general-purpose behavior mining algorithm is to provide relevant parties with two types of information about the user:

- Top-n topics of interest, where n can vary to suit the needs of each particular application,
- The level of interest in a given set of topics, normalized to a reasonable scale.

Our approach works by classifying individual documents viewed in the browser, and keeping related aggregate information of total browsing history in the personal store.

Interest Categories: To characterize user interests, we use a hierarchical taxonomy of document topics maintained by the Open Directory Project (ODP) [22]. The ODP classifies a portion of the web according to a hierarchical taxonomy with several thousand topics, with specificity increasing towards the leaf nodes of the tree. We use only the most general two levels of the taxonomy, which account for 450 topics. To convey the level of specificity contained in our interest hierarchy, a small portion is presented in Figure 2.

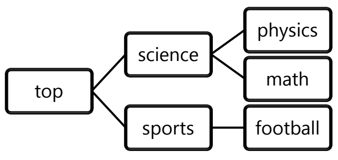


Fig. 2: Portion of taxonomy.

Our taxonomy-based interest classification scheme is similar to those used by targeted advertising networks [7]. As elucidated by Narayanan and Shmatikov [20], care must be taken when selecting the taxonomy to ensure that the target population is not distributed too sparsely among topics in the taxonomy, as anonymity attacks may result. As shown in Figure 2, the depth and specificity of our taxonomy is quite limited.

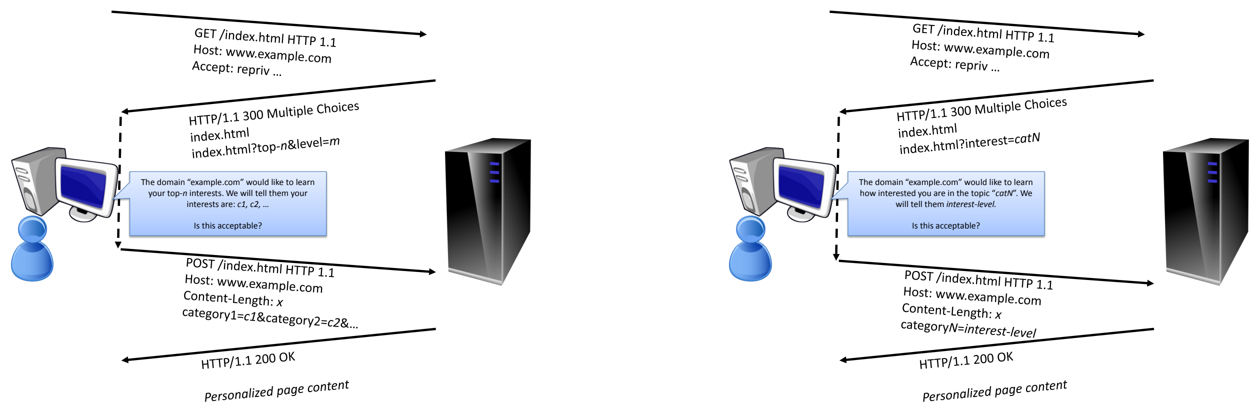
Classifying Documents: Of primary importance for our document classification scheme is performance: REPRIV's default behavior must not impact normal browsing activities in a noticeable way. This immediately rules out certain solutions, such as querying existing web API's that provide classification services. We use the Naïve Bayes classifier for its well-known performance in document classification tasks, as well as its low computation cost on most problem instances. However, REPRIV's high-level functionality is independent of the specific type of classifier used, so this part of the implementation can be varied to suit changing technologies and needs.

To create our Naïve Bayes classifier, we obtained 3,000 documents from each category of the first two levels of the ODP taxonomy. We selected attribute words as those that occur in at least 15% of documents for at least one category, not including stop words such as "a", "and", and "the". We then ran standard Naïve Bayes training on the corpus, calculating the needed probabilities $P(w_i \mid C_j)$, for each attribute word w_i and each class C_j. Calculating document topic probabilities at runtime is then reduced to a simple log-likelihood ratio calculation over these probabilities.

To ensure that the cost of running topic classifiers on a document does not affect browsing activities, this computation is done in a background worker thread. When a document has finished parsing, its `TextContent` attribute is queried and added to a task queue. When the background thread activates, it consults this queue for unfinished classification work, runs each topic classifier, and updates the personal store. Due to the interactive characteristics of internet browsing, i.e. periods of bursty activity followed by downtime for content consumption, there are likely to be many opportunities for the background thread to complete the needed tasks.

Aggregate Statistics: REPRIV uses the classification information from individual documents to relate aggregate information about user interests to relevant parties. The first type of information that REPRIV provides is the "top-n" statistic, which reflects n taxonomy categories that comprise more of the user's browsing history than the other categories. Computing this statistic is done incrementally, as browsing entries are classified and added to the personal store.

The second type of information provided by REPRIV is the degree of user interest in a given set of interest categories. For each interest category, this is interpreted as the portion of the user's browsing history comprised of sites classified with that category. This statistic is efficiently computed by indexing the database underlying the personal store on the column containing the topic category.


```
GET /index.html HTTP 1.1
Host: www.example.com
Accept: repriv ...
```

```
HTTP/1.1 300 Multiple Choices
index.html
index.html?top-n&level=m
```

The domain "example.com" would like to learn your top-*n* interests. We will tell them your interests are: *c1, c2, …*
Is this acceptable?

```
POST /index.html HTTP 1.1
Host: www.example.com
Content-Length: x
category1=c1&category2=c2&...
```

```
HTTP/1.1 200 OK
```

Personalized page content

```
GET /index.html HTTP 1.1
Host: www.example.com
Accept: repriv ...
```

```
HTTP/1.1 300 Multiple Choices
index.html
index.html?interest=catN
```

The domain "example.com" would like to learn how interested you are in the topic "*catN*". We will tell them *interest-level.*
Is this acceptable?

```
POST /index.html HTTP 1.1
Host: www.example.com
Content-Length: x
categoryN=interest-level
```

```
HTTP/1.1 200 OK
```

(a) *top-n* interests

(b) Interest level by category

Fig. 3: Communication protocols for personal information.

Interest Protocol:
REPRIV allows third-party web sites to query the browser for two types of information that are computed by default when REPRIV runs. The protocols are depicted graphically in Figure 3. The design of these protocols is constrained by the following concerns:

1) **Secure dissemination of personal information.** The user should have explicit control over the information that is passed from the browser to the third-party web site. Additionally, it should be possible to communicate this information over a channel secure from eavesdropping.
2) **Backwards compatibility with existing protocols.** Site operators should not need to run a separate daemon on behalf of REPRIV users, or change network infrastructure. Rather, it should be possible to incorporate the information made available by REPRIV with minor changes to existing software.

To address these concerns, we have developed a protocol that utilizes facilities already present in the HTTP specification. This allows implementations to use existing secrecy-preserving HTTP extensions such as HTTPS, without requiring new protocols. We will now walk through each step of the protocol. There are two shown in Figure 3; one for each type of information that can be queried (*top-n* interests and specific interest level by category). However, they differ only in minor ways regarding the types of information communicated.

