Research and Development in E-Business through Service-Oriented Solutions

Katalin Tarnay
University of Pannonia, Hungary & Budapest University of Technology and Economics, Hungary

Sandor Imre
Budapest University of Technology and Economics, Hungary

Lai Xu
Bournemouth University, UK

A volume in the Advances in E-Business Research (AEBR) Book Series

BUSINESS SCIENCE Reference
An Imprint of IGI Global

Published in the United States of America by
Business Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Research and development in e-business through service-oriented solutions / Katalin Tarnay, Sandor Imre and Lai Xu, editors.
pages cm
Includes bibliographical references and index.
Summary: “This book highlights the main concepts of e-business as well as the advanced methods, technologies, and aspects that focus on technical support for professors, students, researchers, developers, and other industryexperts in order to provide a vast amount of specialized knowledge sources for promoting e-business”--Provided by publisher.
ISBN 978-1-4666-4181-5 (hardcover) -- ISBN 978-1-4666-4182-2 (ebook) -- ISBN 978-1-4666-4183-9 (print & perpetual access) 1. Service-oriented architecture (Computer science) 2. Data mining. 3. Electronic commerce. I. Tarnay, Katalin. II. Imre, S?ndor. III. Xu, Lai, 1970-
TK5105.5828.R47 2013
658.8’72--dc23

2013009512

This book is published in the IGI Global book series Advances in E-Business Research (AEBR) Book Series (ISSN: 1935-2700; eISSN: 1935-2719)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 7
Tracking and Fingerprinting in E-Business:
New Storageless Technologies and Countermeasures

Károly Boda
Budapest University of Technology and Economics, Hungary

Gábor György Gulyás
Budapest University of Technology and Economics, Hungary

Ádám Máté Földes
Budapest University of Technology and Economics, Hungary

Sándor Imre
Budapest University of Technology and Economics, Hungary

ABSTRACT

Online user tracking is a widely used marketing tool in e-business, even though it is often neglected in the related literature. In this chapter, the authors provide an exhaustive survey of tracking-related identification techniques, which are often applied against the will and preferences of the users of the Web, and therefore violate their privacy one way or another. After discussing the motivations behind the information-collecting activities targeting Web users (i.e., profiling), and the nature of the information that can be collected by various means, the authors enumerate the most important techniques of the three main groups of tracking, namely storage-based tracking, history stealing, and fingerprinting. The focus of the chapter is on the last, as this is the field where both the techniques intended to protect users and the current legislation are lagging behind the state-of-the-art technology; nevertheless, the authors also discuss conceivable defenses, and provide a taxonomy of tracking techniques, which, to the authors' knowledge, is the first of its kind in the literature. At the end of the chapter, the authors attempt to draw the attention of the research community of this field to new tracking methods.

DOI: 10.4018/978-1-4666-4181-5.ch007

INTRODUCTION

Many Website operators involved in end-user oriented e-business have an interest in monetizing their user base (e.g., by realizing as many clicks on their advertisements as possible). One way of achieving this goal is collecting diverse information about the user (or *profiling* her), the vehicles of which are technologies that can be used for identifying returning visitors, and those that infer sensitive information (such as browsing history) that users are not necessarily willing to disclose (e.g., purchase preferences). This controversial method may be a necessity for better serving the needs of online customers, but it often goes beyond user demands, and is used for business purposes against the will of the clients.

It is not surprising that certain users are concerned about pervasive profiling. In fact, this problem has been discussed from so many viewpoints in academia that not only effective technological, but also more or less effective legislative countermeasures have been implemented in order to mitigate the privacy risks arising from the proliferation of a subset of profiling-related technologies. That said, this chapter is inspired by the rest (primarily by fingerprinting attacks), for which neither defensive technology nor the law seems to be able to keep up with the pace of development dictated by pro-profiling actors, leaving users of the Web powerless against and vulnerable to them.

PROFILING AND USER PRIVACY ON THE WEB

There are several motives of profiling users on the Web. For instance, an enormous number of Web services can be accessed for free; however, contrary to how it looks (or is communicated), free access often comes with a greater sacrifice of user privacy, as many of these companies gain revenue from profiling-related activities, such as pursuing behavioral advertising or monetizing user profiles by other means. According to the report of IAB Internet Advertising Revenue Report (IAB, 2012), the advertising revenues set a new record at $8.4 billion in Q1 2012, clearly showing the significance and the growth of the advertising industry.

In two recent studies, Goldfarb & Tucker (January and May, 2011) concluded that targeted advertising can successfully influence individuals in favor of buying a product, and targeted advertising has a significant positive (economic) impact on advertising. These claims are consistent with the ever-increasing presence of Web tracking techniques (the most used tools for profiling) and the revenues the industry reached in 2012, presumably with a continuously increasing proportion of behavioral advertising. By summarizing the measurements of Krishnamurthy & Wills between 2006 and 2012 (Krishnamurthy & Wills, 2006; 2009; Krishnamurthy, 2010), Mayer & Mitchell (2012) highlighted that the coverage on top sites of large tracking companies increased during these years, and so did the number of trackers per page. Today, researchers estimate that there are trackers capable of monitoring more than one fifth of user activity while browsing online (Roesner et al., 2012).

However, as can be expected, tracking for targeted advertising is not favored by users. A Harris Interactive poll (Krane, 2008), a TRUSTe survey (2009), and a nationally representative telephone survey in the USA conducted by Turow et al. (2009) uniformly confirmed that the majority of individuals (around 60% in all cases) found it uncomfortable when they faced advertisements on Websites adjusted to their preferences by previously observing their online activities. A more recent online survey conducted by McDonald & Cranor (2010) also confirmed this result with 55% of respondents rejecting targeted advertising; another recent survey conducted by TRUSTe in partnership with Harris Interactive (2011) reported a higher rate of respondents, namely 85%, who would not consent to being tracked for targeted advertising.

Nevertheless, targeted advertising is not the only goal of tracking and profiling; there are several other commercial use cases. Besides monetizing profiles in trades (i.e., exchanging large databases of pseudonymous profiles), they can be utilized to provide additional user interface features; for example, recommendations on news or Web shop items, customized user interfaces, or to facilitate price discrimination (dynamic pricing, i.e., adjusting price tags for customer price sensitivity for profit maximization of the vendor).

The case of Amazon.com–namely when the company offered the same DVD at different prices for different customers–is a frequently cited example of dynamic pricing (Davis, 2000). After customers discovered pricing differences while discussing their experiences in online forums, Amazon had to remove this feature and apologize for their experiment. Amazon claimed that prices were merely random, instead of being tailored for customer profiles.

Web users relate a bit ambivalently to dynamic pricing. Although it is a known and unpopular feature among them (Cranor, 2003), it becomes rather popular when it is communicated in the form of discounts: according to the survey of McDonald & Cranor (2010), 80% of respondents would consent to tracking to receive discounts tailored for their purchase interests, suggesting that price discrimination is more acceptable if it is communicated as a reward or bonus rather than a way of invading privacy.

Finally, we mention security, where tracking (and profiling) is applied in order to identify potential malicious users and prevent intrusions. In practice, this is pursued by many security-sensitive companies, such as banks or credit card companies. For instance, ThreatMetrix™ Cybercrime Defender Platform uses device identification to identify clients when they detect fraudulent access to prevent further attacks (ThreatMetrix, 2012); similarly, the Oracle Adaptive Access Manager fingerprints all types of devices (Oracle, 2011). Somewhat in contrast with user attitude towards commercial tracking, a survey of TRUSTe & Harris Interactive (2011) shows that 42% of respondents would consent to tracking to enhance security and to detect frauds. It must be noted that such identification can be used for quite different and questionable purposes, such as identification for censorship or surveillance, too.

Who Needs Users to be Identified?

There are several actors in the online era arguing against or in favor of identification. Normally, the only participant arguing against is the user herself, for whom identification is an undesired feature, up to the point when it brings benefits (it unlocks extra functionality or the user gains extra credit). On the other hand, there are numerous other participants having their own objectives and goals, who want users to be identified regardless of their privacy preferences.

As it has become clear so far, *advertising and marketing companies*, and *other parties* pursuing related activities are the most prominent actors (e.g., search engine optimization, retargeting, analytics services). Profiles can also be used in *web shops* for product recommendation systems, or for assisting pricing strategies, but arbitrary *service providers* may use profiles and tracking to enhance the functionality of their services. Among many others, social networks, content providers and distributors, service platforms, hosting services belong to this category.

Data collectors and traders monetize profiles directly. Identification also plays a strong role in *censorship and surveillance*, and–as discussed previously–also in *security*; for example, identification can help prevent click frauds by limiting the number of paid clicks per client per advertisement (Schmücker, 2011). On the other hand, *malicious parties*, such as identity thieves, phishers, and other kinds of online stalkers may also find profiles useful.

The Process of Profiling

In the early years of the Web, tracking was the only way to profile users. Attributes learned by tracking are called *implicit data*, since these are not expressed by the user, but deduced from her actions. In the early 2000s, with the spread of blogs, social networks, and so forth, and the social content contribution to the services of the Web 2.0 era, another source of information became available, called *explicit data*, referring to self-expressed information, collected by other means (e.g., using web crawlers) for extending profiles. Development of search engines catalyzed this process by supporting the access to public sources for explicit data, to the point where the spread of information became real time (e.g., Shankland, 2010). Instant accessibility and the emergence of a variety of "Web and content archiving" services (e.g., Fitzpatrick, 2012), had a negative effect on user privacy, as it simply eliminated the choice of revocation.

Classification of Information Sources

In their work, Gulyás et al. (2012) categorize profiling sources into three categories (complex profiling techniques may use multiple of these): services of information superpowers (large companies having a large service portfolio capable of capturing a significant fragment of user activity), public data sources, and tracking. Here, we further refine their classification by categorizing sources regarding the data types and the way users typically interact with these sources. (It must be added that exceptional, but less relevant cases do exist; for example, first party trackers can theoretically be realized, but the advertising business is based rather on third-party trackers.) Our classification is shown in Table 1.

According to this classification, a social networking site with a social widget in widespread use over the internet–such as Facebook's Like button–is classified as both using sources of an information superpower and tracking users. N.b. that the case of Facebook may be a bit more complex, as according to an earlier version of their privacy policy, the social service reserved the right to crawl public data sources in order to extend their users' profiles (Gulyás, 2009).

Table 1. Classification of profiling sources

	Implicit data	Explicit data
First party	Services of information superpowers	
Third party	Tracking	Public data sources

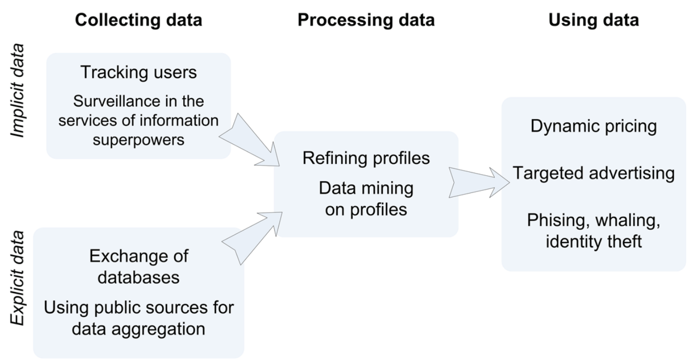
The Three Steps of Profiling

The implications of tracking become visible after understanding the context of identification and related economic activities (i.e., how profiling is performed). Identification itself does no harm to user privacy; the question is how identification is used. Implication of tracking techniques discussed later in this chapter only differ in the extent of identification (i.e., more generic identification implies larger profiles that has a wider range of use): while some techniques identify the user only in the same browser (e.g., tracking cookies, CSS-based history stealing attacks) or the same device (e.g., cross-browser fingerprinting, Flash PIEs), others achieve device-independent personal identification (e.g., real-world identification via history stealing, biometric fingerprinting).

The scheme of the process of personal data collection is depicted on Figure 1. The goal of the process is to create fine-grained user profiles to be used or sold. In the first step, profilers use the previously-discussed information sources for collecting information, the data from which is processed, combined and further refined in the second step. In step three, data can be monetized by several means.

During the collection process, numerous types of data are logged to profiles–we just highlight a few examples. Simply put, trackers can analyze

Figure 1. The process of collecting, processing and using different types of data



the visited Websites from different perspectives, such as regarding their content (for contextual advertising), their meaning (for semantic advertising), or the attitude of the creator towards the content itself (for sentiment analysis)–information that is rather static. That said, behavior profiles can also include clickstream information (i.e., how the user navigates through sites, to determine where she is off the track the advertiser planned, or to predict future behavior, e.g., to determine the willingness to order an item online)Van den Poel & Buckinx, 2005)). There are trackers that operate link shortening services and provide content sharing widgets in order to observe social sharing connections (Regalado, 2012). In our opinion, this trend will gain momentum, and advertisers will extend their view to collect even more metadata on daily routine, (social) behavior and connections.

What Do they Collect?

However, trackers also reach towards a lower metalevel. Jang et al. (2010) report several cases of mouse, keyboard, and clipboard tracking. Within the Alexa top 1,300 Websites, they found such tracking in 115 cases, where the tracking happened without any visual feedback to the user. Of these sites, 7 used the behavior tracking service of tynt.com, which reported mouseover and copy events, and also transferred the copied text to tynt.com. Jang et al. also mention another tracker company, ClickTale, who create an aggregate heatmap of mouse movements. According to ClickTale's website[1], in addition to heatmaps, they record mouse clicks, scroll reach, keystrokes, and offer reports based on the analysis of the data. Besides, Jang et al. note that many sites in their experiments use their own tracking techniques instead of using external services.

