Privacy-Preserving History Mining for Web Browsers

Markus Jakobsson
PARC
Palo Alto, CA, USA
mjakobss@parc.com

Ari Juels
RSA Laboratories
Bedford, MA USA
ajuels@rsa.com

Jacob Ratkiewicz
Dept. of Computer Science
Indiana University
Bloomington, IN USA
jpr@cs.indiana.edu

ABSTRACT

We introduce a new technique that permits servers to harvest selected Internet browsing history from visiting clients. Privacy-Preserving History Mining (PPHM) *requires no installation of special-purpose client-side executables*. Paradoxically, it exploits a feature in most browsers (IE, Firefox and Safari) regarded for years as a privacy vulnerability. PPHM enables privacy-preserving data-mining through the addition of a client-side filter that supports OR and AND queries over the URLs cached in a client.

We describe a lightweight prototype PPHM system designed for targeted advertising. We also discuss audit and policy enhancements that help our PPHM system comply with regulatory guidelines like the OECD Fair Information Practice Principles.

Keywords

detection, browser history, server-side, privacy-preserving

1. INTRODUCTION

Targeted advertising is the major engine of profit for search engines, mail services, and news sites on the Web. Search engines, for example, serve advertisements according to the search queries of a user. Some sites also rely on third-party services that track user behavior across sites. Doubleclick, acquired by Google in 2007, is perhaps the best known service that profiles users by tracking them across networks of affiliated sites.

Doubleclick harvests data passed from clients to servers in the form of cookies. Clients themselves, however, carry rich seams of behavioral data on users in their browers. Web browsers like IE, Firefox, and Safari cache a complete list of the URLs visited by a user over a period of time (typically nine days) to facilitate completion of URLs typed into the navigation bar, and to allow users to use the "back up" feature of the browser. These lists, known as *histories*, are in principle attractive data repositories for personalization services like targeted adverting. While services like Doubleclick can only harvest navigation data over participating sites, browser histories are comprehensive, containing all navigation information over their windows of coverage.

Unlike cookies, browser histories are for the most part exploitable only by clients. Our main focus in this paper, though, is a technique called Privacy-Preserving History Mining (PPHM) that allows server-side exploitation of browser histories. PPHM requires *no client-side software*; it relies exclusively on server-side implementation. The advantages of this approach are clear: Our system is perfectly transparent to users and avoids the complications and burden of user-mediated software installation. This simplifies deployment and avoids stoking the dangerous habit in users of downloading potentially troublesome executables from the Internet.

The basis for our PPHM system is a web-browser feature that we refer to as *URL probing*. As is well understood, a server can — by means of URL probing — ascertain the presence of a particular URL in the browser cache of a visiting client. For example, a server can determine if a visiting client has visited the webpage `www.xyz.com/somepage.html` in the past few days. It is important to note that URL probing only reveals information in the case of an *exact match*. For example, if a client has visited `www.xyz.com/otherpage.html`, a server will not learn this fact by probing the client's browser history for the URL `www.xyz.com/somepage.html`.

As the research community has recognized for some years, URL probing can serve as a vehicle for privacy abuses [3, 2], as it permits servers to harvest data for which they have no legitimate right or use. Some researchers have proposed browser-modifications to restrict URL probing. For example, [4] proposes a browser modification that limits the feature according to same-origin policies. Others [5] have suggested server-side defenses based on making the visited URLs difficult to anticipate to would-be attackers. We emphasize that our PPHM proposal does not create or exploit a new form of URL probing, but makes use of an existing one as a new security tool.

We propose several mechanisms for protecting user privacy in PPHM deployments. The most important is *filtering*. We introduce special techniques that permit a server to learn only relevant, aggregate information about URLs in a user's browsing history—*without communicating information about specific, visited URLs*. For a list L of sites, our techniques allow a server either: (1) To determine if a user has visited *at least one* site in L, without revealing which one (i.e., to perform an OR operation over L) or (2) To determine if a user has visited *all* of the sites in L (i.e., to perform an AND operation over L). Given these two operations, one can implement any monotonous boolean function, although it is a topic of future research how to do so efficiently. Furthermore, we propose simple opt-in mechanisms for user participation in the PPHM process, and describe mechanisms that permit public auditing of PPHM imple-

mentations.