The client signals its ability to provide personal information by including a `repriv` element in the `Accept` field of the standard HTTP header. If the server daemon is programmed to understand this flag, then it may respond with an HTTP 300 message, providing the client with the option of subsequently requesting the default content, or providing personal information to receive personalized content. The information requested by the server is encoded as URL parameters in one of the content alternatives listed in this message. For example, the server in Figure 3(b) requests the user's interest in the topic "*category-n*", which is encoded by specifying `catN` as the value for the `interest` variable. At this point, the browser prompts the user regarding the server's information request, in order to declassify the otherwise prohibited flow from the personal store to an untrusted party. If the user agrees to the information release, then the client responds with a `POST` message to the originally-requested document, which additionally contains the answer to the server's request. Otherwise, the connection is dropped.

B. Miner Support

To support a degree of flexibility and allow future personalization applications to integrate into its framework, REPRIV provides a mechanism for loading third-party software that utilizes the personal store. We call REPRIV extensions *Miners*, to reflect the fact that they are intended to assist with novel behavior mining tasks. Of primary importance to supporting miners correctly is ensuring that *(1)* they do not leak private user data to third parties without explicit consent from the user, and *(2)* they do not compromise the integrity of the browser, including other miners. The majority of our technical discussion regarding miners addresses these concerns.

Security Policies: To support a diverse set of extensions while maintaining control over the sensitive information contained in the personal store, REPRIV allows extension authors to express the capabilities of their code in a simple policy language. At the time of installation, users are presented with the extension's list of needed capabilities, and have the option of allowing or disallowing the installation. Several of the policy predicates deal with information flow and to *provenance labels*, which are $\langle host, extensionid \rangle$ pairs. All sensitive information used by miners is tagged with a set of these labels, which allow policies to reason about information flows involving arbitrary $\langle host, extensionid \rangle$ pairs. A sampling of the predicates available in REPRIV's policy language is presented in Figure 4.

Given a list of policy predicates regarding a particular miner, the policy for that extension is interpreted as the conjunction of each predicate in the list. This is equivalent to behavioral whitelisting: unless a behavior is implied by the predicate conjunction, the miner does not have permission to exhibit it. Each miner is associated with one static security policy that is active throughout the lifespan of the miner; revocation

$CanCaptureEvents(t, \langle h, e\rangle)$	Extension can capture events of type t on elements tagged $\langle h, e\rangle$.
$CanReadDOMElType(t, h)$	Extension can read DOM elements of type t from pages hosted by h.
$CanReadDOMId(i, h)$	Extension e can read DOM elements with ID i from pages hosted by h.
$CanUpdateStore(d, \langle h, e\rangle)$	Extension can update the personal store with information tagged $\langle h, e\rangle$.
$CanReadStore(\langle h, e\rangle)$	Extension can read items in the personal store tagged $\langle h, e\rangle$.
$CanCommunicateXHR(h_1, \langle h_2, e\rangle)$	Extension can communicate information tagged $\langle h_2, e\rangle$ to host h_1 via XHR-style requests.
$CanServeInformation(h_1, \langle h_2, e\rangle)$	Extension can serve programmatic requests to sites hosted by h_1, containing information tagged $\langle h_2, e\rangle$. An example of a programmatic request is an invocation of an extension function from the JavaScript on a site in d.
$CanHandleSites(h)$	Extension can set load handlers on sites hosted by h.

Fig. 4: Selected security policy predicates. A full listing is available in our technical report [15].

is not needed by any of our current applications, and is not supported by the extension framework.

Tracking Sensitive Information: When a miner makes a call to REPRIV requesting information from the personal store, special precautions must be taken to ensure that the returned information is not misused. Likewise, when a miner writes information to the store that is derived from content on pages viewed by the user, REPRIV must ensure that the user's wishes about the privacy of web content are not violated. All REPRIV functionality that returns sensitive information to miners first encapsulates it in a private data type `tracked`, which contains metadata indicating the provenance of that information.

This allows REPRIV to take the provenance of data into account when it is used by miners. The `tracked` type is opaque – it does not allow miner code to directly reference the data that it encapsulates without invoking a REPRIV mechanism that prevents misuse. This means that REPRIV can ensure complete noninterference, to the degree mandated by the miner's policy. Whenever the miner would like to perform a computation over the encapsulated information, it must call a special `bind` function that takes a function-valued argument and returns a newly-encapsulated result of applying it to the `tracked` value. This scheme prevents leakage of sensitive information, as long as the function passed to `bind` does not cause any side effects. We discuss verification of this property below.

Verifying Miners: REPRIV verifies miners against their stated properties statically using security types. This eliminates the need for costly run-time checks, and ensures that a security exception will never interrupt a browsing session. To meet this goal, we require that all untrusted miners be written in Fine [25], a security-typed programming language. Fine allows programmers to express dependent types on function parameters and return values, which forms the basis of REPRIV's verification mechanism. Fine provides a language-level sandbox, so all useful functionality is available to miners only through a set of API functions. The interface for these

```
val MakeRequest:
  p:provs ->
  {host:string | AllCanCommunicateXHR h p} ->
  t:tracked<string,p> ->
  {eprin:string | ExtensionId eprin} ->
  fp:{p:provs | forall (pr:prov).(InProvs pr p)
              <=> (InProvs pr p || pr = (P h eprin))} ->
  mut_capability ->
  tracked<xdoc,fp>

val AddEntry:
  ({p:provs | AllCanUpdateStore p}) ->
  tracked<string,p> ->
  string ->
  tracked<list<string>,p> ->
  mut_capability ->
  unit
```

Fig. 5: Example API definitions.

API's specifies type refinements on key parameters that reflect the consequence of each API function on the relevant policy predicates. Verification occurs at each code point where an API function is invoked: the miner's policy is checked against the dependent type signature of the API function.

Two example interface definitions are given in Figure 5. The first example, `MakeRequest`, is the API used by miners to make HTTP requests; several policy interests are operative in its definition. The second argument of `MakeRequest` is a string that denotes the remote host with which to communicate, and is refined with the formula

```
AllCanCommunicateXHR host p
```

where `p` is the provenance label of the buffer to be transmitted. This refinement ensures that a miner cannot call `MakeRequest` unless its policy includes a `CanCommunicateXHR` predicate for each element in the provenance label `p`. Because the REPRIV API is very limited, we are assured that this is the only function that impacts the `CanCommunicateXHR` predicate.

Notice as well that the third argument, as well as the return value of `MakeRequest`, are of the dependent type `tracked`. `tracked` types are indexed both by the type of the data that they encapsulate, as well as the provenance of that data. The third argument is the request string that will be sent to the host specified in the second argument; its provenance plays a part in the refinement on the host string discussed above. The return value has a provenance label that is refined in the fifth argument. The refinement specifies that the provenance of the return value of `MakeRequest` has all elements of the provenance associated with the request string, as well as a new provenance tag corresponding to $\langle \texttt{host}, \texttt{eprin} \rangle$, where `eprin` is the extension principal that invokes the API. This reflects all of the principals that could affect the value returned by `MakeRequest`. The refinement on the fourth argument ensures that the extension passes its actual `ExtensionId` to `MakeRequest`. These considerations ensure that the provenance of information passed to and from `MakeRequest` is available for all necessary policy considertations.