Service providers, and especially information superpowers, are in an even better position, as they can collect metainformation of the habits of their users. For instance, Google can be aware of their users' acquaintances (via Gmail and Google+), daily routine (with Google Calendar), interests (via Google Reader), and if Google Search is set as the home page of a browser, Google can estimate when the computer is first turned on, and how long it is operated. By utilizing geolocalization techniques, even home and work location can be determined, which, when coupled together, can be used for unique identification (Golle & Partridge, 2009).

Privacy Implications of Identification

When a user is identified and tracked, her activities are linked together in her profile, meaning a greater loss of user privacy for many reasons. For example, a profiled user can be influenced in her buying decisions, or her search results can be biased according to business objectives. A profiled user is tied to her past actions, which can be used for abuses, denigration, or may simply cause uncomfortable situations.

As an example, let us imagine a father looking for gifts for his daughter's birthday. After spending a few hours of searching online and visiting some related sites, he leaves his laptop on his desk. Later, his daughter arrives and uses his laptop to check out a website. We could imagine how surprised she would be to see the specific gift-related advertisements along the site. Clearly, the father would have liked to separate his gift-searching activities from regular ones.

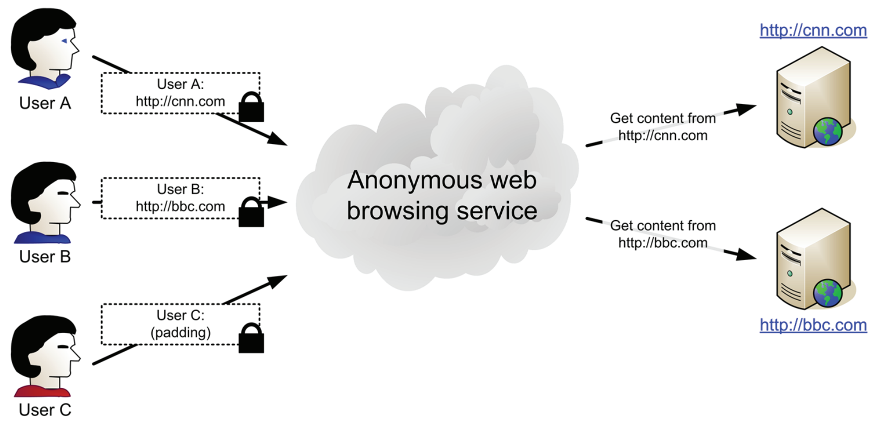
The goal would be to separate activities conducted on different sites, or–more formally–to reach the unobservability of the actions of the user (See Figure 2). Regarding a specific site visit, unobservability means anonymity towards this site (i.e., the visitor cannot be identified within the group of visitors), plus undetectability of the visit for both other sites and other Web actors (Pfitzmann & Hansen, 2010). At this time, these goals cannot be accomplished by using default Web browsing software (not even in private browsing mode; Aggarwal et al., 2010); some success, however, can be achieved by using anonymous Web browsers, such as Tor and JondoFox, which are reaching for such high aims (Perry et al., 2011).

Penetration of User Tracking Techniques

We differentiate between within- and cross-site trackers only; however, further refinement of classification is possible regarding the state-of-the-art tracking landscape (Roesner et al., 2012). In the case of the first type of trackers, the tracking cookie is owned by the visited site, and–since the tracker is never visited directly–the cookie is intentionally leaked for tracking (which is referred to as cookie handover). Cross-site trackers own the cookie themselves, enabling tracking user activities over site boundaries (the cookie can be set from a first- or third-party position, too).

However, both cooperation between trackers and combination of these types have been reported recently (Roesner et al., 2012). Cooperating trackers share tracking information–such as identifiers and location information upon a visit–which means that a tracker that is not present can learn about it. For instance, Roesner et al. mention the example of admeld.com leaking its tracking cookie and the

Figure 2. The scheme of undetectable web browsing: a request is unobservable for all actors, and anonymous towards communication partners (i.e., websites). Ideally, this means no actors (users and Websites) can decide whether a user made a request or not, and no Websites can specify which user made the request it received (responses are proxied through the anonymous Web browsing service).



top level page URL to turn.com. Cooperating trackers can seem to be misleadingly privacy-friendly, as for cooperating parties a single detector that invokes others later or sends feedback on detours can have a larger implication on privacy than it seems. Combination of tracker types implies that a within-site tracker gains cross-site tracker capability: the within-site tracker is embedded into the detector code of the cross-site tracker (the cookie is owned by the cross-site tracker). In this case, the cross-site tracker learns environmental information of the user via its tracker asset, but identifies the user via the within-site tracker.

Empirical measurements on tracking techniques indicate significant penetration. Gomez et al. (2009) analyzed 393,829 unique domains (collected by the users of the Ghostery Firefox plugin) of which 88.4% used Google Analytics (a within-site tracker), and top 50 Websites contained at least one tracker, but some had as many as 100. Ayenson et al. found Google Analytics to be present at 97 of the QuantCast.com top 100 sites in 2011. In their recent work, Roesner et al. (2012) also confirmed significant tracker presence while analyzing the Alexa top 500, as they found 7,264 instances of 524 trackers present on these sites. According to their work, "tracker morbidity" is depicted as the top 20 trackers being present in 26-297 sites in the top 500.

Trackers benefit from being visited as first-party sites as they are allowed to set cookies even when third-party cookies are blocked in the browser agent. Although cross-site trackers are usually not visited directly, there are two exceptions. First, social networking sites are visited voluntarily for their services, and as confirmed by the results of Roesner et al., some sites force visitors to view them from a first-party position (e.g., by opening a popup window or applying a redirection chain). Clearly, both are used to circumvent third-party cookie blocking settings of the browser agent, as in the case of insightexpressai.com, mentioned by Roesner et al.

Penetration of social widgets is also worth mentioning. In May 2011, the presence of Facebook, Google, and Twitter widgets was estimated at 33%, 25%, and 20% respectively among top 1,000 sites (Efrati, 2011), but surprisingly, their presence seems to stagnate. Roesner et al. measured the presence of both Facebook and Google at around 30% among the Alexa top 500 sites, and Twitter at 18.6%. In another recent work, Kontaxis et al. (2012) estimate the presence of the Facebook Like button at 35% among the top 10,000 sites (in June 2012). However, widgets also provide income for another group of services, namely companies hosting embeddable widget collections (e.g., AddThis), who fund their services from selling data obtained by tracking visitors (Mayer & Mitchell, 2012).

As revealed in this section, tracking is widely adopted for tracking users. It is not realistic to assume that these techniques are going to vanish in the near future; rather, on the contrary, we expect them to spread further and to develop new forms to walk one step ahead of consumers. In the next chapters, we review currently known and emerging techniques. We discuss storage-based tracking methods first, and "storageless" techniques–which are expected to dominate in near future–afterwards.

BASIS OF TRACKING: REGULAR TECHNIQUES

In the early years of the Internet, users were identifiable uniquely by their IPv4 addresses, but IP-based tracking soon became obsolete due to the widespread use of network address translation (NAT) and dynamic IP addresses, as in both cases multiple clients may use the same address. Despite its inaccuracy as an identifier, the IP address is still usable when augmented with other information (e.g., the user agent string), and can still play a part in tracking as part of a unique identifier (Yen et al., 2012). In addition, the IPv4 addresses are widely used for visitor localization. Free databases exist for pairing addresses with country-city locations, such as (IpToCountry, 2012), and research indicates that even a finer-grained localization is possible going down to street level (Wang et al., 2011). The role of the IP addresses may change in the future as IPv6 is expected to spread, since its addresses are practically capable of covering all existing and future devices, thereby eliminating the need for dynamic or translated addresses.

As IP addresses were not reliable anymore, Websites and trackers started to store unique identifiers on the visitors' computers, in the storage of their browsers, first in the cookie database. After a while, an increasingly relevant proportion of users started to delete cookies to opt out of tracking, and coevolution has characterized web privacy since then. When trackers develop new tracking techniques, related protection mechanisms follow. In this section, we review the classic storage-based techniques.

Storage-Based Techniques

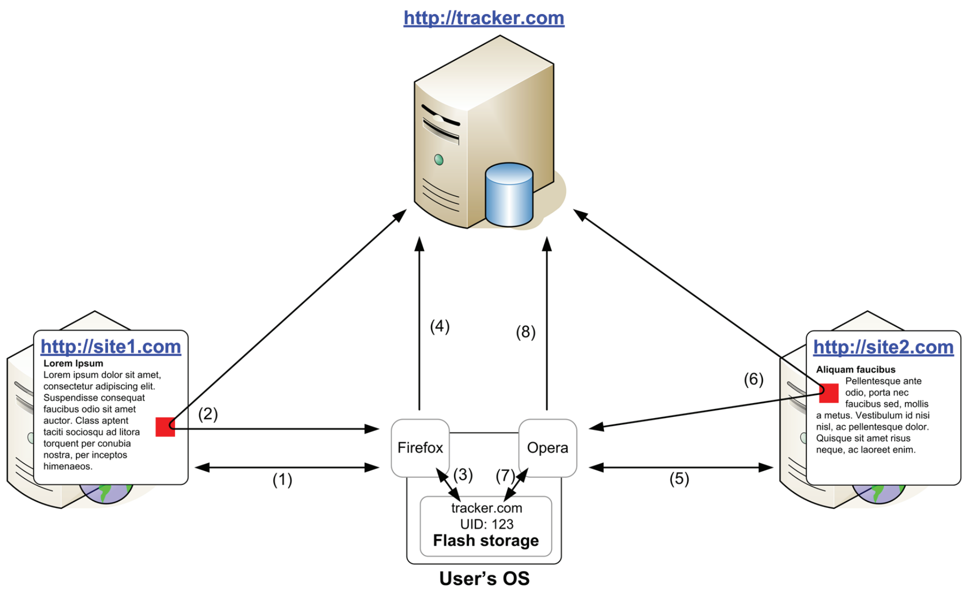
Cookies have some associated basic protection, such as the same-origin policy: if a Website sets a cookie, only the same site can read it later, allowing the identification of returning visitors, but not cross-site tracking. However, Website operators soon realized that if they installed *Web bugs*[2] on multiple sites, they could easily extend their reach. Web bugs are small, usually 1x1 pixel-sized invisible images, and based on a simple mechanism. When the user visits a site, and the content is loaded, the Web bug is downloaded from the third-party site (i.e., the tracker), who has a chance to read and write cookies, and can detect the first-party site address through the referrer HTTP header. Therefore, identification is possible, and the web bug owner can determine particular attributes of the visitor in addition to the IP address and browser agent information. Subsequent tracking techniques usually leverage the same principles with smaller modifications (e.g., using scripts or frames).

Problems related to cookies can be observed with other storage-based mechanisms, too. As we have mentioned previously, some trackers leak cookies intentionally, but malicious third parties may also try to steal cookies, for instance by having their scripts included into the site content, which seems to be common: Yue & Wang (2009) found that 66.4% of sites embedded external scripts in their experiment. In addition, user-contributed content can also contain cookie-stealing malicious scripts (e.g., cross-site scripting). Cookies are sent along with the request, and then, if they are being transmitted without protection, they can be subject to traffic sniffing[3] or active sidejacking[4]. Fortunately, in contrast to cookies, most stored objects are not transmitted automatically (e.g., Flash cookies and HTML5 storages), and therefore not affected by these threats.

As a relevant fraction of users soon started deleting cookies, trackers moved to using *rich internet application storages*, such as the Flash (Local Shared Objects) and SilverLight (Isolated Storage) storages. Both plugins are widespread (respectively 95.78%, 69.33% according to Stat Owl (2012), and their storages share some characteristics with cookies, with additional advantages (for trackers). Being isolated from the browser, these plugins allow the use of the same identifier in multiple browser agents in parallel (See Figure 3), and default expiration dates also favor tracking–perhaps larger cookie sizes, too. For Flash, cookie size starts from 100 KB (for requests having a larger size, the user is asked for permission), and permanent expiration is set by default (Local Shared Object, 2012). In the first years, lower awareness surrounded these techniques, but nowadays users are more aware, and there are standardized ways to control (and clear) their storages such as the NPAPI ClearSiteData (Chandna, 2011).

The first instance of announcing Flash cookies being used for tracking, were the Persistent Iden-

Figure 3. Cross-browser tracking with Flash storage. After contacting the first site (1) a Flash based Web bug is downloaded from the tracker (2). After the Web bug is loaded it sets (or reads if it exists) the identifier (3) and sends it back to the tracker directly (4), augmented with profiling information (e.g., subject of site1.com). When visiting another site with another browser, a similar process is executed (5-8), which uses the same storage.


http://tracker.com
http://site1.com
Lorem Ipsum
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse consequat faucibus odio sit amet auctor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.
http://site2.com
Aliquam faucibus
Pellentesque ante odio, porta nec faucibus sed, mollis a metus. Vestibulum id nisi nisl, ac pellentesque dolor. Quisque sit amet risus neque, ac laoreet enim.
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
Firefox
Opera
tracker.com
UID: 123
Flash storage
User's OS

tifier Elements (PIEs for short), which were announced in 2005 (Unitied Virtualities, 2005). Later, Flash cookies were reported to be used for "cookie respawning" (regenerating) cookies to circumvent cookie deletion and user preferences, and were found to be present on 54 of the top 100 sites (Soltani et al., 2009). Recent follow-up research shows that Flash cookies are still present on 37 of the top 100 sites (Ayenson et al., 2011), but a lower rate was confirmed by Roesner et al. (2012), as in their experiment only 35 of 524 trackers used Flash, and it was used as a backup only in 9 cases. The latter authors highlight the particular case of sodahead.com, where the Flash backup cookie was even encoded.