PPHM can enhance any Web service that relies on user behavioral data. For example, using PPHM (alongside other techniques):

- A *social networking site* can harvest information about users' preferences to connect them with other like-minded users.
- A *security site* can scan clients to alert users if they have recently visited known phishing sites.
- An *e-commerce or content-delivery site* can scan visiting clients to serve them better targeted advertisements.
- A *financial services site* can determine the exposure to risky sites (such as phishing sites and sites known to distribute malware) that a given user has, and use this information to guide decisions about what transactions to give particular scrutiny. Note that PPHM cannot directly detect malware; rather, it can detect that the user has visited known malware distribution sites. Further, if a particular kind of malware is known to cause visits or bookmarks (which count as visits in some browsers) to a number of sites, PPHM can detect this as well.

In this paper, we focus on targeted advertising as our example application.

The major limitation of PPHM is the verbatim form of server probes. A server cannot upload a browser history, but can only make yes / no queries against it. As our prototype demonstrates, it is feasible for a server to make only a few hundred queries without introducing noticeable browsing delays. Our challenge is to devise a system that is: (1) Effective in extracting meaningful data from clients using a limited number of yes/no queries, and also (2) Strongly protective of users' privacy.

Organization

In section 2, we describe the intuition of our approach by sketching a simple example targeted-advertising campaign. In section 3 we survey related work and background literature. We describe an initial PPHM implementation in section 4. In section 5, we discuss surrounding privacy and ethical considerations. We conclude in section 6 with some further research directions.

2. EXPLICATION BY EXAMPLE: AN ADVERTISING CAMPAIGN

As an example of how our PPHM system might work, let us consider a toy advertising campaign. Let us suppose that the search site `A.com` serves ads for two (corporate) customers: A chocolate shop (CS) and a stock broker (SB). So the task of `A.com` is to determine whether a given visiting user is a better match for a CS ad or an SB ad.

The first and simplest approach would be for `A.com` to see if a visiting client has visited either: (1) Any of a list L_1 of chocolate-related sites (e.g., `www.hersheys.com`, `www.seventypercent.com`, etc.) or (2) Any of a list L_2 of stock brokers and/or financial publications (e.g., `vanguard.com`, `barrons.com`, etc.). If a user has visited a site in L_1 but not one in L_2, then `A.com` displays a CS ad. If a user has visited a site in L_2, but not one in L_1, then `A.com` displays an SB ad. Otherwise, `A.com` chooses randomly between SB and CS.

The naïve approach to checking whether a user has visited a site in L_1 or L_2 is to probe a client's browser history individually for each URL in the target list. While this approach will reveal the information needed for `A.com`'s ad-serving policy, it constitutes a potentially unacceptable compromise of privacy. After all, `A.com` will learn *all* of the sites in the target list that the client has visited. As explained above, however, our techniques allow a server to filter its query on the client side (using crafted HTML, as detailed below) so that it learns only whether a client has visited *at least* one site in L_1 or L_2, but no specifics about *which* site or sites. Such "OR" filtering allows `A.com` to harvest exactly the information needed for its advertising policy, and no more. As explained below, a client can also inspect the HTML served by `A.com` to ensure that it is only sending "OR" queries.

`A.com` may wish to refine its data-mining strategy to determine whether SB or CS is the better match for the user for cases in which the user has visited both L_1 and L_2, i.e., when its first query turns up no decisive information about a user. One approach would be for the site to deploy two new lists, L_1' and L_2'. L_1' is a short list of "must-visit" sites for chocolate lovers—those that no chocolate lover would omit. Similarly, L_2' is a list of sites that serious equity traders would not miss. `A.com` might then query the client to determine if it has visited *all* of the sites in L_1' or has visited *all* of the sites in L_2', and given a hit for exactly one of these two lists, serve either an SB or CS ad accordingly.