As discussed above, verifying correct enforcement of information flow properties in REPRIV requires checking that functional arguments passed to `bind` are side effect-free. Fine's language-level sandbox guarantees that side effects are only created via API calls; our verification task reduces to ensuring that API's which create side effects are not called from code that is invoked by `bind`, as `bind` provides direct access to data encapsulated by `tracked` types. We use capability tokens that are given *affine types* [25] to gain this assurance. Roughly, an affine typed-variable can only be used once, so an affine token that is copied in the program text results in a type error. Each API function that may create a side effect takes an affine token `mut_capability` as an argument (short for "mutation capability"), which indicates that the caller of the function has the right to create side effects. REPRIV passes the `main` function of each miner a value of type `mut_capability`, which the miner must in turn pass to each location that calls a side-effecting function. Because `mut_capability` is an affine type, and the functional argument of `bind` does not specify an affine type, the Fine type system will not allow any code passed to `bind` to reference a `mut_capability` value Because the constructor for `mut_capability` is private and the original token cannot be copied, the functional passed to bind has no way of generating a value of type `mut_capability` required to invoke a side-effecting function. As an example of this construct in the REPRIV API, observe that both API examples in Figure 5 create side effects, so their interface definitions specify arguments of type `mut_capability`.

Verification Philosophy: The policy associated with a miner is expressed at the top of its source file, using a series of Fine `assume` statements: one `assume` for each conjunct in the overall policy. An example of this is shown in Figure 8, where the policy assumptions of the miner are 3–5 lines of the source code. Given the type refinements on all REPRIV API's, verifying that the miner correctly implements its stated policy is reduced to an instance of Fine type checking. The soundness of this technique rests on three assumptions:

- The soundness of the Fine type system, and the correctness of its implementation. The soundness of the type system was established via a mechanical proof [25].
- The correctness of the dependent type refinements placed on the API functions. This amounts to less than 100 lines of code, which reasons about a relatively simple logic of policy predicates. Furthermore, because the REPRIV API is very limited, it is easy to argue that refinements are placed on all necessary arguments to ensure sound enforcement. In other words, the API usually only provides one function for producing a particular type of side effect, so it is not difficult to check that the appropriate refinements are placed at all necessary points.
- The correctness of the underlying browser's implementation of functions provided by the REPRIV API. For REPRIV, we used C3, an experimental managed-code HTML5 platform. C3 is written in a memory-managed language (C#), providing assurance that it does not contain memory corruption vulnerabilities. The logical correctness of C3 code needed by REPRIV has not been formally verified, but doing so is a goal of future work.

We stress that these are modest requirements for the trusted computing base, and point towards the overall soundness of REPRIV's security properties.

IV. REPRIV MINERS

In this section, we discuss several miner templates and their corresponding policies, as well as two concrete examples: TwitterMiner and GlueMiner. Two additional miners, BingMiner and NetflixMiner, are discussed in various capacities, but their complete description is available only in the technical report.

A. Miner Patterns

In general, miners can provide a wide range of functionality when it comes to updating the personal store with information that reflects the user's browser-related behaviors. In this section, we present three patterns of functionality that we envision many potential miners following. The policies for each category can be *templatized*, easing the burden on miner developers who wish to create variations on these basic patterns. The three patterns are summarized in Figure 6.

The first miner pattern, "site-specific parsing", includes extensions that are aware of the layout and semantics of specific web sites, and are able to update the user's interest profile accordingly. For example, TwitterMiner invokes REPRIV's document classifier over the text contained in the user's latest tweets, and BingMiner classifies the user's search terms. Miners that follow this pattern either need to send HTTP requests to relevant web API's, as in the case of TwitterMiner, or read the relevant DOM elements from particular sites, as with BingMiner. They invariably require permission to update the personal store with information derived from these sources.

The second pattern, "category-specific information", returns detailed information about the user's interactions with specific types of sites to services that request it via a JavaScript interface. NetflixMiner is an example of this pattern; the user's interactions with pages hosted by `netflix.com` are monitored, and information is added to the personal store to reflect this.

Pattern	Policy Template
Site-specific parsing	For the domain d of interest, either *CanCommunicateXHR*(d) or *CanReadDOM**(d, _) *CanUpdateStore*(_, d) *CanHandleSites*(d) (optional, depending on the semantics of the miner) *CanCaptureEvents*(_, d) (optional, depending on the semantics of the miner)
Category-specific information	For the domain d of interest, either *CanCommunicateXHR*(d) or *CanReadDOM**(d, _) *CanUpdateStore*(*Tag*(_, d)) *CanHandleSites*(d) (optional, depending on the semantics of the miner) *CanCaptureEvents*(_, d) (optional, depending on the semantics of the miner) *CanReadStore*(*Tag*(_, d)) For each domain p that can request category-specific information, *CanServeInformation*(p, *Tag*(_, d))
Web service relay	For the API provider a and each provenance tag t sent to a *CanCommunicateXHR*(a, t) and *CanReadStore(t)* For each domain p that can make requests, *CanServeInformation(p, t)* and *CanServeInformation(p, a)*

Fig. 6: Miner patterns and their policy templates.

	Lines of code		Verification
Name	**C#**	**Fine**	**Time (s)**
TwitterMiner	89	36	6.4
BingMiner	78	35	6.8
NetflixMiner	112	110	7.7
GlueMiner	213	101	9.5

Fig. 7: Miner characteristics.

When a third-party site, such as `fandango.com`, would like to personalize based on the user's recent movie interests, NetflixMiner queries the store to retrieve the list of most recently-viewed entries by genre, and returns the relevant titles to the third-party site. In addition to the capabilities required by site-specific parsing miners, miners that follow this pattern also need the ability to read from the store, and return tagged information to specific sites via a programmatic interface.

The final pattern, "web service relay", acts as a privacy-conscious intermediary between the user's personal information, and web sites that provide useful services using this information. Miners in this category expose functionality via a JavaScript interface, and query a third-party web service with data from the personal store to implement this functionality. For example, GlueMiner returns movies similar to those recently viewed by the user by reading store entries created by NetflixMiner, sending them to the API provided by Getglue.com, and returning the results to the JavaScript that requested this information.

B. Miner Examples

In this section we discuss examples of miners that we wrote for REPRIV.

TwitterMiner: TwitterMiner utilizes the RESTful API exposed by `twitter.com` to periodically check the user's twitter profile for updates. When the user posts a new tweet, TwitterMiner analyzes its content using REPRIV's classifier to determine how to update the personal store accordingly.