Ayenson et al. (2011) additionally report that trackers are also moving to *HTML5 storages*, as they found that 17 of the top 100 sites used them. The HTML5 storages allow larger storage (5 MB), and similarly to Flash permanent expiration is set as default. Roesner et al. (2012) find that only one percent of the Alexa top 500 stored identifiers in the HTML5 LocalStorage. However, they report two cases where LocalStorage was used instead of cookies, and a case where cookies were respawned from LocalStorage.

Different layers of the *browser cache* are also exploited for storing identifiers. A *cache control mechanism*, e-tags (or entity-tags) is used as backup for cookies, for instance, Ayenson et al. (2011) mention kissmetrics.com, and Wramner (2011) describes the use of e-tags in the tracking mechanisms of TradeDoubler. In their intended use, e-tags are used to determine if the online document has changed compared to the local cache, similarly to fingerprinting (hashing) documents for comparison. As a disadvantage (from the viewpoint of a tracker), e-tags are harder to handle, since reading and writing them requires exact URL matches. On the other hand, e-tags cannot be blocked with cookies, and they are even available in private browsing mode. Another control field, the last modified timestamp can also be exploited similarly to e-tags, simply by giving a unique date to each visitor.

Cached content can also be exploited for storing identifiers. The evercookie (2010) also uses this method: it draws the identifier on some pixels of an image, and then makes the browser cache it. In order to read the image, it is loaded onto an HTML5 canvas, where images can be managed on a pixel-level basis. Other content caches can also be used to store identifiers, such as variables in JavaScript, or entity properties in CSS. *Operational caches* are also available for tracking, such as the cache of HTTP authentication cache (Grossmann, 2007), the HTTP 301 redirect cache (Bursztein, 2011), the HTTP Strict Transport Security cache (Davidov, 2011), and the TLS session resumption cache, including TLS session IDs (Perry, 2011c).

There are even more storages exploited for tracking. The *window.name* property can store up to 2 MB; nevertheless, it starts from a clear value whenever a new window or tab is opened, and thus is cumbersome to use for tracking (evercookie, 2010). Last but not least, the *Internet Explorer userData* storage also emerged as an alternative to the Flash storage (Benninger, 2006), and can be used for tracking, as it provides a relatively large storage (64 KB per page, and in total 640 KB per domain), and not cleared when temporary files and cookies are removed.

It must be noted that many of the prior discussed techniques are often used in a combined manner, such as in the case of the evercookie (2010), replicating the identifier into as many places as possible. Trackers also prefer to use multiple storages at once; for example, Soltani (2011) reported that kissmetrics.com used Flash cookies, e-tags, and the HTML5 local storage to respawn deleted cookies.

Further Privacy Issues and Techniques Applied

The techniques discussed before can be applied to RSS (or Atom) channels; this is referred to as *RSS tracking*. In his recent thesis, Danis (2011) analyzed the possibilities of applying regular techniques to RSS channels, and found several flaws in the implementations in browsers and channel reader software. In our opinion, one of his most important finding is that by using internal frames (represented by the <iframe> HTML tag) security precautions (e.g., JavaScript blocking) can be bypassed easily, and, in addition, most reader software allows loading third-party images by default (with no options to block them), and also accepts cookies via these images.

Danis analyzed the channels of 40 Web pages selected from top listings, from which 11 displayed tracking advertisements, and reported two particular cases (nydailynews.com and microsoft.com), which used Web bugs to track their visitors, and collected browser, plugin, and OS information. Another major privacy problem of subscribing to RSS channels via browser agents is that the feed owner can observe the daily routine of subscribers, can determine workplace and home locations. This is also true in case of setting default home pages in browsers (especially in the case of services of information superpowers, such as Google search).

As another privacy threat, multiple sources of information can be used to enrich profiles. The browser itself provides information on several settings and variables which can be used for fingerprinting (e.g., the user agent string, plugin names and versions in an explicit list, screen resolution, and time zone), and plugins such as Java and Flash also leak identifying information on the operating system, hardware, or user settings, from which some–such as the OS information in Flash–cannot be spoofed (The Simple Computer, 2012). Plugins can also be used to evade the same-origin policy.

In addition, we highlight the example of URL referers, which are usually used to indicate the origin of the arriving user (in the form of an URL). However, the URL referer also indicates the URL of the host environment for images and other embeddable content (Flash, Java, internal frames), and can also indicate the keywords the user searched for before arriving on the Website. Lastly, we mention that all on-disk information (especially traces left on public computers) is a potential target to offline attacks via malware, and therefore should be protected carefully.

HISTORY STEALING AND COUNTERMEASURES

History stealing is a way of extracting a part of the history of the browser–information which is otherwise hidden from the prying eyes of a Website operator. In our broader interpretation of the term, it can refer to any way of determining if a site has been visited by the user. N.b. that none of these attacks can directly obtain the full browsing history; the adversary must choose a set of URLs to check before delivering the attack.

Leaking even the partial history can be dangerous; for instance, it can be enough to uniquely identify the user within the group of visitors of a Website. Of course, an attack can also take the more direct approach of attempting to determine if the user regularly browses controversial Websites, or collecting the list of service providers with which the user does business (e.g., in order to facilitate a subsequent social engineering attack). It must be noted that, while most history stealing attacks normally cannot do more direct damage to the user than that, some go as far as identifying her based on data obtained from a social networking Website (See Classical CSS-Based Attacks), or determining if she is a member of a Website where users must sign in to access certain features (See Cross-Site Timing-Based Attacks).

Early techniques of history stealing exploit a vulnerability that had been included in browsers since the adoption of Cascade Style Sheets (CSS) until the developers of Firefox patched the gaping security hole in the 4th version of their browser (See Classical CSS-based Attacks). During that period, various authors suggested different methods of acquiring the browser history, but the terminology has never been sufficiently uniform, leading to several different names for the same class of attacks.

The term "history stealing" is used in–besides nonacademic contexts such as blog posts–(Wondracek et al., 2010), while many authors prefer the use of "history sniffing" (Jakobsson & Stamm, 2006; Jang et al., 2010; Weinberg et al., 2011). Finally, others refer to these attacks simply as "history detection" (Janc & Olejnik, 2010a; 2010b), or "history leakage" (Wramner, 2011).

Attacks can be categorized along various properties. Most of them are scripted (e.g., implemented in JavaScript), though there are some markup language-based methods, too. Secondly, a nonscripted method can either require user interaction to complete the attack, or leak the browser history otherwise (e.g., through CSS). Thirdly, attacks vary in terms of accuracy (i.e., how certain their result is) and robustness (i.e., how much the set of identified elements of browsing history change over time). Fourthly, certain attacks only query domain names, while others work on the subdomain level. Fifthly, different algorithms perform differently in terms of the number of URLs that can be checked in a reasonable timeframe, and the CPU load they impose. Finally, other important factors include the repeatability and the generality of an attack.

In this section, we survey the history stealing techniques known to date, and categorize them based on the properties discussed previously, and also discuss possible protection mechanisms.

Timing-Based Attacks

Timing-based history stealing methods are timing attacks that infer browsing habits by querying the cache of the browser and that of the Domain Name

System or Service (DNS). The algorithms are based on the intuitive assumption that obtaining an object from a cache is significantly faster than getting it from a remote server.

The attack of Felten & Schneider (2000) on the browser cache makes use of a cacheable object (e.g., an image file from the main page of a news portal). The scripted version (Row L, Table 3) of the method measures the access time of the object, and decides that its hosting site has already been accessed if the retrieval takes longer than a predefined threshold. In the nonscripted version (Row K, Table 3), a dummy file is embedded into the attacker's page, followed by the test object, and then another dummy file; then, the access time of the test object can be inferred by inspecting the timestamps of retrieving the dummy objects in the log of the Web server. Both variants have an accuracy above 90%, as reported by the authors, based on their experiments. It is worth noting that the adversary may also use the cache for tracking purposes via "cache cookies" (Row M, Table 3).

The DNS-based (Row J, Table 3) attack is similar to the previous one, and has the exact same variants, but it is the time to execute a DNS query to a domain that is measured. Felten & Schneider argue that the attack is feasible if the cache miss penalty is significant for a specific domain name server; in such cases, the accuracy is above 90%.

It is uncertain if these attacks are still viable with modern computers. For instance, private browsing mode makes the browser wipe the cache when it terminates, thereby decreasing the accuracy of the browser cache-based attack. This is no longer a serious penalty for the users with high-speed, flat-rate internet access. In addition, we argue that the use of the first two attacks as a means of tracking a user or profiling her is not viable, as both of them are nonrepeatable (at least in theory). Cache cookies, on the other hand, may be efficient, provided that the cache is not wiped very often, but they require an adversary who can manipulate the expiry times of cached objects.

It must be noted that cache timing-based attacks (Rows N and O, Table 3) seem to be having their renaissance (mansour, 2011; Zalewski, 2011). These new algorithms work around the problem of nonrepeatability by aborting the loading of the cacheable resource after a predefined duration. If the object has finished loading during that timeframe, it is assumed to have been cached. We have not experimented much with these proofs-of-concept, but a short test with Microsoft Internet Explorer 9, Mozilla Firefox 12.0, and Google Chrome 20 on a moderately powerful laptop concluded that these algorithms might not be especially reliable, as they produced a large number of false negatives.

Cross-Site Timing-Based Attacks

It is also possible to mount a cross-site timing attack (Row P, Table 3) in order to infer history for a site where users must log in to access certain content. Such methods embed a test page and a reference page from a domain into the attacker's Webpage, and compare their loading times through the onerror and onload event handlers of JavaScript.

In the settings of Bortz et al. (2007), the reference page is chosen such that it displays similarly for a logged-in user and a guest, while the contents of the test page are significantly different between these groups. With certain Websites, it may also be possible to distinguish a logged-out member of the site from non-members through this technique.

Protecting users from cross-site timing attacks is not easy. A conceivable server-side countermeasure is to make each request on a Web server execute in a fixed time, while a client-side countermeasure could be to enforce the same-origin policy in JavaScript for the onload and onerror event handlers.

Classical CSS-Based Attacks

Several attacks rely on CSS, but the automated attacks in Janc & Olejnik (2010) are among the earliest and simplest ones. They exploit the way browsers handle the CSS pseudoclass of visited links–the feature that makes it possible to color (or otherwise format) an already visited hyperlink differently from one that is absent from the history.

The nonscripted, automated, CSS-based attack (Row A, Table 2) uses a style sheet that loads a unique background image for a hyperlink if it is in the history. When the victim loads the Web page, the browser makes a request for each URL that corresponds to a visited domain. The scripted variant of the attack (Row B, Table 2) uses the getComputedStyle function of JavaScript to access the style that was actually applied on an HTML element; then, the script can directly decide if the corresponding domain has been visited. Both attacks have been well-described at least since 2002 (Baron, 2002).

An attacker can also query addresses under the domain name of a social networking site (Row C, Table 2) with the goal of inferring the actual, real-world identity of the user (Wondracek et al., 2010). In order to pull this off, the adversary must have already crawled the social network; then, a CSS-based attack can be used to discover if the user visited the site of a predefined set of groups (i.e., online "clubs" with predefined topics to which interested users can subscribe). Assuming that a hit in this set equals an actual membership in the group, the set of hits can be matched against the previously crawled social network data. According to Wondracek et al., 42.06% of the users of the Xing social network could be uniquely identified by this method.

Privacy-preserving history mining (Jakobsson et al., 2008) was proposed to amend the attack with privacy-preserving features (Row D, Table 2), by leaking only predefined categories of visited Websites, instead of distinct domain names.

Table 2. Classification of query-based history stealing techniques

ID	Attack	Accuracy	Speed	Technology	Counter-measures	Harm
A	Pure CSS (Janc & Olejnik, 2010)	High	High	CSS	Baron's defenses; pollution; personalization; same-origin caching and styling	Partial history leakage
B	Scripted CSS (Janc & Olejnik, 2010)	High	High	CSS, JS	Baron's defenses; pollution; personalization; same-origin caching and styling	Partial history leakage
C	Social network HS (Wondracek et al., 2010)	High	High	CSS, JS	Baron's defenses; pollution; personalization; same-origin caching and styling	Complete identification (with real name)
D	Privacy-preserving history mining (Jacobsson et al., 2008)	Mode-rate	High	CSS, JS	opt-out; Baron's defenses; pollution; personalization; same-origin caching and styling	Coarse-grained information leakage about browsing habits
E	Word CAPTCHA (Weinberg et al., 2011)	High	Low	CSS	Unknown	Partial history leakage
F	Character CAPTCHA (Weinberg et al., 2011)	High	Low	CSS	Unknown	Partial history leakage
G	Pawn task (Weinberg et al., 2011)	High	Low	CSS	Unknown	Partial history leakage
H	Jigsaw puzzle (Weinberg et al., 2011)	High	Low	CSS	Unknown	Partial history leakage
I	Webcam attack (Weinberg et al., 2011)	Mode-rate	Low	Flash, image acquisition and processing	Unknown	Partial history leakage

Table 3. Classification of timing-based history stealing techniques

ID	Attack	Accuracy	Speed	Technology	Counter-measures	Harm
J	DNS timing (Felten & Schneider, 2000)	High	Moderate	DNS (JS)	Unknown	Partial history leakage
K	Naive cache timing (Felten & Schneider, 2000)	High	Moderate	N/A	Unknown	Partial history leakage
L	Scripted naive cache timing (Felten & Schneider, 2000)	High	Moderate	JS	Unknown	Partial history leakage
M	Cache cookie (Felten & Schneider, 2000)	High	Moderate	JS	Unknown	The cache cookie allows tracking of the user
N	Enhanced cache timing (mansour, 2011)	Low	High	JS	Unknown	Partial history leakage
O	Enhanced cache timing (Zalewski, 2011)	Low	High	JS	Unknown	Partial history leakage
P	Cross-site timing (Bortz et al., 2007)	De-pends on site	N/A	JS	Fixed-time requests; same-origin onerror and onload	Discovery of mem-bership in an online service

This algorithm is equal to a CSS-based attack, with hyperlinks to domains belonging to the same category behaving the same way, which makes them indistinguishable for the attacker.