We have, of course, described a rather simplified advertising here. Our interest is not to describe the mechanics of ad targeting, but rather the mechanics of the privacy-protecting querying language we have developed for PPHM. Two aspects of our example are noteworthy. First, we have described an *adaptive* advertising strategy. By making a second query conditioned on its first, `A.com` is able to target users more precisely. Of course, a policy can be arbitrarily adaptive. The only limitation is the number of PPHM queries that a service can make to a user's browser without inconveniencing users. (The matter of inconvenience, here, would be a potential delay — the user would never have to react to or become aware of queries sent by the server performing the scan.)

More sophisticated advertising strategies are also possible through the mining of indirect information. For example, `A.com` might query users on a list L_3 of candy sites—not chocolate-specific ones—or might tailor its queries or ads based on other contextual information like a user's geographical location as determined by IP address. Further, `A.com` might use (anonymous) cookies to maintain user identity across sessions, effectively lengthing its probing window beyond the standard nine-day cache lifetime of browser histories while retaining user anonymity.

Of course, as illustrated here, PPHM imposes austere limits on the information accessible to a server. If `A.com` required client installation of a special-purpose executable, it could simply upload a client's history and mine it as desired (not to mention enforcing the maintenance of a richer, longer history). The limitations of PPHM, again, arise as a tradeoff against its benefits: The lack of special client-side software and the privacy filtering it allows.

3. RELATED WORK

Our PPHM techniques rely on server-side probing of client browser caches to determine what URLs a user has visited. Such probing was first studied by Felten and Schneider [3], who considered invasive forms of timing analysis. SecuriTeam [2] later described more accurate privacy-infringing techniques that directly access browser history files. Jackson et al. [4] have outlined client-side defenses against such infringements. Drawing on another recent vein of research [7], our PPHM work explores the flip-side of browser probing, namely how it can create *benefits* for users using browser histories.

Our proposed tool is effectively a filtered data-mining service. Social networking sites such as Facebook and LinkedIn harvest data directly from server-stored user profiles. Online mail services such as Hotmail and GMail mine users' e-mail content to serve ads. Servers without access to such rich data sources tend to rely instead on cookies as a means for establishing user browsing patterns. First party cookies are commonly used to identify repeat visitors, and customize their experience based on past actions; third-party cookies and first-party cookies with an aggregator (such as Doubleclick) are used to track browsing patterns across domains. In contrast, PPHM does not identify users, but simply attempts to identify browsing patterns and online accesses.

A number of widely deployed systems relay potentially sensitive client information to centralized data repositories. For example, the Microsoft operating system relays information on software failures to Microsoft; the system explicitly prompts users for permission before doing so. A number of toolbars, such as the Alexa toolbar and the Google toolbar, perform centralized harvesting of the browsing behavior of individual users. Users assent to the transmission of this information by merit of their installing the toolbars; all browsing activity is associated with a client-specific identifier, communicated and recorded. In comparison, our tracking is substantially less invasive, and can—where appropriate—dispense entirely with identities or pseudonyms. More importantly, our techniques involve dynamic *filtering* of data on the client to ensure protection of user privacy, in the spirit of academic proposals like, e.g., [6].

4. IMPLEMENTATION

4.1 Intuition

In broad terms, our method is as follows. When the user requests a page from an PPHM server, the server returns a page that, in addition to the normal content, defines an invisible section of the page containing a number of links. Each of these links is given a CSS style that causes the browser to "call home"— that is, to notify the server if the link has been visited. This is implemented by directing the browser to download a CSS style sheet from the server in order to color visited links. The server is then able to tell which links are visited by tracking which style sheets are downloaded. Due to the fact that browser caching normally prevents the same URL from being fetched twice in a page load, user privacy can be preserved by a careful partitioning of the links to be detected into classes, each of which is given the same "call home" link.

4.2 Proof-of-concept details

We first outline the server-side infrastructure, then how detection is carried out on the client side. Of particular interest are our techniques for protecting client privacy during the scanning process. In particular, we wish to avoid the possibility of performing scanning on a customer who has explicitly agreed to it, despite potential advantages to the scanner, e.g. in detecting malware before it has the opportunity to see a user's login. Thus we assume in the following that the user has opted in, perhaps in one of the following ways:

- If PPHM is to be employed to detect malware, the user should opt-in when they open their account (perhaps with a bank or other financial institution). Scanning will then take place after the user authenticates, but before sensitive parts of their account are unlocked. If malware is detected, the server is thus able to take protective measures such as suspending access to some account features or even disabling the account. Again, though, we emphasize that PPHM provides an indication of risks like malware infection, not a definitive litmus test.