TwitterMiner needs only two capabilities from REPRIV, as the `twitter.com` API simplifies its task:

1) It must be able to make XHR-style requests to `twitter.com`. The second argument of the *CanCommunicateXHR* capability must indicate that TwitterMiner cannot send any sensitive information derived from the store in such a request.
2) It must be able to update the store to reflect data derived from `twitter.com`

```
module TwitterMiner

open Url
open RePrivPolicy
open RePrivAPI

// Policy assumptions
assume extid: ExtensionId "twitterminer"
assume PAx1: CanCommunicateXHR "twitter.com"
assume PAx2: forall (s:string) . (ExtensionId s) =>
  CanUpdateStore (P "twitter.com" s)

// Miner code
val GetDescription: xdoc -> string
let GetDescription d =
 let allMsgs =
 ReadXDocEls d "item" (fun x -> true) "description" in
 match allMsgs with
  | Cons h t -> h
  | Nil -> ""

val CollectLatestFeed: ({s:string | ExtensionId s}) ->
                       mut_capability ->
                       unit ->
                       unit
let CollectLatestFeed extid mcap u =
 let twitterProv = simple_prov "twitter.com" extid in
 let reqUrl =
       mkUrl "http" "twitter.com" "statuses..." in
 let twitFeed =
       MakeXDocRequest reqUrl extid twitterProv mcap in
 let currentMsg =
       bind twitterProv twitFeed GetDescription in
 let categories =
       bind twitterProv currentMsg ClassifyText in
 AddEntry twitterProv currentMsg "tweet" categories mcap

val main: mut_capability -> unit
let main mcap =
 let collect =
  (CollectLatestFeed "twitterminer" mcap) in
 SetTimeout 600000 collect
```

Fig. 8: Twitter miner in Fine, abbreviated for presentation.

The source code for TwitterMiner is shown in Figure 8 There are only two places in the Fine code in which the programmer must justify to the compiler that the stated policy is in fact being enforced. The first is in the type signature of `CollectLatestFeed`, where a refined type is used to tell the compiler that the identifier `extid` refers to the extension ID stated in the policy manifest. The second location is the first statement in `CollectLatestFeed`, where a provenance label is constructed to reflect the source of information that will be collected by TwitterMiner, e.g. `twitter.com`. This allows the compiler to verify that the tracked information being sent to the store at the end of `CollectLatestFeed` is in accordance with the policy. Refinements on the type of API function `MakeXDocRequest` make it impossible for the programmer to forge this provenance label; if the constructed label does not accurately reflect the URL passed to `MakeXDocRequest`, a type error will indicate a policy violation.

GlueMiner: GlueMiner is different from TwitterMiner in that it does not add anything to the store; rather, it provides a privacy-preserving conduit between third-party web sites that want to provide personalized content, the user's personal store information, and another third party (`getglue.com`) that uses personal information to provide personalized content recommendations. The function `predictResultsByTopic` is the core of its functionality, effectively multiplexing the user's personal store to `getglue.com`: a third-party site can use this function to query `getglue.com` using data in the personal store. This communication is made explicit to the user in the policy expressed by the extension. Given the broad range of topics on which `getglue.com` is knowledgeable, it makes sense to open this functionality to pages from many domains. This creates novel policy issues: the user may not want information in the personal store collected from `netflix.com` to be queried on behalf of `linkedin.com`, but may still agree to allowing `linkedin.com` to use information from `twitter.com` or `facebook.com`. Likewise, the user may want sites such as `amazon.com` and `fandango.com` to use the extension to ask `getglue.com` for recommendations based on the data collected from `netflix.com`.

This usage scenario suggests a fairly complex policy for the proposed extension.

- The extension must only communicate personal store information from Twitter.com and Facebook.com to Linkedin.com through the return value of `predictResultsByTopic`. Additionally, the information that is ultimately returned will be tagged with labels from `getglue.com`, as it was communicated to this host to obtain recommendations. Thus, GLUEMINER must be able to communicate these sources to Getglue.com, and it must be able to send information tagged from Getglue.com to Linkedin.com through the return value of `predictResultsByTopic`.
- Similarly, the extension must only leak information from Netflix.com to Getglue.com on behalf of Amazon.com or Fandango.com. This creates policy requirements analogous to those of the previous case.

The policy requirements of `GlueMiner` are made possible by REPRIV's support for multi-label provenance tracking. Note also the assumption that Getglue.com is not a malicious party, and does not otherwise pose a threat to the privacy concerns of the user. This judgement is ultimately left to the user, as

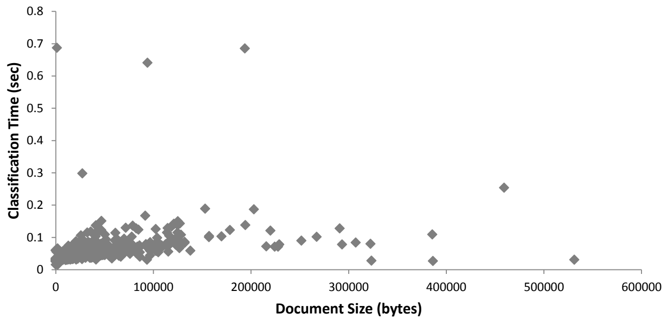


Fig. 9: Document classification time.

REPRIV makes explicit the requirement to communicate with this party, and guarantees that the leak cannot occur to any other party.

V. EXPERIMENTAL EVALUATION

The experimental section is organized as follows. First, we characterize the performance overhead of REPRIV on browsing activities, with respect to both the default behavior mining that occurs in the background and the topic-specific extensions discussed in Section IV. Then, we talk about the quality of our document classifier, that is used for all default in-browser behavior mining. Finally, we discuss the usability concerns that arise with REPRIV.

A. *Performance Overhead*

We evaluated the effect of REPRIV on the performance of web browsing activities. Several aspects of REPRIV can affect the performance of browsing. This section is organized to provide a separate discussion of each such aspect: the effect of default in-browser behavior mining, the effect that each proposed personalization extension (Section IV) has on document loading latency, and the performance of primary extension functionality.

In-Browser Behavior Mining: One of the major components of REPRIV is the behavior mining that happens by default inside the browser, as the user navigates sites. In this section, we characterize the cost of performing this type of mining and the impact that it has on browser performance. Figure 9 depicts the amount of time in seconds needed by REPRIV to classify a document, plotted against the size of the document. Nearly all documents are classified in around one-tenth of a second; given this result, it is clear that REPRIV will not adversely affect the performance of the browser.

Personalization Extensions: One concern with REPRIV's support for miners is the possibly arbitrary amount of memory overhead that it can introduce. We sought to characterize the memory requirements of REPRIV miners, by loading many compiled copies of the four miners presented in the previous section into a running instance of C3. We found that even in an extreme case, with one-hundred miners loaded into memory, only 20.3 megabytes of memory are needed.

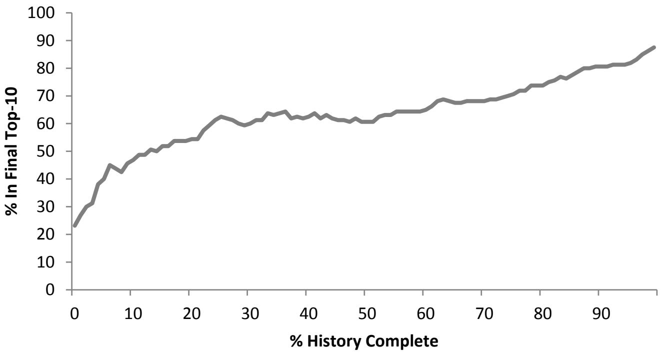


Fig. 10: Convergence curves.

B. *Classifier Effectiveness*

We sought to characterize the quality of the default in-browser classifier. However, doing so is not straightforward, as the task of document classification is inherently subjective. Our evaluation focuses on two metrics: the rate at which a user's interest profile converges, and human-perceived accuracy of the classifiers.