Early proposed countermeasures against automated CSS-based history stealing attacks include URL personalization (Jakobsson & Stamm, 2006) and history pollution (Jakobsson & Stamm, 2007). The first proposition alters all URLs under the domain according to a user-specific pseudonym, thereby making guessing URLs impossible. The second one would insert entries into the history that point to sites that are of a similar nature to those actually visited by the user–this way, the attacker cannot distinguish between fake and real history entries. Other authors proposed the extension of the same-origin policy to caching and visited link differentiation (Jakobsson et al., 2006).

N.b. that these defenses did not make automated CSS-based history stealing completely infeasible. However, in Firefox 4, the vulnerability was thoroughly patched in April, 2010 by making CSS behave similarly for visited and unvisited links (Baron, 2002), and also by removing certain CSS features (Baron, 2010; Stamm, 2010).

Interactive CSS-Based Attacks

Modern browsers inhibit classical CSS-based attacks; nonetheless, style sheets can still be applied to hyperlinks. If the user can be tricked into disclosing the applied style of a set of elements, history stealing can still be feasible, albeit at a lower speed and for fewer URLs (Rows E through J, Table 2). Weinberg et al. (2011) detail four attacks, all of which are meant to be disguised as CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart; a puzzle that can easily be solved by a human, but is normally difficult for computer software). CAPTCHAs normally ask the user to, for example, recognize and type some slightly distortedly rendered letters, or to provide the right answer to a dynamically generated, simple question, such as “How much is 2 + 2?”

The first of the four attacks shows words to the user, while the second one displays specially constructed characters on seven-segment displays. The user is asked to type all words or characters into a text field, which completes the attack. The first attack infers one visited URL per word, while a character represents four for the second attack. The third attack draws a chessboard, and maps each square to a URL. Then, a chess piece is drawn into the square if the corresponding URL is visited; otherwise, the square is empty. The user is then asked to click all squares where she sees a chess piece. Finally, the fourth interactive attack is similar to a jigsaw puzzle: the user must click on the pieces of which a composite picture is made up. These attacks work, but at a very low URL detection speed; therefore, they are less viable for general profiling, but may be used in some targeted attacks (e.g., in phishing attacks).

Side-Channel CSS-Based Attacks

Miscellaneous methods include the webcam attack (Weinberg et al., 2011), which renders visited links with a blinking style, and recognizes this by processing images recorded with the user's Web camera (Row I, Table 2). Obtaining permission to use the Web camera may be problematic, but the required image processing algorithm is not very complex (but a simple, homogeneous background may be required behind the person using the computer).

Leaking History Through Security Policies

HTTP Strict Transport Security (HSTS) defines an HTTP header that forces the browser to initiate a secure HTTPS connection to a Website that was originally requested through plain HTTP. The browser stores this setting, so that subsequent requests are made via HTTPS. Previously we mentioned that this functionality can be exploited to store an identifier, but also to leak history (Davidov, 2011).

For storage purposes, an adversary can "burn in" a unique alphanumerical identifier into the browser as a set of HSTS policy entries–that is, through subdomains under the control of the attacker–and query it later on. Upon retrieval, the browser will initiate an HTTPS connection for the "right" domains only. This mechanism can be used similarly to determine visited links where such a setting was used. One can think of this algorithm as a mixture of a tracking cookie and history stealing. However, it is arguably more effective than that, since HSTS policy entries are meant to stay in the browser for a long time. A viable countermeasure would be to enforce a single policy for all subdomains.

Summary of History Stealing Attacks

In the empirical study of Jang et al. (2010) over the Alexa top 50,000 sites, it was shown that 485 sites inspected the style properties of elements that could leak browser history. Out of these sites, 46 confirmedly performed the classical, JavaScript-enhanced, CSS-based history stealing, and then transferred the result to some server; 36 of them used "off-the-shelf" history stealing code from third-party domains. Further 326 sites inspected a vast amount of domains, but Jang et al. could not confirm that the history information was sent to any server.

We have summarized the key features of all attacks discussed hitherto in Table 2 and Table 3; we have separated query-based and timing-based attacks (See our taxonomy) for better clarity. The following properties are listed for each attack:

Accuracy: High accuracy means few or zero false positives and false negatives; it can be seen that most attacks belong to this category. However, both the construction and accuracy of cross-site timing are highly dependent on the site to be attacked, and therefore a general result cannot be given for it. Moreover, privacy-preserving history mining is moderately accurate on purpose. The webcam attack is, however, moderately accurate per se, according to Weinberg et al. (2011). Other attacks by the same authors are highly reliable for a compliant user. Finally, the result of enhanced cache timing is based on our very limited experiments; in other setups, the attacks might perform better.

Speed. Most variants of classical, CSS-based history stealing can query hundreds of thousands of URLs per minute, while hundreds may be feasible with cache-based attacks (except the newer, enhanced ones). Obviously enough, attacks that require user interaction have the lowest speed. We have not been able to determine the speed of cross-site timing attacks, and we find the two sites tested by Bortz et al. too few to draw a conclusion.

Technology: Here we enumerate the technologies that are used by the attacks. We have not included self-evident ones, such as HTTP or TCP/IP.

Countermeasures: Here we enumerate the countermeasures that can be used to defeat (or at least cripple) an attack. Again, we have not included any self-evident methods (such as deleting the browsing history and/or the cache of the browser, disabling JavaScript in order to avoid JavaScript-based attacks, or covering the Web camera in order to defeat webcam attacks), so "unknown" does not always imply that the user is powerless; then again, many of these countermeasures do not allow a fine-grained tradeoff between browsing experience and privacy, so the implementation of better defenses might be desirable. By "Baron's defenses," we mean all countermeasures discussed by Baron (2010), and later implemented as a bug fix in (Baron, 2002), that is, making the style that is applied to hyperlinks inaccessible for JavaScript programs, modifying the way how CSS styles are applied to hyperlinks, and disabling certain CSS features.

Harm: In this column, we list the consequences of a successful execution of each attack, i.e., how detrimental it is to the privacy of the user.

FINGERPRINTING ON THE WEB

Fingerprinting is an emerging, storageless identification technique on the Web replacing storage-based techniques. When the client device is being fingerprinted, a reproducible, unique identifier is calculated, which can be easily recalculated with a high probability during a subsequent visit or when visiting another site, even if all client-side storages are cleared. Various information can serve as a basis of fingerprinting, such as the hardware parameters, unique features in network communication (e.g., the IP address or timing), or software settings and capabilities (such as OS brand, list of installed plugins, browser agent information). Regardless of the considered features, economically valuable fingerprinting techniques should not be sensitive for changes in the attributes taken into account during identification.

Although both history stealing and fingerprinting techniques are storageless, we argue that it makes sense to differentiate between them, as the prior are based on state (browsing history, caches, etc.), while the latter are setting- and attribute-based. The principal difference is that the client state can be influenced by the attacker and Websites (this is why client state can be exploited to operate as a storage, e.g., in the case of operational caches), but attributes and settings can only be changed by the user. This classification is further refined in a subsequent section.

Probably the earliest fingerprinting attempt used for identification was mentioned in the thesis of Mayer (2009), and the Panopticlick project (Eckersley, 2010) was the first empirical experiment on a large user base. There are even more earlier fingerprinting attempts like the passive OS fingerprinting of Miller (2002), or tools such as the browser identification tool, the Browserrecon (Ruef, 2008), which used HTTP request headers for identification; however, these rather focused on revealing real device and software attributes (i.e., discovering the OS type and the user agent string, respectively (also called OS and application fingerprinting)), instead of measuring uniqueness and large-scale identifiability. Although these are not applicable for tracking, they can be used for detecting client attributes, as an additional source for another type of fingerprinting.

Shortly after the Panopticlick project was published, companies started to switch from cookie-based to fingerprinting tracking techniques (Marshall, 2011), since fingerprinting works even in the case of privacy-conscious users who delete cookies regularly and use private mode, as in both cases, these efforts are useless against fingerprinting (Boda et Al., 2012). Although we must note that some companies, such as 41st Parameter, had used fingerprinting even before the thorough academic analysis began (Eckersley, 2010). In addition, the EU regulation on tracking cookies–often referred to as the "cookie law"–merely added fuel to the fire, since it prohibited the use of all kinds of storages for tracking unless the user consented (Loveless, 2011), and inspired advertisers to seek ways of circumvention.

Nowadays, several companies offer fingerprinting based behavioral tracking such as BlueCava Inc., Iovation Inc., 41st Parameter, claiming to provide device identification applicable to all kinds of devices; however, some trackers use fingerprinting only as a complementary solution to regular techniques, as is the case of TradeDoubler, where the device fingerprint is calculated as a hash of the user agent string and the IP address, and then used to track ad clicks (Wramner, 2011).

In this section, we review fingerprinting techniques, the related anonymity paradox (i.e., when forged information makes the subject even more outstanding from the crowd than as without), and possible defenses from the literature. We rigorously focus on technology-based solutions; however, there is another type of fingerprinting that may be incorporated into business practices in the future: biometry-based fingerprinting. There is scientific evidence that real-life behavior and human-computer interaction can be used for biometrical identification (Yampolskiy & Govindaraju, 2008), and it has already been shown that typing patterns are also personally identifying (Chairunnanda et al., 2011), even mouse movement fingerprinting (Feher et al., 2012). We believe that, in the near future, similar techniques, with a wider-scale use will emerge on the market of tracking.

Information-Based Fingerprinting Techniques

Information-based fingerprinting techniques query and collect high-level client attributes (read values of variables, constants, or measure certain characteristics) and settings for fingerprinting (e.g., from the OS or the browser). Here we review the most relevant information-based fingerprinting techniques, and the related countermeasures.

One of the earliest fingerprinting techniques, Browserrecon, targeted the family and version of the browser agent. Browserrecon regarded the user agent string to be fake, and instead it inspected the sent HTTP headers, as the header lines and their values differ for each browser family, sometimes even between versions, too. Norbert (2011) confirms this for Firefox, and in addition mentions a method of distinguishing Firefox from other browsers, as it is the only browser that makes a second request for the favicon if it is not found for the first time.

Occasionally, particular (or even identifying) information is sent through the headers. For example, an installation of Microsoft Office causes changes to the Internet Explorer HTTP accept headers (Wramner, 2011); or as another example, some mobile ISPs intentionally leak private information (such as mobile phone numbers, roaming status) of their subscribers through the request headers (Mulliner, 2010). However, headers need to be augmented with additional information for effective, wide-scale fingerprinting, in order for them to be suitable for tracking. In their work, Yen et al. (2012) analyzed millions of hosts who visited the Hotmail and the Bing search services, and concluded that 80% of users can be tracked by simply using the user agent strings augmented with IP prefix information as an identifier.

The thesis of Mayer (2009) was the first step towards classic information-based fingerprinting.

In his experiment, a test site was run, where the navigator object, screen resolution, list of plugins, and acceptable MIME types were hashed together for creating the browser fingerprint. Although only 1,328 clients were measured, this experiment showed the potential in fingerprinting, as unique fingerprints were obtained in 96.23% of all cases. It also turned out that, when plugin and MIME type lists are provided from Mozilla- and WebKit-based browsers, they provide additional entropy for fingerprinting. Eckersley showed that this is true for Flash- and Java-based font detection (Eckersley, 2010).

Making use of the underlying principles, the Panopticlick project was designed to test the uniqueness of browser fingerprints on a larger scale (i.e., to determine the commercial viability of such fingerprinting methods (Eckersley, 2010)). During the experiment, the uniqueness of different attributes such as the user agent string, screen resolution, font and plugin lists were measured, and self-information of combined attributes were also measured–but not the joint entropy of correlating attributes, as Perry et al. notice (2011). Until the beginning of the analysis, the project had gathered 286,777 fingerprints, from which 94.2% were unique if plugins were enabled (and only a further 4.8% had anonymity sets at least of two users), and Eckersley provided a simple but precise algorithm that can follow changes in fingerprints (i.e., due to browser or plugin updates) with an accuracy of 99.1%.