- If PPHM is to be employed to serve better targeted advertisements or for some such similar goal, opt-in and opt-out could be implemented by a cookie set in the user's browser if they chose to opt-in. The PPHM server could check for the cookie and only transmit code to perform the scanning if it was found. Should the user later wish to opt out, they could revisit the site and choose to do so, or simply delete the cookie.

Our server-side PPHM implementation consists of three scripts:

`pphm.cgi` This script produces a page that invisibly performs detection, refreshing when complete to the page that shows the results (`action.cgi`).

`callhome.cgi` This is the script that is notified when a class of links is triggered. It is responsible for keeping track of the classes of links triggered by a particular client.

`action.cgi` This script is called when the logged-in user attempts to perform another action. It consults the database maintained by `callhome.cgi` to determine which of a predetermined list of advertisements should be served to a user. (In our prototype, we don't actually serve any advertisements. We have only implemented a framework for doing so.)

We now discuss each of these scripts in turn. Figure 1 illustrates the sequence of events in a single detection session.

4.2.1 pphm.cgi

This script is responsible for performing the browser reconnaissance, which it accomplishes as follows. It first reads a description of the links it should detect from an XML file on the server. This file defines a number of link classes, each with a unique name and a type, which may be "AND" or "OR." Each link class also contains a list of URLs. If the type of the class is "AND," the class registers a hit when *all* of its associated URLs have been visited by the client's browser; if its type is "OR," it registers a hit when *any* of its links are visited. Note that each class registers a binary

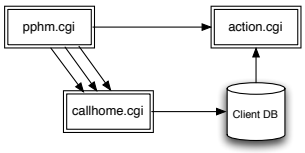


Figure 1: Sequence of events in a single detection session. `pphm.cgi` produces a page which is interpreted by a user's web browser. CSS code in the web browser causes communication with `callhome.cgi`, which registers any hits in a database associated with each client. When the user visits `actions.cgi`, this database is consulted to make a final determination as to which advertisement should be served to the client.

result, either no hit or one hit, regardless of the number of visited links in the corresponding class. How each class of links is detected is outlined below; further details on the link-detection techniques we employ are available in [5].

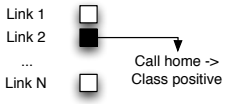
In each of the following discussions, we assume that each user has a unique ID number. In our demo system this is computed by incrementing a persistent ID, but other unique numbers (such as a customer ID number) would serve equally well. This ID is indicated in URLs by `XX`. Figure 2 gives a graphical depiction of each scenario.

Detecting an OR-class

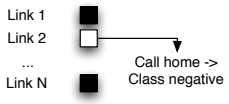
Suppose that the detection XML file specifies a class of type OR that contains the links `www.a.com`, `www.b.com`, and `www.c.com`. Thus, this class should produce a single hit if any of these three sites has been visited by the user. This is implemented by the following CSS and HTML code, which is automatically generated by `pphm.cgi`:

```
<style type="text/css">
#someclass:visited {
  background: url(
   'http://www.server.com/callhome.cgi?id=XX&
                      class=someclass&type=or' );
}
...
</style>
...
<a href="http://www.a.com" id="someclass"></a>
<a href="http://www.b.com" id="someclass"></a>
<a href="http://www.c.com" id="someclass"></a>
```

When the HTML-rendering subsystem of the user's browser renders the page, it consults the CSS code to determine how to format the first visited link (if any of the links are visited). This results in the URL given as the link background being loaded, causing `callhome.cgi` to be called. This registers with the server the fact that one of the links was visited, but the server does not learn *which* link. Even if the user has visited multiple links in this class, the client will load the link-background URL at most one time; once loaded, the link persists in the browser's cache. Thus, *the server will also never know how many of the links in the class the user has visited*, and the user's privacy is preserved. Note that even attacks such as timing attacks will be ineffective in general, as it is not a requirement that the browser render all links in the order that the appear in the document.