Profile convergence: The rate at which a user's interest profile converges is an important property of our implementation, as it indicates the reliability of the personalization information provided by REPRIV. To measure the convergence of a profile, we require a notion of its final form. All of our measurements are taken over history traces of IE8 users, so the final profile that we use in these measurements is simply the profile computed by our classifier after processing an entire trace. All convergence measurements for a given trace are taken relative to the final profile for that trace, computed in this manner.

We use two measures of convergence. The first is the percentage of current entries in the top-ten list of interest categories that are also present in the final top-ten list. This measure is relevant because we foresee many web sites querying REPRIV for top interests using the protocol outlined in Section III. The second measure is the average distance of each interest category in the current ordering from its position in the final ordering; this gives a global view of interest profile stability.

The results of these experiments are presented in Figures 10 (a) and (b), which depict top-ten and distance convergence, respectively. They key point to notice about both of these curves is the state of the computed interest profile after 20% completion: 50% of the final top-ten categories are already present, and the global convergence curve has reached a point of gradual decline. This implies that the results returned by the core mining algorithm will not change dramatically from this point.

In-browser vs. public data mining: We claim that a major incentive for web service providers to utilize the personalization features enabled by REPRIV is the high quality of personal information that is available within the browser, relative to other types of information used for this purpose. In this subsection, we compare REPRIV's mining algorithm when used over browsing history data to the results obtained by gathering publicly-available information given a person's name. This approach is being used to facilitate personalization by a number of web sites [26].

We see a fundamental problem with this approach, in that most names have several homonyms, and the precision and accuracy of a behavior profile will be adversely affected by this condition. To demonstrate this fact, we began by measuring the number of distinct homonyms for 48 names selected at random from a phone book. To take this measurement, we used a search engine called "WebMii" [29] which returns a listing of much of the publicly-available information about a particular name on the web, in addition to a list of homonyms for that name. The results are displayed in Figure 11 (a): each bucket on the x-axis contains all of the values between the listed number, and that immediately left of it. Noteworthy is the fact that fewer than ten of the names were found to be unique on WebMii; the remaining names either had no visible web presence, or from dozens to hundreds of homonyms. Clearly, these names would be very difficult to build an accurate profile for content personalization without additional input.

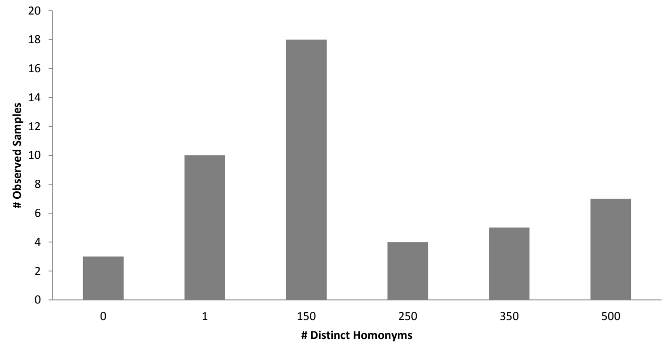
Figure 11(b) relates the confidence in result accuracy that REPRIV's core mining algorithm produces for documents collected by searching the web for documents with a given name, versus running the algorithm over a user's search history. The confidence is the sum of the probabilities computed for each interest category in the user's final top-10 interest profile, normalized by the number of documents used to build the profile to fit a scale of 0 to 1. The public profiles and user histories do not correspond to the same person when grouped at the same point on the x-axis; rather, they are sorted by confidence. To build a public profile for a given name, we searched for that name on Yahoo.com, Facebook.com, Twitter.com, Hi5.com, and Myspace.com. The browsing histories are a subset of those used to compute the data in Figure 10. The results in figure 11(b) show that in all but a very few cases, the behavior mining algorithm was able to come to a much stronger conclusion given browsing histories.

VI. CASE STUDIES

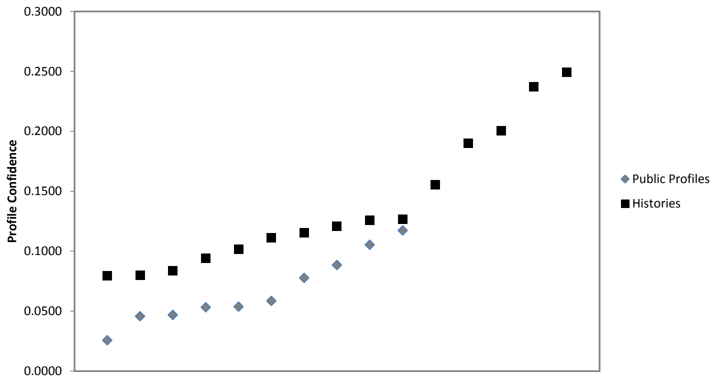
While the previous section provided a basic experimental evaluation of both the core mininig strategy and miners used in REPRIV, this section goes more in depth using two case studies, both evaluated on large quantities of real data. Section VI-A talks about our search personalization experiment. Section VI-B discusses news personalization.

A. *Search Personalization*

We wrote an extension that uses REPRIV's APIs to personalize the results produced by the main Bing search engine. The extension operates by observing the user's previous behavior on Bing, and memoizing certain aspects relevant to future searches. Specifically, for a given search term, the extension records which sites the user selected from the results pages, as well as the frequency with which each host is selected in search results (across all searches). When a new search query



(a) Homonyms for 48 randomly-selected names



(b) Profile confidence

Fig. 11: In-browser vs. public information-based personalization.

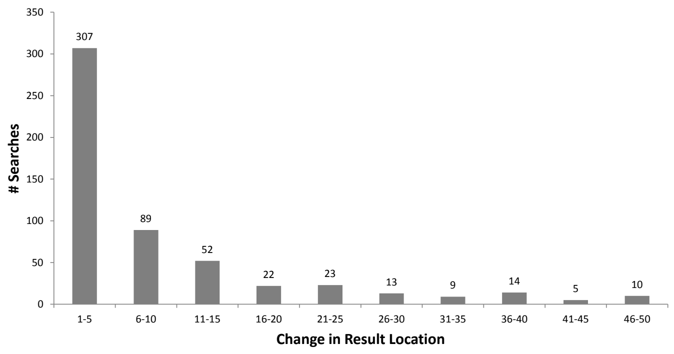


Fig. 12: Search personalization effectiveness.

is submitted, the extension checks its history of previously-recorded searches for an identical match, and places the previously-selected results at the top of the current ranking. The remaining results are ranked by the frequency with which the user visits the host of each result.

This type of search personalization is appealing for two reasons. First, the quality of results it provides is quite good, as discussed below. Second, it is not particularly invasive, as it requires observing user interaction on a single domain (Bing.com). Furthermore, this information is leaked back to no site other than Bing.com through re-arranging the result pages of queries submitted to the search engine; if the user has cookies enabled, then Bing.com learns this information by default. It is also important to note that information is only leaked to Bing.com if the results pages contain JavaScript code that reflects on the layout of the DOM, and takes note of the relative position of search results. This activity would not be possible to hide from the Internet community, effectively minimizing its risk to end-user privacy and giving Bing.com disincentive to do it.