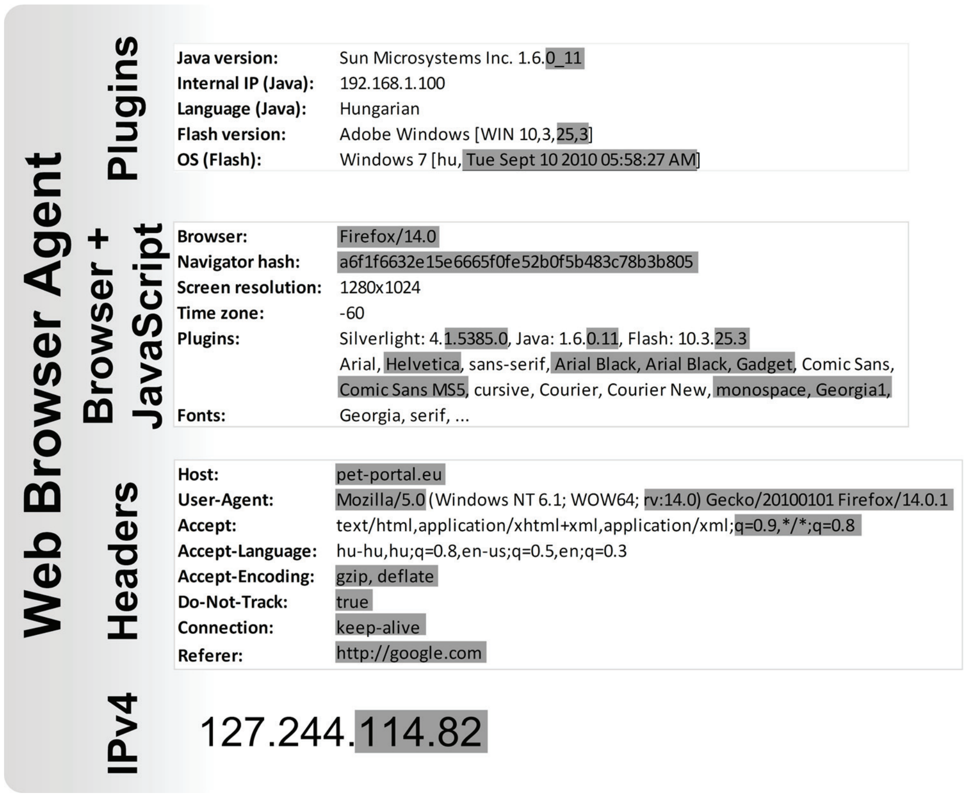
The Panopticlick project inspired the idea of cross-browser fingerprinting (Boda et Al., 2012), where two major improvements were made. Panopticlick used attributes that were browser-dependent (e.g., the user agent string), and it was also plugin-dependent, since precise results were provided only by using either Flash or Java to collect font lists (which is a rather important source of entropy). Although Eckersley (2010) mentioned CSS-based font enumeration, the work of Boda et al. (2012) is the first known attempt to implement JavaScript and CSS-based font detection for fingerprinting.

Moreover, the list of queried fonts was chosen as the basis of identification, based on the following assumption: as the list of installed software on a computer is unique, and so is the list of available fonts, since new applications can silently install fonts, too. Therefore, fingerprints were hashed from the first two octets of the IP address, the screen resolution, the time zone, and some selected fonts–where all attributes were browser-independent (See Figure 4). The font feature set was created in a way to eliminate values of uncertain and browser-dependent fonts, as some fonts behave differently in different browsers.

In a 6-month period of collection, a total of 989 fingerprints were obtained, on which it was shown that JavaScript-based font detection was sufficient for unique identification (at least for Windows and MacOS systems); however, due to the low number of fingerprints created in multiple browsers in parallel, the cross-browser property has remained a mere concept. This inspired the Cross-browser fingerprinting test 2.0 (2012), as an attempt to prove the viability of such tracking. The basic concepts remained the same, but there were some changes: the font feature list was refined (based on lessons learned from the first test), the first two IP octets were omitted, and an evercookie was optionally set to support future analysis (for which some additional info is also collected but not included in the fingerprint, such as the list of plugins).

There are several other fingerprinting experiments similar to Panopticlick and the cross-browser fingerprinting test, but we chose to omit them, as we deemed that they did not deliver groundbreaking novelties; however, there are fundamentally different ones that are worth mentioning. Whitelist fingerprinting (Mowery et al., 2011) is an attack against Firefox with the NoScript extension installed. NoScript allows users to completely block JavaScript, unless the site in question is included on a whitelist. The

Figure 4. Single vs. cross-browser fingerprint: values need to be filtered to obtain system- and not-browser-specific information as an input for the fingerprint


Web Browser Agent
Plugins
Browser + JavaScript
Headers
IPv4

Java version:	Sun Microsystems Inc. 1.6.0_11
Internal IP (Java):	192.168.1.100
Language (Java):	Hungarian
Flash version:	Adobe Windows [WIN 10,3,25,3]
OS (Flash):	Windows 7 [hu, Tue Sept 10 2010 05:58:27 AM]

Browser:	Firefox/14.0
Navigator hash:	a6f1f6632e15e6665f0fe52b0f5b483c78b3b805
Screen resolution:	1280x1024
Time zone:	-60
Plugins:	Silverlight: 4.1.5385.0, Java: 1.6.0.11, Flash: 10.3.25.3
Fonts:	Arial, Helvetica, sans-serif, Arial Black, Arial Black, Gadget, Comic Sans, Comic Sans MS5, cursive, Courier, Courier New, monospace, Georgia1, Georgia, serif, ...

Host:	pet-portal.eu
User-Agent:	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:	hu-hu,hu;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding:	gzip, deflate
Do-Not-Track:	true
Connection:	keep-alive
Referer:	http://google.com

127.244.114.82

rationale behind the algorithm is that the user is likely to whitelist sites of her interest that would not work with blocked JavaScript. The attack embeds a JavaScript program into a Web page and checks for hallmarks of a successful execution. For a large set of domains, the pattern of "whitelist hits" may be enough to fingerprint a user, but the efficiency of this fingerprinting scheme on a large user base is not known. Another problem of the attack is its small-scale usability.

Reschl et al. (2011) discuss a fingerprinting method that aims to discover the version of the browser through examining the behavior of the JavaScript implementation. Globally-available JavaScript test suites are used to discover the exact browser version. In a survey with 189 participants, the test suite ran in 90ms on an average PC, and 200ms on a smartphone, and identified supported browser versions with an accuracy of 100%–this claim was verified by inspecting the UAS of the browser and by asking the user about the exact version of the browser. Arguably, this information alone is little for unique identification, but may be used as an additional source in tracking methods. Norbert (2011) proposes to use somewhat rarely-used JavaScript calls like arguments.callee().toString() to discover subtle differences between the JavaScript implementations of browsers.

Mowery et al. (2011) constructed another browser and OS fingerprinting algorithm, but based it on off-the-shelf JavaScript test scripts (i.e., benchmarks). The accuracy of guessing the browser family was 98.2%, and guessing the correct browser version was successful in 79.8% of all cases. The operating system fingerprinting was performed within a chosen browser version, namely Firefox 3.6. Versions of Windows (7, Vista, and XP) could be distinguished with an accuracy of 98.5% (due to the lack of volunteers other systems were not measured).

Hardware and Network Level Fingerprinting Techniques

Besides browser and OS fingerprinting, Mowery et al. (2011) also discussed a CPU architecture fingerprinting algorithm that had a success rate of 45.3%. This is not very precise, and has an as long runtime as their OS and browser fingerprinting technique. That said, hardware fingerprinting is also possible by looking for minor differences between images rendered by different hardware (Mowery & Shacham, 2012) onto the <canvas> element of the HTML5 standard. Furthermore, versions of Safari return some quite detailed information about the Graphics Processing Unit (GPU) and the version of the rendering engine, which can further enhance a comprehensive fingerprint. Mowery & Shacham describe four different canvas-based algorithms. According to their empirical analysis on a sample of 300 volunteers, their test algorithms produced 116 groups, resulting in a distribution entropy of 5.73 bits.

In our opinion, this is quite formidable (albeit insufficient for creating unique identifiers per se), especially when combined with other fingerprints such as Panopticlick or the cross-browser test, and even more so if we consider that it runs in the fraction of a second, completely invisibly to the user, and how hard it is to implement countermeasures without imposing significant restrictions (e.g., completely disabling WebGL, adding noise to the rendered image, or rendering in software through a standardized graphics library) on the capabilities of the browser. Mowery & Shacham envision a pop-up window to ask for permission to retrieve pixel data as an effective defense, which may be viable, but it is uncertain how users would react to yet another permission dialog box.

Kohno et al. (2005) experimented with fingerprinting computers based on the clock skews they exhibit. Skew of two clocks is defined as the difference between the rates with which they advance. Two attacker models are discussed: one where the clock skew is calculated for TCP timestamps, and one where the attacker bombards the fingerprinted computer with ICMP Timestamp Requests, and estimates the skew of the system clock based on the values in the incoming ICMP Timestamp Replies. According to Kohno et al., the TCP timestamp-based fingerprint remains stable even if the attacked computer frequently synchronizes its system clock with a Network Time Protocol (NTP) time server. Furthermore, both methods yield similar results if the system clock is not synchronized with such a time server, as was found to be the default with many operating systems from 2005.

Kohno et al. conducted their experiments on traffic metadata obtained from an American Tier 1 ISP, and data obtained from tests with a computer laboratory with identically provisioned computers. Based on the first data set, the entropy of TCP timestamp-based clock skew fingerprinting was shown to range from 4.87 bits to 6.41 bits. The analysis of the second data set showed that such clock skews became stable shortly after bootup. Moreover, by testing their fingerprinting algorithm on the same laptop at different geographical locations and network access media, it was also proven that the clock skew estimate is oblivious to these factors.

Huang et al. (2012) discuss another clock skew-based device fingerprinting method with the aim of being an additional security layer for the login process of a cloud-based service (i.e., to recognize if a "usual" client is attempting to sign in to an account). The skew of the clock of a connecting client with respect to a server-side reference clock is measured by triggering an AJAX (Asynchronous JavaScript And XML) request every 5 seconds, and then the clock skew is estimated through statistical methods. According to their measurements, the so-estimated clock skew is largely independent from the network medium, and the only notable exception is the use of a virtual machine, which produces a different–but stable–clock skew upon every reboot. Furthermore, Huang et al. conducted an experiment with 100 devices, and found that the false negative and false positive rates are at most 8%.

However, we would like to highlight that the applicability of the algorithm for its original purpose of enhancing authentication is somewhat questionable, as at least 200 AJAX requests are required to get a meaningfully precise clock skew estimate, which takes 16.7 minutes. We argue, however, that the algorithm can successfully and clandestinely fingerprint a user if the attacker's Web page is kept open for a long time (e.g., when reading a news site or taking a coffee break, but not when logging in to a webmail service).

Yen et al. (2009) try to identify browser families based on other types of summarized traffic metadata of information flows. They examine 9 features of TCP connections (e.g., their byte count and duration). Firstly, the authors tested their classification method based on a data set comprising traffic from Windows-based hosts. The algorithm was tested by data originating from a single browser instance, and trained by the rest of the data set. It was found that the correct browser family could be identified with a precision of at least 71%, but even 100% was achievable after slight adjustments to the parameters. Then, the algorithm was also tested in a real-world-like scenario, where the precision of classifying Firefox and Opera browsers was calculated for several parameters, and it peaked at 74.56%.

In our opinion, this algorithm can probably be extended to several browser families (and possibly versions), as implied by the authors. However, requiring the summary of a vast amount of data imposes a lower bound on the capabilities of the attacker. As far as the possible countermeasures are concerned, it might be possible to make the browser "randomize" its traffic characteristics (e.g., varying the number of resources that are fetched in one go), but that could possibly introduce inconsistencies into the user experience.

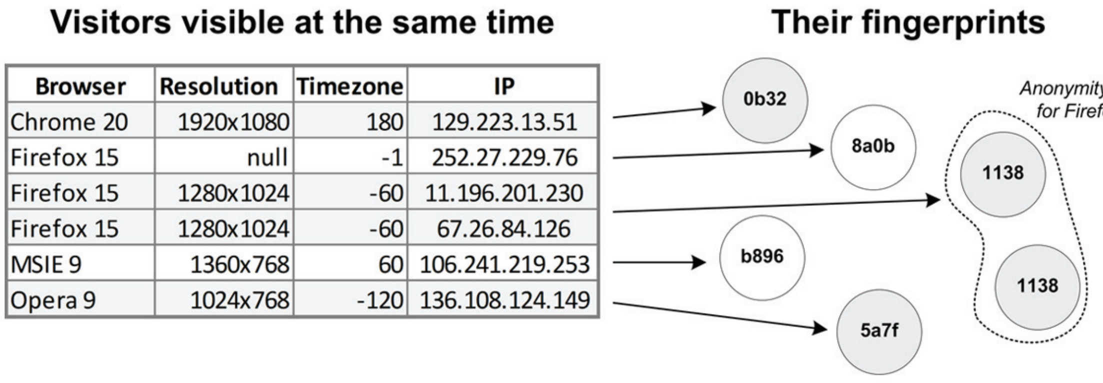
The Anonymity Paradox: Use Camouflage with Wisdom

Before discussing defenses against fingerprinting, we should mention the anonymity paradox (i.e., when one's efforts result in even stronger identifiability instead of preserving her privacy). Let us imagine that, as a means of protection, someone changes the user agent string of the browser to an empty string. This clearly preserves some privacy as it prevents information loss, but it is also likely that it uniquely identifies her (and makes her traceable), since such user agent strings are not very common. This is like putting on a ski mask in a bank; if you are the only one doing this, you are anonymous, but also in focus (and very likely to be caught). See Figure 5 for a fingerprinting-related example.

Eckersley (2010) brings up Privoxy users as examples, whose user agent strings had 15.5 bits of identifying information alone. Similarly, browsing the Web with Tor can also be a marker for fingerprinting, according to the measurements of Hubner et al. (2010), who found that only 22% of Tor users used TorButton to browse the Web. Perry et al. (2011) go even further and state that every single altered option can be used for fingerprinting; for example, user customized filters can be suitable targets, just as in the case of whitelist fingerprinting.

In connection to fingerprinting, such and similar phenomena–that is, when client state is altered in order to protect privacy, but due to the low user base doing the same it achieves identification rather than anonymity–are referred to in the literature as the Panopticlick or the fingerprinting paradox (The Simple Computer, 2012; Broenik, 2012) and also discussed in Eckersley (2010) and Perry et al. (2011). However, this concept can be generalized beyond fingerprinting, and can occur

Figure 5. The anonymity paradox illustrated: camouflaging is not enough, as anonymity set size also matters. There are three visitors who were using a counter fingerprinting technology, masking themselves as Firefox 15 users. However, one of the visitors used unusual camouflage settings which made her identifiable and trackable, but the others had an anonymity set size of 2.