(a) Detecting links in an OR class. The server is signalled by the first visited link. Note that subsequent visited links will not signal the server, as they will be handled by browser cache; thus, the server cannot tell how many of the links were visited.



(b) Detecting links in an AND class. The server is signalled by the first *un*-visited link. Note that at most one signal will be recieved, so the server will not know which (or how many) links were unvisited.

Figure 2: Privacy-preserving detection of visited links by means of link classes. In each case, the server can tell only if the conditions of the class are satisfied; not the degree to which they are (or are not). Note that these strategies are through application of a variant of De Morgan's laws for propositional formulae.

Detecting an AND-class

Preserving the user's privacy in this case is slightly more involved. Suppose in this example that we wish to tell if a user has visited sites `www.x.com`, `www.y.com`, and `www.z.com`. We wish the server to be notified if the user has been to *all three* of the sites; if the user has been only to some subset of the sites, we wish to know nothing. The following CSS and HTML perform this detection with a class named `anotherclass`:

```
<style type="text/css">
#anotherclass:link {
    background:url(
       'http://www.server.com/callhome.cgi?id=XX&
                      class=anotherclass&type=and' );
}
...
</style>
...
<a href="http://www.x.com" id="anotherclass"></a>
<a href="http://www.y.com" id="anotherclass"></a>
<a href="http://www.z.com" id="anotherclass"></a>
```

Note that here, the CSS class used is `link` rather than `visited`. This is the class that is used to specify the appearance of *un*-visited links. Now, the PPHM server receives exactly one request if any of `x.com`, `y.com`, or `z.com` are *un*-visited. If the server receives a request, the server knows that *at least one* (and possibly all) links are unvisited, and thus the class is not a hit. If it receives no request, it knows that all links are visited. The fact that the class is a hit when the server receives *no* requests means that the server must adopt a "guilty until proven innocent" approach in determining if the class is a hit. Thus, if a user's browser crashes, or the user navigates away from the page, before detection is complete, the class might erroneously be marked as a hit. This can be avoided by adding a reference to a second URL at the end of the list of URLs to be scanned. The second reference would be unconditional, and so, would be made

whether the first call was made or not. While there is no guarantee that the browser will render the links in order, this is a heuristic solution. One can tolerate a certain probability of misclassification: targeted advertising is always a hit-or-miss affair.

4.2.2 callhome.cgi

This script is triggered by all CSS styles. It simply records in a database the user's unique ID, as well as the set of classes that the user has triggered.

4.2.3 action.cgi

Called when detection is finished, this script consults the database built by `callhome.cgi` to determine the appropriate advertisement to serve to a user.

In the above, it might at first seem easier to use client-side JavaScript, rather than the combination of client-side CSS and HTML, to harvest the data. However, to the authors' knowledge, there is no way for CSS to communicate with JavaScript code in browsers other than Internet Explorer.

4.3 Implementation Alternatives

4.3.1 Fuzzy PPHM

At the cost of some loss in privacy, we can broaden our privacy-preserving PPHM techniques that a server can determine if a client has visited at least k in a target list of n sites. By returning a randomized number of spurious hits to the list, a client-side script can conceal the exact number from the server. (This technique resembles a number of other statistical privacy-preserving approaches, e.g., [1].)

Let $U = \{u_1, \ldots, u_n\}$ be a set of target URLs. Let us suppose that the server wishes to detect clients that have visited at least k URLs in U. The server sends the client a script that does the following:

1. Makes a call to X.cgi (a client-specific CGI script) for each $u \in U$ that the client has visited.
2. Makes $r \in_U [0, m]$ additional calls to X.cgi.
3. Accepts a single query r' from the server, and returns '1' if $r \geq r'$ and '0' otherwise.

On receiving $R > k$ total hits to X.cgi from a client, the server computes $r' = R - k$, and sends the query r' to the client. The server determines that the client has visited k links in U if the client returns a '1.'

The degree of privacy preservation in this scheme depends on the number of client visits v to sites in U, as well as the value of m. (It also depends on any *a priori* server knowledge about v for a given client.) In general, a client enjoys the minimum degree of privacy when $v = k$; in that case, the server learns v exactly with probability $1/m$. We believe, however, that m can be made fairly large–on the order of thousands–thereby acceptably minimizing exposure of client data. Of course, r can be drawn from a non-uniform probability distribution to meet particular privacy needs.