To provide this functionality, the extension needs the following capabilities:

- To determine which search results the user selects from Bing.com sessions, the extension must be able to receive `onclick` events from pages hosted by Bing.com.
- To access a full list of search results over which it can perform re-ranking, the extension uses a public web API. For this, it must be able to make HTTP requests to either Bing.com, Yahoo.com, or Google.com (search API providers).
- To re-arrange the results pages from Bing.com, the extension must be able to change the `TextContent` of HTML elements on Bing.com, as well as well as call change the `href` attribute of `a` elements.
- To memoize search engine interactions, the extension must be able to write data from Bing.com to the personal store.

Implementation details: We implemented the extension for C3 as 382 lines of Fine. The code is presented in our corresponding technical report. The extension uses several of the API's exposed by REPRIV: XMLHttpRequest, SetAttribute, SetTextContent, GetElementById, and GetChildren. When loaded into the browser, the extension requires approximately 200 KB of memory.

Experimental methodology: To evaluate the effectiveness of search personalization, we utilized the histories of nineteen users of the Bing search toolbar. Each history represents seven months of Bing search activity. Our methodology for evaluating the effectiveness of search personalization algorithm is based on the results selected by users for a given query. For each search performed by a particular user, we split the search history into two chronologically-contiguous halves. We construct the relevant portions of a personal store needed to perform search personalization using the first half, and use the second half to evaluate the effectiveness of the algorithm. For each query in the second half of each trace, we evaluated the effectiveness of our search personalization algorithm as follows:

1) Submit the query to the Yahoo BOSS API [31], and collect the default search result ranking.
2) Re-rank the results according to the algorithm discussed above.
3) Note the difference in position for the search result se-

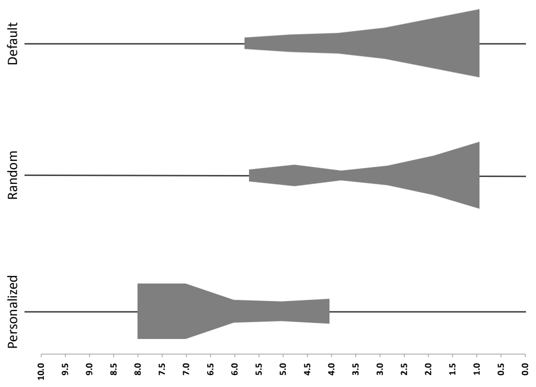


Fig. 13: News personalization effectiveness.

lected by the user between the default and personalized rankings. A positive difference indicates that the selected result is ranked higher in the personalized results, whereas a negative difference indicates the opposite.

This process simulates the user's interaction with a personalized and non-personalized search engine, giving us a baseline for comparison.

Evaluation: The results of our evaluation are summarized in Figure 12. This histogram shows the number of positions the user's selected result moved towards the top of the ranking when the search personalization extension was able to improve results.

We found that for a given user, *the extension was able to improve results 49.1% of the time by raising the user's selected result 8.2 positions toward the top, on average*. 7.7% of the time, the extension lowered the ranking of the user's selected result, but when this occurred, the result was moved downwards an average of only 2.4 positions. For the remaining percentage of time, 43.2%, the extension had no effect on the ranking of the user's selected result. These results show that our search personalization algorithm is able to provide useful functionality for a large portion of the user's web searching activities, while giving the user explicit control over the way in which personal information is used in the process.

B. News Personalization

We wrote an extension that uses REPRIV's computed behavior profile to personalize the New York Times front page. The extension utilizes the collaborative filtering provided by the Digg.com community by matching the user's top interest categories with topic names understood by Digg.com, and periodically querying its web API for "hot" stories in those topics. When the user visits NYTimes.com, New York Times articles cached from Digg.com API queries are presented at the top of the page, in place of the default headlines.

To perform this personalization, the extension needs several capabilities.

- To query the Digg.com API, it must be able to send HTTP requests to Digg.com and access the formatted responses containing news stories.
- To locate the appropriate HTML elements on the NYTimes.com front page for personalized re-formatting, the extension must be able to call `GetElementById` and `GetAttribute("class")` on DOM nodes hosted by NYTimes.com.
- To re-format the NYTimes.com front page, the extension must be able to change the `TextContent` of nodes on NYTimes.com nodes, as well as call `SetAttribute("href")` on them.
- To construct te appropriate query to Digg.com, it must be able to query the personal store to learn the top interests of the user.

Implementation details: We implemented the extension for C3 as 124 lines of Fine. The code is presented in our corresponding technical report.

The extension uses several of the API's exposed by REPRIV: XMLHttpRequest, GetAttribute, SetAttribute, SetTextContent, GetTopInterests, GetElementById, SetTimeout, and GetChildren. When loaded into the browser, the extension requires approximately 200KB of memory.

When navigating to NYTimes.com, we found that the extension introduced a latency of 6% over the default loading time without any personalization, which is a consequence of the fact that the extension modifies the DOM after initial loading is complete. This overhead does not reflect the time needed to query the Digg.com API, which occurs periodically in a background thread that runs when the CPU is otherwise idle.

Experimental methodology: We performed a set of experiments using Amazon's Mechanical Turk service [19] to demonstrate that our news personalization system does not trivialize the problem of delivering personalized content in fulfilling the goal of preserving user privacy. In other words, we sought to show that the type of personalization offered by our extension is relevant to internet users.

To do so, we generated 1920 artificial behavior profiles. 900 of the profiles contained three randomly-selected user interest topics, and the rest contained three topics related by the same top-level ODP category. This distribution models users with both focused and diverse interests. We then seeded our personalization algorithm with each profile, and captured an image of the stories that would be presented by the extension. The image contained the headline of each story, as well as a short summary of each story, in a manner similar to the default NYTimes.com layout.

Using the images and interest profiles, we generated a set of Mechanical Turk surveys. Each survey consisted of twelve questions, where each question paired a news content image with a potential behavior profile, and asked the user how relevant the stories presented in the image were to the given set of interest topics, on a scale of 1 to 10. For each survey, approximately half of the questions matched the image with the interest profile our algorithm used to generate them, and the other half were paired randomly. Each survey contained an additional question that paired the default NYTimes.com front page stories with a random interest profile. The latter two pairings served as our control, to determine how relevant users found hypothetical interest profiles to general news stories.

Evaluation: "Personalized" denotes real pairings of personalized news stories to behavior profiles, "Random" refers to pairings of news stories to randomly-generated behavior profiles that do not bear a meaningful connection, and "Default" denotes the stories presented on the default NYTimes.com front page paired with a random behavior profile. For each column, the statistical mean among survey responses, as well as the surrounding vicinity of one standard-deviation, is plotted.

As the figure indicates, respondents gave stories personalized with our algorithm significantly higher relevance scores than the control samples. For personalized content, ratings between 6.5 and 8 recieved the most responses, with markedly lower variance than the control. While some overlap in response exists between personalized content and the control, the majority of control responses mass around low relevance scores, indicatating a clear improvement in percieved relevance for content personalized using our algorithm.

In summary, the results of this news personalization experiment show that REPRIV enables useful and effective personalization of news content without sacrificing control over private information.