Browser	Resolution	Timezone	IP
Chrome 20	1920x1080	180	129.223.13.51
Firefox 15	null	-1	252.27.229.76
Firefox 15	1280x1024	-60	11.196.201.230
Firefox 15	1280x1024	-60	67.26.84.126
MSIE 9	1360x768	60	106.241.219.253
Opera 9	1024x768	-120	136.108.124.149

in other scenarios while using privacy enhancing technologies (PETs), thus we call it the anonymity paradox.

Therefore, substitution values must be chosen carefully, or should be used by a sufficient number of people. Attributes should blend the device into the mass of others to achieve anonymity, and multiple attributes regarded together should be chosen carefully. As an example of the latter, an iOS Safari user agent string coupled with a resolution of 1280x1024 do not sound very realistic, and provides a good source of information for identification, too. In addition, one should consider the possibility of session linkability, and should not change values randomly too often (e.g., at every page load).

Defending Against Fingerprinting Attacks

One needs to consider many aspects in order to achieve unobservability of her actions against fingerprinting attacks; first of all, as a starting point it is presumed that no regular attacks will work against her (including IP-based tracking). Therefore, for the sake of simplicity (from the user's viewpoint), we propose the use of off-the-shelf solutions created by professionals, such as Tor Browser Bundle (Tor, 2012), or JonDoFox with JonDo Proxy (JonDo, 2012), as both provide network-level anonymization, and also protection against most regular tracking techniques and privacy-violating attacks. In addition, developers of both seek solutions to avoid fingerprinting attacks, too.

Particularly suspicious users may use static virtual machine snapshots (burnt to a DVD or reverted regularly) to protect their system against tracking (The Simple Computer, 2012), which also provides some protection against clock skew attacks as described previously, but, of course, this requires a sacrifice of some comfort. However, we note that in case of bad configuration, such systems provide a static fingerprint for tracking their user.

Customizing the Browser

For some reason, there are users who do not want to use complex solutions as previously discussed, but to build their own compilations instead (e.g., to add PETs as extensions to the Web browsing applications they use every day). There are many browser and system parameters to cover (e.g., time zone, user agent string, language and character set settings, accepted content types, headers), and there are some which cannot be masked without serious investigation and cumbersome hacking. For instance, information related to screen and content window resolutions need to be reduced, as alone this option is estimated to leak 29 bits of identifying information (Perry, 2011b), but a list of "talkative" attributes may be further convincing (e.g., JonDo Test, 2012).

In conclusion, in our opinion, building a custom software package seems futile, as a complete solution requires building an anonymous Web browser (Gulyás et al., 2008) with several extensions and modifications. We agree with Perry et al. (2011), who stated that "each option that detectably alters browser behavior can be used as a fingerprinting tool," and we just highlight whitelist fingerprinting again (Mowery et al., 2011) as an example.

Protection against JavaScript Engine Fingerprinting

Unfortunately, none of the discussed JavaScript-based fingerprinting algorithms are trivial to protect against (Norbert, 2011; Reschl et al., 2011). However, some attacks seem less threatening–even if we take the lack of effective countermeasures into consideration–as they are not feasible to be used in real-life situations. As a drawback of the fingerprinting suite of Mowery et al. (2011), its runtime of several minutes seems to be prohibitive, bar for Web pages that are normally left open for a long time (e.g., news sites). Mowery et al. argue that it might be possible to reduce the delay between the individual test scripts, which would result in a more universally usable algorithm.

We argue that functional aspects–such as which JavaScript test cases fail or execute in a certain timeframe–will always reveal the subtle differences between browser families and eventually–as bugs get fixed and JavaScript engines evolve–versions within a family, too. This is inevitable, unless all browsers include the exact same JavaScript interpreter; however, this would probably seriously impede competition between browser vendors and innovation.

One way of avoiding fingerprinting based on the characteristics of the JavaScript engine would be to allow the user to choose the implementation at will (e.g., with a browser extension). Of course, this is hardly viable between different browser families, and outright impossible for engines of closed-source browsers. Furthermore, this approach would possibly expose the user to security vulnerabilities of certain old JavaScript interpreters, or break the functionality of some Web pages that are optimized for a certain version of an engine.

Recent Developments on Protecting the Font List

The entropy of the font list was found to be 13.9 bits in the Panopticlick experiment, having the second highest value of all inspected attributes (Eckersley, 2010). Due to this finding, and the fact that fonts can be detected in multiple ways (via Flash, Java or JavaScript), fonts are likely to play an important role in fingerprint-based tracking. In addition, fonts also significantly influence the user experience of Web browsing, and therefore it is not possible to restrict the browser to a handful of fonts (or at least such a modification will not be accepted by the majority of users).

Firefox offers a simple option to restrict all possible fonts to a selection of four (i.e., Content > Fonts & Colors > Advanced > Allow pages to choose their own fonts), which also helps in impeding font-based fingerprinting, but it is not a very user experience-friendly solution. Therefore, it seems most likely that Firefox will apply a patch to this solution (Viecco, 2012), which was developed for the Tor browser (Perry, 2011a).

The latter includes two novel options, namely the browser.display.max_font_count and browser.display.max_font_attempts to limit the number of fonts, and font load attempts, regulated on a per page load basis (currently, test values are set as 5, 10 respectively). This method limits JavaScript-based detection only; therefore, affected plugins need to be disabled.

For the problem of session linkability, we suggest further modifications, as an attacker can test different fonts during sequential page loads. It is not likely that font lists collected this way would have a high entropy (at least for tracking only by font lists), but they can be regarded as a significant additional entropy source. Therefore, we propose to use these options in a per-domain setting, with a user interface for necessary interaction (e.g., clearing the cache of previously loaded fonts), which would not allow loading new fonts if the user navigates to a new page on the same domain.

The FireGloves add-on is a proof-of-concept utility (i.e., not a standalone extension providing enhanced privacy) aiming to show a compromise between good user experience and fingerprintability, thus it chooses a different approach to disable font-based fingerprintability–namely rewriting the offsetWidth and offsetHeight getters in order to prohibit JavaScript font detection (FireGloves, 2012). Although it has problems on some pages, it offers a usable alternative; however, the solution of the Tor Browser with a per-domain setting would admittedly embody a better solution.

Defending Low-Level Fingerprinting Attacks

Protecting users against the threats of these fingerprinting attacks is hard. Hardware-based techniques such as font rendering-based fingerprinting can hardly be defeated from a browser, as the fingerprintable information is provided by low-level software, such as the driver of a GPU, or a layer of the operating system. Clock skew is another example of a system feature that is out of the reach of the browser. Of course, certain features of the browser (such as the number of objects that are fetched in a single TCP session, or the content of HTTP request headers) can be manipulated, but it might impose some–possibly hard-to-predict–changes on user experience.

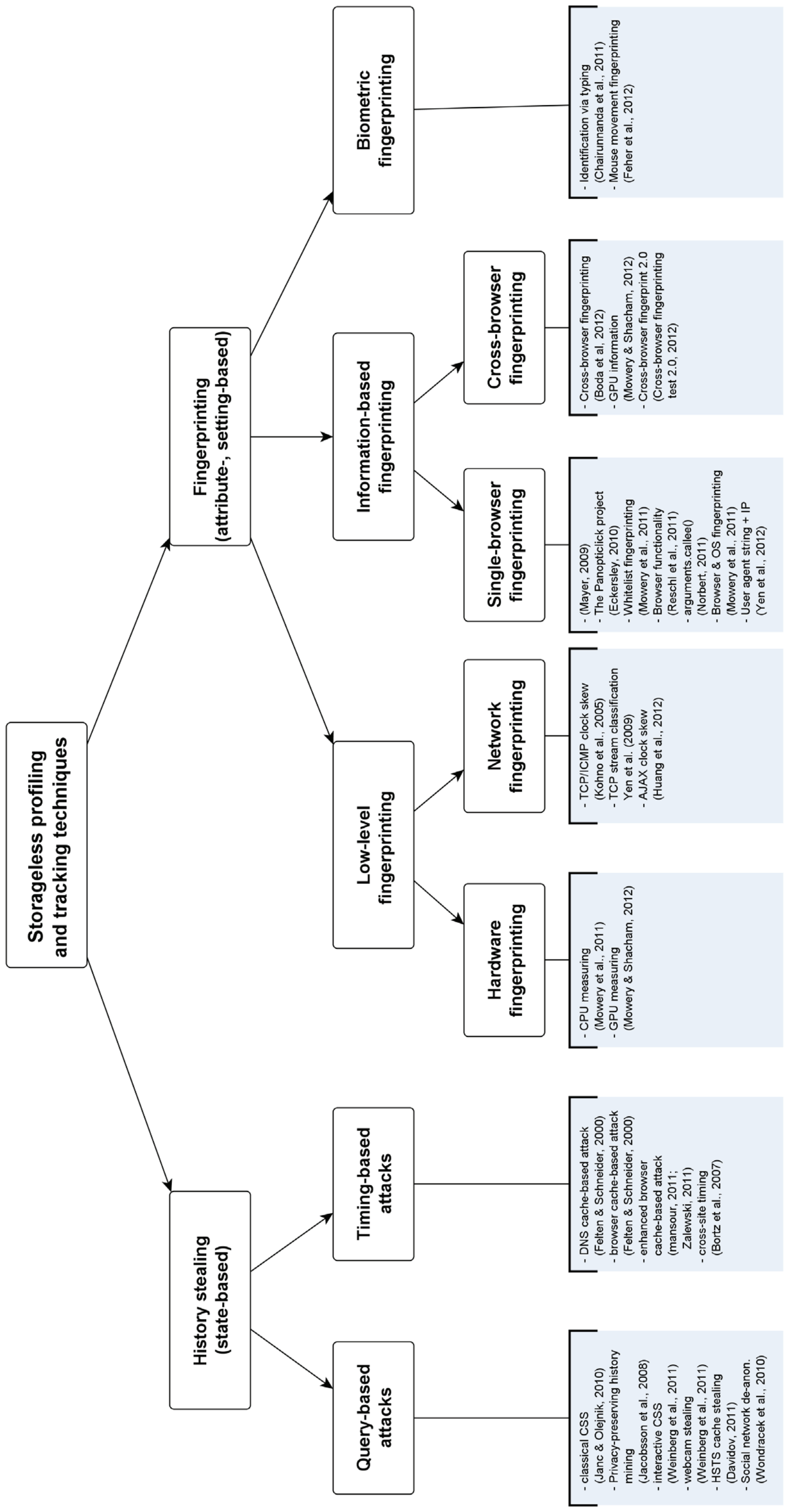
Taxonomy for Storageless Tracking Techniques

To the best of our knowledge, this is the first taxonomy provided for storageless tracking techniques. More precisely, we classify techniques using a single type of source at once; however, techniques using multiple sources may emerge in the future (e.g., hardware-level fingerprinting with browser functionality). Basically, there are three types of sources to take into account: state information (e.g., caches), attributes (e.g., screen resolution, font list, functionality testing) and settings (e.g., cookies enabled or not, whitelists).

In our taxonomy (depicted on Figure 6), first we differentiate history stealing from fingerprinting techniques by the fact that they are strictly state-based: history stealing attacks aim to extract state information from the client (i.e., from the user's device or software (browser, OS, etc.)). As the state of the client changes during browsing the Web, it can also be altered by the attacker in order to exploit it as a storage (e.g., HSTS cache). This is the same reason why it is not used for fingerprinting, as it can change frequently. Attacks discussed under history stealing can also be additionally categorized either query-based or timing-based attacks.

We furthermore divide fingerprinting into low-level and information-based fingerprinting subclasses. Exactly as discussed in this chapter, low-level fingerprinting sources incorporate hardware- and network-level fingerprinting techniques, and the rest of the tracking techniques constitute the group of information-based fingerprinting methods. In addition, we distinguish single-browser and cross-browser fingerprinting techniques and classify discussed procedures as on Figure 6. As an alternative, it is possible to distinguish information-

Figure 6. Taxonomy of the storageless tracking techniques


Storageless profiling and tracking techniques
Fingerprinting (attribute-, setting-based)
History stealing (state-based)
Biometric fingerprinting
Information-based fingerprinting
Low-level fingerprinting
Cross-browser fingerprinting
Single-browser fingerprinting
Network fingerprinting
Hardware fingerprinting
Timing-based attacks
Query-based attacks
- Identification via typing (Chairunnanda et al., 2011)
- Mouse movement fingerprinting (Feher et al., 2012)
- Cross-browser fingerprinting (Boda et al, 2012)
- GPU information (Mowery & Shacham, 2012)
- Cross-browser fingerprint 2.0 (Cross-browser fingerprinting test 2.0, 2012)
- (Mayer, 2009)
- The Panopticlick project (Eckersley, 2010)
- Whitelist fingerprinting (Mowery et al., 2011)
- Browser functionality (Reschl et al., 2011)
- arguments.callee() (Norbert, 2011)
- Browser & OS fingerprinting (Mowery et al., 2011)
- User agent string + IP (Yen et al., 2012)
- TCP/ICMP clock skew (Kohno et al., 2005)
- TCP stream classification Yen et al. (2009)
- AJAX clock skew (Huang et al., 2012)
- CPU measuring (Mowery et al., 2011)
- GPU measuring (Mowery & Shacham, 2012)
- DNS cache-based attack (Felten & Schneider, 2000)
- browser cache-based attack (Felten & Schneider, 2000)
- enhanced browser cache-based attack (mansour, 2011; Zalewski, 2011)
- cross-site timing (Bortz et al., 2007)
- classical CSS (Janc & Olejnik, 2010)
- Privacy-preserving history mining (Jacobsson et al., 2008)
- interactive CSS (Weinberg et al., 2011)
- webcam stealing (Weinberg et al., 2011)
- HSTS cache stealing (Davidov, 2011)
- Social network de-anon. (Wondracek et al., 2010)

based fingerprinting techniques as passive or active, based on whether they query client-side information (via JavaScript), or solely use information automatically revealed by the browser (Broenik, 2011; Mayer & Mitchell, 2012). In our opinion, browser-dependency is rather important that this attribute, and therefore we defined our taxonomy accordingly.