It is important to note that this this rather roundabout method of determining a bound on the number of visited links is necessitated by the security policies in modern browsers. Step 1 must necessarily be accomplished by means of CSS style sheets on the client side, which cannot communicate with programmatic web page elements such as Javascript; only with the server by the method this paper describes. Steps 2 and 3, however, must be accomplished by client-side Javascript. Thus, it is not possible for the query in step 3 to relate to the number of visited links, as the Javascript used to implement step 3 cannot know this information.

4.3.2 Non-standard browsers

In addition to the above browser-independent approach, we outline some features of particular browsers that can enhance detection on these browsers.

- Internet Explorer version 6 (but not version 7) allows CSS descriptors to call JavaScript functions. This allows finer-grained detection (such as the "fuzzy scan" described above) without any compromise in privacy, as all computation can be done on the client side.
- In Safari, a link which is bookmarked counts as permanently visited (while a normally-visited link reverts to unvisited after a few days). This is beneficial to malware detection efforts, as malware that affects browsers often adds sites as bookmarks as well as visiting them. Under Safari, such sites would be detected even long after the initial malware installation due to this feature.

5. PRIVACY

A common approach to protection of consumer data is anonymization (or pseudonymous identification). This approach is impractical, however, for many of the environments in which PPHM can be most beneficially deployed. Online e-commerce web sites, in particular, require authentication of user identities in the course of transactions. As we have explained, therefore, data-filtering is the crux of our approach to privacy. The granularity and exact nature of PPHM filtering may of course be adjusted to meet various security and privacy needs, and ultimately depends upon the policy of a PPHM deployer. We propose two other important measures, however, that empower users to monitor and control their privacy in a PPHM environment:

1. **User notice and consent:** We recommend that PPHM be deployed on a strict opt-in basis. Potential users should be notified of the goals and operating parameters of an PPHM system before they are enrolled in an PPHM program. Only when an informed user has explicitly consented to enrollment in writing or on a web form should PPHM scanning of her browser take place. Furthermore, we believe that a server should perform PPHM scanning of the browser of a consenting user only after adequately authenticating the user as well as her client machine. This requirement helps ensure that an PPHM deployment does not transgress the bounds of user consent by scanning a client machine or operating environment that does not belong to the user.
2. **Auditability:** Because our PPHM system harvests information through a CSS style sheet and HTML document downloaded by the client, *any client can determine exactly what information the PPHM system gathers and reports* simply by viewing these documents. Most users of course lack the motivation and/or expertise to read HTML or CSS. Detection of widespread

abuse in a system of this kind, though, requires only that a small number of users monitor its behavior. PPHM-related code can be short and transparent.

It is a concern that this approach does rely on the server to self-regulate to some degree. While in principle the server may be audited by savvy users at any time, in reality most users will not bother. Of course an open-source client-side browser plugin that verifies good behavior on the part of PPHM could be developed, but this defeats the purpose of PPHM in that it forces the user to install client-side software. The user should keep in mind, when consenting to PPHM, that he or she must trust the server to scan in a privacy preserving way. Of course, this privacy model is no worse (and indeed arguably better) than that of browser plugins and toolbars which users install; these are afforded far more power over the user's machine and are less auditable than PPHM if not open-source.

Fair Information Practice Principles. The Organization for Economic Co-Operation and Development enunciated a set of eight basic privacy principles in 1980 [8] that have served as a basis for a number of subsequently published Fair Information Principles (FIP), such as the EU privacy directive and the FIP guidelines of the United States Federal Trade Commission (FTC). Appendix A enumerates the eight principles in an excerpt from the original OECD document. These principles are an excellent touchstone for gauging the privacy properties of a system.

We believe that PPHM may be readily deployed in compliance with the OECD principles and related FIPs. Referring the reader to Appendix A, we note that our filtering approach helps support the Collection Limitation and Use Limitation Principles by strictly limiting the amount of information harvested by a server. By revealing exactly what information the system scans and collects, the auditability property of PPHM supports the Accountability and Purpose Specification Principles. Compliance with the remaining OECD principles requires careful notice and consent for consumers, as well as careful data security practices around PPHM-enabled servers.