VII. DISCUSSION

In this section, we discuss issues surrounding the adoption and feasibility of REPRIV.

A. Incentives

The incentives for users to adopt REPRIV are immediate: REPRIV was designed to facilitate the types of personalized web experience that have become popular today, while allowing users to maintain control of their personal information. REPRIV also helps to solve the *cold-start problem*, where a user visits a new web site and is not able to recieve personalized content for lack of data. Finally, we have demonstrated that REPRIV's performance overhead is minimal, so there is very little disincentive for a user to adopt REPRIV.

While a truly anonymous browsing mode would leave content providers without an alternative, incentives already exist for service providers to adopt REPRIV without the need for such measures. The first such incentive is the quality of information that REPRIV can provide relative to other techniques. REPRIV gives service providers the opportunity to utilize data that is not impeded by tracker blockers on the client, that is derived using information from the user's complete browsing experience. Secondly, because REPRIV gives content providers a way to respect user privacy without sacrificing functionality, they can differentiate themselves from competitors by appealing to the users' desire for privacy.

Finally, we forsee a number of likely scenarios to incentivize miner authorship. First observe that incentive must already exist, as developers already produce browser extensions that track user behavior; this is typically done without the user's consent, and is sometimes referred to as *spyware* [16] (one famous example is the Alexa toolbar, published by Amazon.com). REPRIV gives these developers a way of writing similar functionality, but in a manner that is verifiably benign. Another likely scenario arises with content recommendation services, such as Getglue.com and Hunch.com. These sites allow users to create profiles of their interests for sharing with other users and receiving content recommendations. Key to the effectiveness of these services is the amount of personal information that can be used for recommendation. REPRIV miners are a safe way for these sites to gather this information.

B. Usability Concerns & Distribution Model

At some point, the user must manually consent to the information being disseminated by REPRIV. The structure of core mining data was designed to be highly informative to content providers and intuitive for end-users: when prompted with a list of topics that will be communicated to a remote party, most users will understand the nature and degree of information sharing that will take place if they consent.

However, the usability problems posed by miners is more difficult. While the privacy policies imposed on miners are expressive and precise, it is difficult to make their implications explicit to an average user. To remedy this, we suggest a distribution model that provides high-level policy review of miners prior to their release, and allows for revocation. This model is similar to that adopted by Firefox, Apple, and Symbian for supporting third-party functionality. The owner of such a repository is expected to posess considerably more technical sophistication than most browser users. Unlike existing distribution mechanisms, the automatic verification of miners discussed in Section III-B allows the repository owner to focus entirely on the high-level privacy implications of miner policies, assured that the code cannot subvert it.

C. Anonimization, Blocking Techniques and Privacy Modes

Recently, major browsers have come to support some form of a "private browsing mode" [1]. Although the precise meaning of this term varies between browsers, the common idea behind this feature is to prevent web sites from reading persistent data such as cookies for a particular session. There have been a number of other browser add-ons and modifications that attempt to anonymize the user on the web; an incomplete list includes TrackMeNot [10], Torbutton, SafeCache [24], SafeHistory [24], and IE8's InPrivate browsing. While it is clear that a truly anonymous browsing mode would force content providers to use REPRIV, no such mode has been successfully implemented [1], and it is not clear that doing so is technically feasible [5]. However, we assert that REPRIV does in fact facilitate end-user privacy on the web, by creating incentives for content providers to use privacy-sensitive personalization techniques, rather than relying on the invasive collection mechanisms currently available. In this respect, REPRIV is complementary to private browsing modes; it provides a mechanism for allowing personalized content without the need for the tracking mechanisms currently used by content providers, which are not compatible with anonymous browsing.

D. Profile Management

The behavior profiles generated by REPRIV are currently maintained entirely within the browser, and are distinct on a per-user, per-browser basis. However, there is no reason to preclude additional profile management schemes in REPRIV. One possibility is to maintain the primary copy on a cloud server, encrypted using a symmetric key. Because the cloud does not need direct access to profile data, key distribution for this scheme is straightforward: the user manually loads the symmetric key into each browser that updates or consumes the profile; this is realistic assuming the user is physically present at each browser that accesses the profile. Updates to the personal profile are performed locally at each browser instance, and synced with the cloud server periodically. The major upshot of this scheme is that the behavior profile is no longer constrained on a per-browser basis, as the user can transfer the same profile between multiple instances using the cloud host.

VIII. Related Work

A. Privacy and Web Applications

As a reaction to the decrease in privacy on the web, many have started exploring techniques that can be applied to restore some degree of privacy while still allowing for the rich web applications that people have come to expect. Jakobsson *et al.* [12] considered the problem of third-party sites mining users' navigation history. They developed a system that allows third parties to learn *aggregate* information about users' navigation histories, rather than the full listing. All privacy assurances offered by this system derive from the fact that its mechanism is easily auditable by end-users, so parties who wish to mine history data have disincentive to cheat.

Becker and Chen [2] found that it is possible to deduce specific personal characteristics given only a list of their friends on a social network. Worse yet, they found that it is very difficult to defend against this type of inference, assuming an attacker has access to the user's entire social graph: on average, they found that users would have to remove hundreds of friends from their connections in order to ensure the privacy of their own characteristics.

Narayanan and Shmatikov [21] studied the privacy implications of social network participation. Their observation is that the operators of online social networking sites now share user data with third parties, but only scrub personally-identifying information in an ad-hoc fashion. They developed a *re-identification* algorithm that relates users' privacy in a social network to node anonymity in the social network graph, and attempts to identify particular users from scrubbed social network data. They found that if a user subscribed to both Twitter and Flickr, then the algorithm can correctly identify them with 88% accuracy.

McSherry and Mironov [18] attempted to restore a certain degree of privacy to collaborative recommendation algorithms, such as those used by Netflix and amazon.com. Citing the work of Narayanan and Shmatikov [20] in de-anonymizing users who take part in such systems, they worked in the framework of *differential privacy* [4] to build a an algorithm that preserves the privacy of each individual rating entered by a participating user. The performance is comparable to that of the original Netflix recommendation algorithm.

B. Privacy in Advertising

One problem that has received much recent attention is that of delivering targeted advertisements to web users without violating their privacy. Freudiger *et al.* [6] observe that the prevalent mechanism for targeting advertisements to individual users is the *third-party cookie*. They propose a browser extension that allows users to directly manage third-party cookies in order to decide the degree to which advertisers are able to track them. However, unlike with REPRIV, this solution does not give users arbitrary, fine-grained control over the type of information that is given to third-parties. Furthermore, advertisers have no incentive to obey the privacy safeguards instantiated by this mechanism. In a slightly different vein, several recent systems [9, 13, 28] attempt to remedy the problem by storing the necessary sensitive personal data on the client, along with all possible ads in the network. When an ad is displayed, it is matched to personal information locally, thus sidestepping the need to leak to the ad network. Accounting and click-fraud prevention are addressed using either additional semi-trusted parties, or homomorphic encryption. The primary difference between these systems and REPRIV is generality: REPRIV asks the user to provide content providers (in this case, an advertising network) with small amounts of selected personal data in return for full application generality, whereas these tools effectively hide all personal data needed to drive the single application of targeted advertising.