CONCLUSION: IS THERE ROOM FOR COOKIELESS TRACKING?

As we highlighted in the chapter discussing the current state of the advertising and tracking business, it is entirely clear that the industry will not abandon tracking in response to user countermeasures, but rather switch between technologies to bypass self-helping attempts of privacy-conscious users and also to avoid related regulations. Users are not as aware of fingerprinting technologies as cookie- and other storage-based techniques, and, consequently, this favors fingerprint tracking. There are even tracker companies who advertise their service as a solution for companies affected by the "cookie law" in the EU (BlueCava, 2012). Therefore, we expect that trackers will favor fingerprint-based tracking against other techniques.

However, going forward, regulation will hit fingerprint-related tracking, too (at least in the EU), but the industry will no doubt find a way to circumvent these regulations. For instance, it is not possible to forbid sites to query browser attributes entirely, since some are necessary for everyday operation. Even if these attributes are left out from the fingerprint measurements either because of regulation or technical barriers, passive fingerprinting can still be used for tracking. Therefore, we conclude that–regarding the current scenario–it is most likely that fingerprinting will spread among trackers, and in places having behavioral advertising and tracking regulations in effect, a longer battle will take place.

In the last few years, a kind of a cat-and-mouse game has taken place in the area of tracking techniques, especially remarkably for storage-based ones. When a novel tracking technique was discovered, in some time (after a sufficient rise in user awareness) a related protective solution arrived, which was shortly followed by a new identification method. This cycle seems to repeat endlessly, over and over. Fortunately, history stealing seems to have been defeated; due to strong industrial intervention, it is very unlikely for new, widely adoptable and effective techniques to emerge. Nevertheless, for fingerprinting techniques, the match has just begun.

This process indicates where research is needed for enhancing user privacy. As the target of tracking shifted from the browser to the device, we expect a similar shift to biometric fingerprinting (i.e., tracking the person herself across multiple browsers, OSes, and devices). Although fingerprinting countermeasures are far from being perfect (or widespread), and complex protective methods are yet examined by the community (e.g., protecting both against font-based cross-browser fingerprinting and clock-skew measurement), we believe it is time for researchers to start thinking about how to cover biometric information sources.

REFERENCES

Aggarwal, G., Bursztein, E., Jackson, C., & Boneh, D. (2010). An analysis of private browsing modes in modern browsers. In *Proceedings of the 19th USENIX Conference on Security* (6-6). Berkeley, CA: USENIX Association.

Ayenson, M., Wambach, D. J., Soltani, A., Good, N., & Hoofnagle, C. J. (2011). Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. *Social Science Research Network*. Retrieved November 13, 2012, from http://ssrn.com/abstract=1898390

Baron, D. (2002). Bug 147777 –:Visited support allows queries into global history. *Bugzilla@Mozilla*. Retrieved November 13, 2012, from https://bugzilla.mozilla.org/show_bug.cgi?id=147777

Baron, D. (2010). Preventing attacks on a user's history through CSS:Visited selectors. *David Baron's homepage*. Retrieved November 13, 2012, from http://dbaron.org/mozilla/visited-privacy

Benninger, C. (2006). AJAX Storage: A Look at Flash Cookies and Internet Explorer Persistence. *Technical report*. Retrieved November 13, 2012, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.128.2523

BlueCava. (2012). BlueCava releases "cookie-less" device identification technology for online advertisers. *Press release*. Retrieved November 13, 2012, from http://www.bluecava.com/news-release/bluecava-releases-cookie-less-device-identification-technology-for-online-advertisers/

Boda, K., Földes, Á. M., & Gulyás, G. Gy., & Imre S. (2012). User Tracking on the Web via Cross-Browser Fingerprinting. In P. Laud (Ed.), *Information Security Technology for Applications: 16th Nordic Conference on Secure IT Systems, NordSec 2011, Tallinn, Estonia, October 26-28, 2011, Revised Selected Papers* (pp. 31-46). Heidelberg, Germany: Springer-Verlag Berlin. Retrieved fromhttp://dx.doi.org/10.1007/978-3-642-29615-4_4

Bortz, A., Boneh, D., & Nandy, P. (2007). Exposing Private Information by Timing Web Applications. In *Proceedings of the 16th International Conference on World Wide Web* (pp. 621–628). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/1242572.1242656

Broenink, R. (2012). *Using Browser Properties for Fingerprinting Purposes*. Presented at the 16th Twente Student Conference on IT. Enschede, The Netherlands.

Buckinx, W., & Van den Poel, D. (2005). Predicting Online Purchasing Behavior. *European Journal of Operational Research*, *166*(2), 557–575. doi:10.1016/j.ejor.2004.04.022.

Bursztein, E. (2011, July 22). Tracking users that block cookies with a HTTP redirect. *Elie Bursztein's homepage*. Retrieved November 13, 2012, from http://elie.im/blog/security/tracking-users-that-block-cookies-with-a-http-redirect/

Chairunnanda, P., Pham, N., & Hengartner, U. (2011). *Privacy: Gone with the Typing! Identifying Web Users by Their Typing Patterns*. Presented at 4th Hot Topics in Privacy Enhancing Technologies, in conjunction with the 11th Privacy Enhancing Technologies Symposium. Waterloo, Canada.

Chandna, P. (2011). Adobe Releases Flash Player 10.3, Solves the 'Flash Problem'. *Maximum PC*. Retrieved November 13, 2012, from http://www.maximumpc.com/article/news/adobe_releases_flash_player_103_solves_flash_problem

Cranor, L. F. (2003). 'I Didn't Buy it for Myself ' Privacy and Ecommerce Personalization. *Institute for Software Research*. Retrieved November 13, 2012, from http://repository.cmu.edu/isr/37

Cross-browser fingerprinting test 2.0 (2012). *Cross-browser fingerprinting test 2.0*. Retrieved November 13, 2012, from http://fingerprint.pet-portal.eu/?lang=en

Danis, M. (2011). Analysis of user tracking techniques in web feeds. *Bachelor of Science Thesis*. Retrieved November 13, 2012, from http://pet-portal.eu/files/articles/2011/rss_tracking/rss_tracking.pdf (in Hungarian)

Davidov, M. (2011). The Double-Edged Sword of HSTS Persistence and Privacy. *Leviathan Security Group*. Retrieved November 13, 2012, from http://www.leviathansecurity.com/blog/archives/12-The-Double-Edged-Sword-of-HSTS-Persistence-and-Privacy.html

Davis, J. (2000). American consumers will force e-tailers to just say no to dynamic pricing. *Info-World*, *22*(41), 116.

Eckersley, P. (2010). How unique is your web browser? In M. J. Atallah, & N. J. Hopper (Eds.), *Privacy Enhancing Technologies: 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings* (pp. 1-18). Heidelberg: Springer-Verlag Berlin. doi:10.1007/978-3-642-14527-8_1

Efrati, A. (2011, May). 'Like' Button Follows Web Users. *The Wall Street Journal*. Retrieved November 13, 2012, from http://online.wsj.com/article/SB10001424052748704281504576329441432995616.html

Feher, C., Elovici, Y., Moskovitch, R., Rokach, L., & Schclar, A. (2012). User Identity Verification via Mouse Dynamics. *Journal Information Sciences: An International Journal*, *201*, 19–36. doi:10.1016/j.ins.2012.02.066.

Felten, E. W., & Schneider, M. A. (2000). Timing Attacks on Web Privacy. In *Proceedings of the 7th ACM conference on Computer and Communications Security* (pp. 25–32). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/352600.352606

FireGloves. (2012). FireGloves 1.2.3. *Add-ons for Firefox*. Retrieved November 13, 2012, from https://addons.mozilla.org/hu/firefox/addon/firegloves/

Fitzpatrick, A. (2012). 'Politwoops' Collects Politicians' Deleted Tweets. *Mashable*. Retrieved November 13, 2012, from http://mashable.com/2012/05/30/politwoops/

Goldfarb, A., & Tucker, C. E. (2011, January). Privacy Regulation and Online Advertising. *Management Science*, *57*(1), 57–71. doi:10.1287/mnsc.1100.1246.

Goldfarb, A., & Tucker, C. E. (2011). Online advertising, behavioral targeting, and privacy. *Communications of the ACM*, *54*(5), 25–27. doi:10.1145/1941487.1941498.

Golle, P., & Partridge, K. (2009). On the Anonymity of Home/Work Location Pairs. In H. Tokuda, M. Beigl, A. Friday, A. J. Brush, & Y. Tobe (Eds.), *Pervasive Computing*: *7th International Conference, Pervasive 2009, Nara, Japan, May 11-14, 2009. Proceedings* (pp. 390-397). Berlin, Germany: Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-01516-8_26.

Gomez, J., Pinnick, T., Soltani, A., Carver, B., Makker, S., & Mccans, M. (2009). *Know Privacy*. Technical report. Retrieved November 13, 2012, from http://knowprivacy.org/report/KnowPrivacy_Final_Report.pdf

Grossmann, J. (2007, April 20). *Tracking users with Basic Auth* [Web log comment]. Weblog. Retrieved from http://jeremiahgrossman.blogspot.hu/2007/04/tracking-users-without-cookies.html (2012, November 13 G.

Gulyás (2009). *Facebook – The Big Brother is watching you* [Web log comment]. Retrieved from http://pet-portal.eu/blog/read/174/ (2012, November 13)

Gulyás, G. Gy., Schulcz, R., & Imre, S. (2008). Comprehensive analysis of web privacy and anonymous web browsers: Are next generation services based on collaborative filtering? In L. Capra, I. Wakeman, N. Foukia, & S. Marsh (Eds.), *Proceedigns of the Joint SPACE and TIME Workshops 2008 (Part of IFIPTM 2008 - Joint iTrust and PST Conferences on Privacy, Trust Management and Security)* (pp. 17-32). Trondheim, Norway.

Gulyás, G. G., Schulcz, R., & Imre, S. (2012). Separating Private and Business Identities. In R. Sharman, S. Das Smith – M. Gupta (Eds.), Digital Identity and Access Management: Technologies and Frameworks (pp. 114-132). Hershey, PA: Information Science Reference. doi: doi:10.4018/978-1-61350-498-7.ch007.

Huang, D., Yang, K., Ni, C., Teng, W., Hsiang, T., & Lee, Y. (2012). Clock Skew Based Client Device Identification in Cloud Environments. In *Proceedings of 2012 IEEE 26th International Conference on Advanced Information Networking and Applications* (pp. 526–533). Fukuoka, Japan: IEEE. Retrieved from http://doi.ieeecomputersociety.org/10.1109/AINA.2012.51

Huber, M., Mulazzani, M., & Weippl, E. (2010). Tor http usage and information leakage. In B. De Decker, & I. Schaumüller-Bichl (Eds.), *Communications and Multimedia Security: 11th IFIP TC 6/TC 11 International Conference, CMS 2010, Linz, Austria, May 31 – June 2, 2010. Proceedings* (pp. 245-255). Berlin, Germany: Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-13241-4_22

Interactive Advertising Bureau (IAB). (2012, June 11). Internet Advertising Revenues Set First Quarter Record at $8.4 Billion. *Interactive Advertising Bureau Press Release*. Retrieved November 13, 2012, from http://www.iab.net/about_the_iab/recent_press_releases/press_release_archive/press_release/pr-061112

IpToCountry. (2012). FREE IP to Country Database. *WEBNet77*. Retrieved November 13, 2012, from http://software77.net/geo-ip/

Jackson, C., Bortz, A., Boneh, D., & Mitchell, J. C. (2006). Protecting Browser State from Web Privacy Attacks. In *Proceedings of the 15th International Conference on World Wide Web* (pp. 737–744). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/1135777.1135884

Jakobsson, M., Juels, A., & Ratkiewicz, J. (2008). Privacy-Preserving History Mining for Web Browsers. In *Proceedings of the Web 2.0 Security and Privacy 2008*. Washington, DC: IEEE Computer Society.

Jakobsson, M., & Stamm, S. (2006). Invasive Browser Sniffing and Countermeasures. In *Proceedings of the 15th International Conference on World Wide Web* (pp. 523–532). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/1135777.1135854.

Jakobsson, M., & Stamm, S. (2007). Web Camouflage: Protecting Your Clients from Browser-Sniffing Attacks. *Security and Privacy, IEEE*, *5*(6), 16–24. doi:10.1109/MSP.2007.182.

Janc, A., & Olejnik, L. (2010). Feasibility and Real-World Implications of Web Browser History Detection. In *Proceedings of the Web 2.0 Security and Privacy 2010*. Washington, DC: IEEE Computer Society.

Janc, A., & Olejnik, L. (2010). Web Browser History Detection as a Real-World Privacy Threat. In D. Gritzalis, B. Preneel, & M. Theoharidou (Eds.), *Computer Security – ESORICS 2010: 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings* (pp. 215–231). Berlin, Germany: Springer-Verlag.

Jang, D., Jhala, R., Lerner, S., & Shacham, H. (2010). An Empirical Study of Privacy-Violating Information Flows in JavaScript Web Applications. In *Proceedings of the 17th ACM conference on Computer and communications security* (pp. 270–283). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/1866307.1866339

JonDo. (2012). JonDoFox – Anonymous and secure web surfing. *JondoNym – The anonimisation service*. Retrieved November 13, 2012, from https://anonymous-proxy-servers.net/en/jondofox.html

JonDo Test. (2012). IP Check: Next generation of website tracking analysis. *JondoNym – The anonimisation service*. Retrieved November 13, 2012, from http://ip-check.info/description.php

Kamkar, S. (2010, September 20). Evercookie. *Samy Kamkar's homepage*. Retrieved November 13, 2012, from http://samy.pl/evercookie/

Kohno, T., Broido, A., & Claffy, K. C. (2005). Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, *2*(2), 93–108. doi:10.1109/TDSC.2005.26.