6. CONCLUSION

We have proposed a privacy-preserving auditing tool that permits a server to determine if a client has recently visited webpages from specified sets of URLs. Our proposed technique allows AND and OR operations over the members of the sets, allows simple auditing of functionality, and can be built to support opt-in mechanisms.

7. REFERENCES

[1] Agrawal, R., and Srikant, R. Privacy-preserving data mining. In *ACM SIGMOD Conference on Management of Data* (2000), ACM Press, pp. 439–450.

[2] Clover, A. Timing attacks on Web privacy (paper and specific issue), 20 February 2002. Referenced 2008 at `http://www.securiteam.com/securityreviews/5GP020A6LG.html`.

[3] Felten, E. W., and Schneider, M. A. Timing attacks on Web privacy. In *ACM Conference on Computer and Communications Security* (2000), ACM Press, pp. 25–32. Referenced 2008 at `http://www.cs.princeton.edu/sip/pub/webtiming.pdf`.

[4] Jackson, C., Bortz, A., Boneh, D., and Mitchell, J. Protecting browser state from web privacy attacks. In *Proceedings of The 15th annual World Wide Web Conference (WWW2006)* (2006), pp. 737–744.

[5] Jakobsson, M., and Stamm, S. Invasive browser sniffing and countermeasures. In *Proceedings of The 15th annual World Wide Web Conference (WWW2006)* (2006), pp. 523–532.

[6] Juels, A. Targeted advertising ... and privacy too. In *RSA Conference – Cryptographers Track (CT-RSA)* (2001), D. Naccache, Ed., Springer-Verlag, pp. 408–424. LNCS no. 2020.

[7] Juels, A., Jakobsson, M., and Jagatic, T. Cache cookies for browser authentication (extended abstract). In *IEEE Symposium on Privacy and Security* (2006), pp. 301–305.

[8] Organization for Economic Co-Operation and Development. OECD guidelines on the protection of privacy and transborder flows of personal data, 1980. Referenced 2008 at `http://www.oecd.org/document/18/0,2340,en_2649_34255_1815186_119820_1_1_1,00.html`.

APPENDIX

A. OECD FAIR INFORMATION PRACTICE (FIP) PRINCIPLES

In its 1980 Guidelines on the Protection of Privacy and Transborder Flows of Personal Data [8], the OECD enunciated eight basic principles for data privacy. These have served as a basis for a number of other regulatory guidelines, which are often referred to generically as Fair Information Principles (FIP). The United States Federal Trade Commission FIP and the EU Privacy Directive draw on the OECD document. The eight principles, drawn verbatim from the OECD document, are:

1. *Collection Limitation Principle*: There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.
2. *Data Quality Principle*: Personal data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date.
3. *Purpose Specification Principle*: The purposes for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfilment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose.
4. *Use Limitation Principle*: Personal data should not be disclosed, made available or otherwise used for purposes other than those specified in accordance with [the previous principle] except:
   (a) with the consent of the data subject; or
   (b) by the authority of law.
5. *Security Safeguards Principle*: Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorised access, destruction, use, modification or disclosure of data.

6. *Openness Principle*: There should be a general policy of openness about developments, practices and policies with respect to personal data. Means should be readily available of establishing the existence and nature of personal data, and the main purposes of their use, as well as the identity and usual residence of the data controller.
7. *Individual Participation Principle*: An individual should have the right:
   (a) to obtain from a data controller, or otherwise, confirmation of whether or not the data controller has data relating to him;
   (b) to have communicated to him, data relating to him:
      - within a reasonable time;
      - at a charge, if any, that is not excessive;
      - in a reasonable manner; and
      - in a form that is readily intelligible to him;
   (c) to be given reasons if a request made under subparagraphs(a) and (b) is denied, and to be able to challenge such denial; and
   (d) to challenge data relating to him and, if the challenge is successful to have the data erased, rectified, completed or amended.
8. *Accountability Principle*: A data controller should be accountable for complying with measures which give effect to the principles stated above .