C. Managing Private Browser State

A number of researchers have studied ways to identify users and preferences from browser interactions. Wondracek *et al.* [30] found that a subtlety in the W3C specification that allows browser history to be inferred can be leveraged to de-anonymize users of popular social networking sites. Jackson *et al.* [11] attributed the problem of history sniffing to the fact that browsers do not extend the same-origin policy to the history state leveraged in the attack. Recently, Mozilla has taken steps to prevent history sniffing [27], at the cost of breaking certain parts of the W3C specification. In a broader development, Eckersley [5] introduced a technique dubbed *browser fingerprinting*, wherein a large number of publicly-visible browser attributes are combined to produce an identifying string shared by only one in $\sim$286,777 browsers.

Several researchers have approached the technical problem of maintaining user anonymity while browsing. Howe and Nissenbaum [10] created TrackMeNot, a Firefox extension that attempts to anonymize search behavior by periodically submitting random search queries to major search engines. McKinley [17] examined the privacy modes of popular browsers, as well as their ability to clear private state when directed by the user. She found that while some browsers do in fact clear private state when instructed, none of the browsers' privacy modes performs as advertised; each browser left some form of persistent state that could be later retrieved by web pages in different browsing sessions.

D. Web Personalization and Mining

The basis on which personalization is performed varies from application to application. Pierrakos *et al.* [23] surveyed the topic of mining users' behavior on a set of web services to infer information that will aid personalization. They found that almost all web personalization efforts fall into one of four broad categories: memorizing information for later replay, guiding the user towards likely relevant information, customizing content to match users' interests, and supporting users' efforts to complete tasks. REPRIV is designed primarily to support the implementation of the second and third points, but it can be used to support aspects of all types of personalization.

There are several browser add-ons (*toolbars*) that perform data collection and user behavior mining. Perhaps the most popular among them is the Alexa Toolbar, which for each user collects a complete browsing history, search engine query list, and summary of the advertisements presented to the user. This information used by Alexa to compute a number of analytic functions, some of which are returned to toolbar users as a service. Among the analytics are traffic statistics (including a comprehensive, internet-wide ranking of popular sites), related links, audience demographics, and clickstream statistics. Similarly, Bing [3], Google [8], and Yahoo [32] all offer toolbars, although they vary in the amount of mining and automatic personalization that they perform.

IX. CONCLUSIONS

This paper presents REPRIV, an in-browser approach that aims to perform personalization without sacrificing user privacy. REPRIV accomplishes this goal by requiring explicit user consent in any transfer of sensitive user information. We showed how efficient and effective behavior mining can be added to a web browser to automatically infer the information needed to facilitate many personalized web applications, and evaluated this mechanism on real-world data. We also showed how third-party code can be incorporated into the system, and given access to sensitive user information, without sacrificing control and the possibility of user consent. Finally, we presented two end-to-end case studies of useful personalized applications, that showcase the abilities of REPRIV. We evaluated several aspects of these case studies over data collected from real browsing sessions, as well as human participants. Given our results, we are able to conclude that REPRIV allows a wide range of personalized web applications to exist, without requiring the user to sacrifice control over their personal information: personalized content and privacy can coexist on the web.

REFERENCES

[1] G. Aggarwal, E. Bursztein, C. Jackson, and D. Boneh. An analysis of private browsing modes in modern browsers. In *Proceedings of the Usenix Security Symposium*, 2010.

[2] J. Becker and H. Chen. Measuring privacy risk in online social networks. In *Proceedings of the Workshop on Web 2.0 Security and Privacy*, 2009.

[3] The Bing Toolbar. http://www.discoverbing.com/toolbar.

[4] C. Dwork. Differential privacy: a survey of results. In *Proceedings of the International Conference on Theory and Applications of Models of Computation*, 2008.

[5] P. Eckersley. How Unique Is Your Web Browser? Technical report, Electronic Frontier Foundation, 2009.

[6] J. Freudiger, N. Vratonjic, and J.-P. Hubaux. Towards Privacy-Friendly Online Advertising. In *IEEE Web 2.0 Security and Privacy)*, 2009.

[7] Google AdSense privacy information. http://www.google.com/privacy_ads.html#toc-faq.

[8] The Google Toolbar. http://toolbar.google.com.

[9] S. Guha, A. Reznichenko, K. Tang, H. Haddadi, and P. Francis. Serving Ads from localhost for Performance, Privacy, and Profit. In *Proceedings of Hot Topics in Networking*, 2009.

[10] D. C. Howe and H. Nissenbaum. TrackMeNot: Resisting surveillance in web search. In I. Kerr, V. Steeves, and C. Lucock, editors, *Lessons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society*, chapter 23, pages 417–436. Oxford University Press, 2009.

[11] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell. Protecting browser state from web privacy attacks. In *Proceedings of the 15th International Conference on World Wide Web*, 2006.

[12] M. Jakobsson, A. Juels, and J. Ratkiewicz. Privacy-Preserving History Mining for Web Browsers. In *Proceedings of the Workshop on Web 2.0 Security and Privacy*, 2010.

[13] A. Juels. Targeted advertising ... and privacy too. In *Proceedings of the Conference on Topics in Cryptology*, 2001.

[14] B. Lerner, H. Venter, B. Burg, and W. Schulte. An experimental extensible, reconfigurable platform for html-based applications, Oct. 2010.

[15] B. Livshits and L. A. Meyerovich. ConScript: Specifying and enforcing fine-grained security policies for JavaScript in the browser. Technical Report MSR-TR-2009-158, Microsoft Research, Nov. 2009.

[16] McAfee Inc. Spyware information. http://www.mcafee.com/us/security_wordbook/spyware.html.

[17] K. McKinley. Cleaning Up After Cookies Version 1.0. Technical report, ISEC Partners, 2010.

[18] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2009.

[19] Amazon Mechanical Turk. https://www.mturk.com/mturk/welcome.

[20] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2008.

[21] A. Narayanan and V. Shmatikov. De-anonymizing social networks. *IEEE Sympolsium on Security and Privacy*, 2009.

[22] The Open Directory Project. http://dmoz.org.

[23] D. Pierrakos, G. Paliouras, C. Papatheodorou, and C. D. Spyropoulos. Web usage mining as a tool for personalization: A survey. *User Modeling and User-Adapted Interaction*, 13(4), 2003.

[24] Same origin policy: Protecting browser state from web privacy attacks. http://crypto.stanford.edu/safecache/.

[25] N. Swamy, J. Chen, and R. Chugh. Enforcing stateful authorization and information flow policies in fine. In *In Proceedings of the European Symposium on Programming*, 2010.

[26] TargetAPI. http://www.targetapi.com.

[27] The Mozilla Team. Plugging the CSS History Leak. http://blog.mozilla.com/security/2010/03/31/plugging-the-css-history-leak, 2010.

[28] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings of the Network and Distributed System Security Symposium*, Feb. 2010.

[29] WebMii: A person search engine. http://www.webmii.com.

[30] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *IEEE Symposium on Security and Privacy*, 2010.

[31] Yahoo! BOSS API. http://developer.yahoo.com/search/boss/.

[32] The Yahoo Toolbar. http://toolbar.yahoo.com.