Kontaxis, G., Polychronakis, M., Keromytis, A. D., & Markatos, E. P. (2012). Privacy-Preserving Social Plugins. In *Proceedings of the 21st USENIX Conference on Security Symposium* (pp. 30-30). Berkeley, CA: USENIX Association.

Krane, D. (2008). Majority Uncomfortable with Websites Customizing Content Based Visitors Personal Profiles. *Harris Interactive press release*. Retrieved November 13, 2012, from http://www.harrisinteractive.com/vault/Harris-Interactive-Poll-Research-Majority-Uncomfortable-with-Websites-Customizing-C-2008-04.pdf

Krishnamurthy, B. (2010). Privacy leakage on the Internet. Presented at 77th Internet Engineering Task Force. Anaheim, CA.

Krishnamurthy, B., & Wills, C. E. (2006). Generating a privacy footprint on the Internet. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (pp. 65-70). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/1177080.1177088.

Krishnamurthy, B., & Wills, C. E. (2009). Privacy diffusion on the web: A longitudinal perspective. In *Proceedings of the 18th International Conference on World Wide Web* (pp. 541-550). New York: ACM Press. Retrieved from http://doi.acm.org/10.1145/1526709.1526782

Local Shared Objects. (2012). Local shared object. *Wikipedia*. Retrieved November 13, 2012, from http://en.wikipedia.org/wiki/Local_shared_object

Loveless, A. (2011). EU Cookie Law – It ain't all about cookies (or browsers). *Enchilada*. Retrieved November 13, 2012, from http://www.enchiladadigital.com/2011/07/24/eu-cookie-law-cookies-browsers/

Mansour. (2011). visipisi. Retrieved November 13, 2012, from http://oxplot.github.com/visipisi/visipisi.html

Marhsall, J. (2011). Device Fingerprinting Could Be Cookie Killer. *ClickZ – Marketing News & Expert Advice*. Retrieved November 13, 2012, from http://www.clickz.com/clickz/news/2030243/device-fingerprinting-cookie-killer

Mayer, J. (2009). *"Any person... a pamphleteer" Internet Anonymity in the Age of Web 2.0*. Unpublished thesis. Retrieved November 13, 2012, from http://www.stanford.edu/~jmayer/papers/thesis09.pdf

Mayer, J. R., & Mitchell, J. C. (2012). Third-Party Web Tracking: Policy and Technology. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (pp. 413-427). Washington, DC: IEEE Computer Society. Retrieved fromhttp://dx.doi.org/10.1109/SP.2012.47

McDonald, A. M., & Cranor, L. F. (2010). Beliefs and behaviors: Internet users' understanding of behavioral advertising. In *Proceedings of the 38th Research Conference on Communication, Information and Internet Policy*. Arlington, VA: TPRC.

Miller, T. (2002). Passive OS Fingerprinting: Details and Techniques. *The SANS institute*. Retrieved November 13, 2012, from http://www.ouah.org/incosfingerp.htm

Mowery, K., Bogenreif, D., Yilek, S., & Shacham, H. (2011). Fingerprinting Information in JavaScript Implementations. In *Proceedings of the Web 2.0 Security and Privacy 2011*. Washington, DC: IEEE Computer Society.

Mowery, K., & Shacham, H. (2012). Pixel Perfect: Fingerprinting Canvas in HTML5. In *Proceedings of the Web 2.0 Security and Privacy 2012*. Washington, DC: IEEE Computer Society.

Mulliner, C. (2010). Random tales from a mobile phone hacker. Presented at CanSecWest 2010. Vancouver, WA.

Norbert. (2011). Web browser fingerprinting. *SecurityAnaly.st Blog*. Retrieved from http://securityanaly.st/browser-fingerprinting/

Oracle. (2011). Oracle Adaptive Access Manager. *Oracle Fusion Middleware*. Retrieved November 13, 2012, from http://www.oracle.com/technetwork/middleware/id-mgmt/oaam11gr1ps1-ds-398161.pdf

Perry, M. (April 10, 2011). Torbutton Design Documentation. *Technical report.* Retrieved November 13, 2012, from https://www.torproject.org/torbutton/en/design/#fingerprinting

Perry, M. (2011, April 9). Ticket #2872: Limit the fonts available in TorBrowser. *Tor Bug Tracker & Wiki*. Retrieved November 13, 2012, from https://trac.torproject.org/projects/tor/ticket/2872

Perry, M. (2011, September 26). Ticket #4099: Disable TLS Session resumption and Session IDs. *Tor Bug Tracker & Wiki*. Retrieved November 13, 2012, from https://trac.torproject.org/projects/tor/ticket/4099

Perry, M., Clark, E., & Murdoch, S. (2011, December 28). *The Design and Implementation of the Tor Browser*. Technical report. Retrieved November 13, 2012, from https://www.torproject.org/projects/torbrowser/design/

Pfitzmann, A., & Hansen, M. (2010). A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. *Privacy and Data Security, TU Dresden, Faculty of Computer Science, Institute of Systems Architecture.* Retrieved November 13, 2012, from http://dud.inf.tu-dresden.de/Anon_Terminology.shtml

Regalado, A. (2012, June 22). Online Ads That Know Who You Know. *MIT Technology Review*. Retrieved November 13, 2012, from http://www.technologyreview.com/news/428048/online-ads-that-know-who-you-know/

Reschl, P., Mulazzani, M., Huber, M., & Weippl, E. (2011). Poster Abstract: Efficient Browser Identification with JavaScript Engine Fingerprinting. In *Proceedings of the Annual Computer Security Applications Conference 2011 (ACSAC).* Orlando, FL: ACM.

Roesner, F., Kohno, T., & Wetherall, D. (2012). Detecting and Defending Against Third-Party Tracking on the Web. In *Proceedings of 9th USENIX Symposium on Networked Systems Design and Implementation* (pp. 12-12). Berkeley, CA: USENIX Association.

Ruef, M. (2008). Browserrecon. *Browserrecon project – Advanced web browser fingerprinting*. Retrieved November 13, 2012, from http://www.computec.ch/projekte/browserrecon/

Schmücker, N. (2011). *Web Tracking*. Technical report. Retrieved November 13, 2012, from http://www.snet.tu-berlin.de/fileadmin/fg220/courses/SS11/snet-project/web-tracking_schmuecker.pdf

Shankland, S. (2010). Google real-time search: 6 min. to spot quake. *Deep Tech – CNET News*. Retrieved November 13, 2012, from http://news.cnet.com/8301-30685_3-10428590-264.html

Simpson, G. H. (2005, March 31). United Virtualities develops ID backup to cookies, Browser-Based 'Persistent Identification Element' will also restore erased cookie. *Whiptech Technologies*. Retrieved November 13, 2012, from http://www.whiptech.com/services.security/PIE.and.Cookies.html

Soltani, A. (2011) Respawn redux (Follow up to Flash Cookies and Privacy II). *Ashkan Soltani's homepage*. Retrieved November 13, 2012, from http://ashkansoltani.org/docs/respawn_redux.html

Soltani, A., Canty, S., Mayo, Q., Thomas, L., & Hoofnagle, C. J. (2009, August 10). Flash Cookies and Privacy. *Social Science Research Network*. Retrieved November 13, 2012, from http://ssrn.com/abstract=1446862

Stamm, S. (2010). Plugging the CSS History Leak. *Mozilla Security Blog*. Retrieved November 13, 2012, from https://blog.mozilla.org/security/2010/03/31/plugging-the-css-history-leak/

Stat Owl. (2012). Rich Internet Application Market Share. Retrieved November 13, 2012, from http://www.statowl.com/custom_ria_market_penetration.php

The Simple Computer. (2012, June 10). Coping Mechanisms: Fingerprinting, CDI and How to Deal with it. *the_simple_computer, digital security and privacy made easy*. Retrieved November 13, 2012, from http://thesimplecomputer.info/fingerprinting-and-clientless-device-identification.html

ThreatMetrix. (2012). ThreatMetrix™ Cybercrime Defender Platform Datasheet. *ThreatMetrix | Advanced Cybercrime Prevention Solutions*. Retrieved November 13, 2012, from http://www.threatmetrix.com/docs/ThreatMetrix-Cybercrime-Defender-Platform.pdf

Tor. (2012). Tor Browser Bundle. *Tor Project*. Retrieved November 13, 2012, from https://www.torproject.org/projects/torbrowser.html.en

TRUSTe. (2009). 2009 Study: Consumer Attitudes About Behavioral Targeting. *Research report*. Retrieved November 13, 2012, from http://www.truste.com/pdf/TRUSTe_TNS_2009_BT_Study_Summary.pdf

TRUSTe & Harris Interactive. (2011). *Privacy and online behavioral advertising*. Retrieved November 13, 2012, from http://truste.com/ad-privacy/TRUSTe-2011-Consumer-Behavioral-Advertising-Survey-Results.pdf

Turow, J., King, J., Hoofnagle, C. J., Bleakley, A., & Hennessy, M. (2009). Americans reject tailored advertising and three activities that enable it. *Social Science Research Network*. Retrieved November 13, 2012, from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1478214

Viecco, C. (2012). Bug 732096 – Add a preference to prevent local font enumeration. *Bugzilla@Mozilla*. Retrieved November 13, 2012, from https://bugzilla.mozilla.org/show_bug.cgi?id=732096

Wang, Y., Burgener, D., Flores, M., Kuzmanovic, A., & Huang, C. (2011). Towards Street-Level Client-Independent IP Geolocation. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation* (pp. 27-41). Berkeley, CA: USENIX Association.

Weinberg, Z., Chen, E. Y., Jayaraman, P. R., & Jackson, C. (2011). I Still Know What You Visited Last Summer: Leaking Browsing History via User Interaction and Side Channel Attacks. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy* (pp. 147–161). Washington, DC: IEEE Computer Society. Retrieved fromhttp://dx.doi.org/10.1109/SP.2011.23

Wondracek, G., Holz, T., Kirda, E., & Kruegel, C. (2010). A Practical Attack to De-Anonymize Social Network Users. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (pp. 223–238). Washington, DC: IEEE Computer Society. doi: 10.1109/SP.2010.21.

Wramner, H. (2011). *Tracking Users on the World Wide Web*.Unpublished Master's thesis. Retrieved November 13, 2012, from http://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2011/rapporter11/wramner_henrik_11041.pdf

Yampolskiy, R. V., & Govindaraju, V. (2008). Behavioural biometrics: A survey and classification. *International Journal of Biometrics*, *1*(1), 81–113. doi:10.1504/IJBM.2008.018665.

Yen, T., Huang, X., Monrose, F., & Reiter, M. K. (2009). Browser Fingerprinting from Coarse Traffic Summaries: Techniques and Implications. In U. Flegel, & D. Bruschi (Eds.), *Detection of Intrusions and Malware, and Vulnerability Assessment: 6th International Conference, DIMVA 2009, Como, Italy, July 9-10, 2009. Proceedings* (pp. 157–175). Berlin, Germany: Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-02918-9_10

Yue, C., & Wang, H. (2009). Characterizing Insecure JavaScript Practices on the Web. In *Proceedings of the 18th International Conference on World wide Web* (pp. 961-970). New York: ACM Press. doi:10.1.1.153.3166.

Zalewski, M. (2011). Rapid history extraction through non-destructive cache timing (v8). *[lcamtuf.coredump.cx]*. Retrieved November 13, 2012, from http://lcamtuf.coredump.cx/cachetime/

KEY TERMS AND DEFINITIONS

Anonymity Paradox: The problem of increasing identifiability when concealing (private) information. In other words, a user is likely to end up in a rather small anonymity set if the substitute (fake) information is not chosen carefully.

Behavioral Tracking and Advertising: A way of observing and analyzing the user's interaction with one or more Websites, rather than simply registering Webpage downloads. The collected information may include the contents of the clipboard, mouse heatmaps, keystrokes, scroll reach, and so forth, which is later used for behavioral advertising (i.e., when advertisements are tailored to predicted user behavior).

Fingerprint: Set of attributes or settings that is uniquely identifying for a selected entity within a set of others (e.g., browser/device of a visitor among other visitors).

History Stealing: Also referred to as history leakage, history detection or history sniffing, history stealing is a way of extracting a part of the history of the browser (or other state-descriptive information) without the consent of the user. Such an attack normally cannot acquire the entire browsing history; the adversary must choose a set of addresses to check before delivering the attack.

Identification, Tracking, and Profiling: In the context of the Web, these terms refer to the act of a website uniquely identifying visitors, tracking theirs actions in order to create profiles on them. These profiles are later monetized; for example, by showing advertisements tailored for the profile, and therefore to the preferences of the visitor.

Implicit and Explicit Data: Explicit data refers to some piece of information communicated by the client (e.g., user agent string) or published on the Web, while implicit data refers to information that can be extrapolated from the actions, behavior, status, attributes, and so forth, of the client or from other explicit data.

Storage-Based Tracking: Regular tracking techniques store an identifier on the client (e.g., as browser cookies) in order to be able to identify returning visitors and monitor their activities.

ENDNOTES

1 http://www.clicktale.com

2 Also called Web beacons, clear GIFs, 1x1 GIFs, tracking pixels, pixel tags or simply pixels.

3 Firesheep raised awareness on Facebook session cookies transmitted without protection: http://codebutler.github.com/firesheep/

4 Stealing Gmail session cookies via active sidejacking: http://seclists.org/bugtraq/2007/Aug/70