Doppelganger: Better Browser Privacy Without the Bother

Umesh Shankar
ushankar@cs.berkeley.edu
UC Berkeley

Chris Karlof
ckarlof@cs.berkeley.edu
UC Berkeley

ABSTRACT

We introduce Doppelganger, a novel system for creating and enforcing fine-grained, privacy preserving browser cookie policies with low manual effort. Browser cookies pose privacy risks, since they can be used to track users' actions in detail, but some cookies also enable useful functionality, like personalization features. Web browsers currently lack an effective cookie management mechanism. Users must choose between two unpalatable options: a permissive, privacy-compromising policy for every site they visit, or a seemingly endless series of questions to which they must supply underinformed opinions. Doppelganger takes a big step forward: it makes automated determinations of cookies' value to enable a cost-benefit analysis, and offers an automated recovery system when that mechanism—or the user—makes an incorrect judgment. Doppelganger leverages client-side parallelism to automatically and simultaneously explore multiple cookie policies, enabling each user to create her ideal cookie policy. We tackle important and difficult subproblems along the way: mechanisms for recording and replaying web sessions; improved handling of third-party cookies; and enforcing fine-grained, per-site cookie mediation. We implemented Doppelganger as a Firefox extension; we discuss experimental results comparing it to various browser settings, as well as lessons learned from the real-world engineering challenges we faced in our implementation.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communication Applications – Information Browsers

General Terms

Security, Human Factors

Keywords

Cookies, Web Privacy, Usable Security

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.
Copyright 2006 ACM 1-59593-518-5/06/0010 ...$5.00.

1. INTRODUCTION

1.1 Background

An *HTTP cookie* (from here on, simply "cookie") is a small data item sent by a web site to a web browser, then sent back to the originating site on subsequent requests. While the original intent of cookies was to provide a session state mechanism for the stateless HTTP protocol, cookies have since been used not just for things like shopping carts and authentication, but also for tracking users' web surfing habits and building targeted advertising profiles. The result is that site operators or third parties can gain undesirable insight into users' habits and browsing history. Cookies can identify a user at sites where she believes herself to be anonymous and track her actions across sites and browsing sessions. The problem is exacerbated if web sites can correlate the collected data to users' real-world identities.

The difficulty, therefore, is in deciding which cookies are worth accepting and which are not. Ideally, a user should be able to compare the privacy cost of a cookie with the functionality benefit the cookie enables. Most users are not equipped to make these decisions manually and accept the global defaults in their browsers, which often apply a single policy to all sites. These defaults tend to err on the side of functionality rather than privacy. Since web site features such as shopping carts and logins often require cookies and users may become confused or annoyed if these features don't work, the default policies liberally accept cookies. Although this approach will minimize users' frustration, it will also accept many useless tracking cookies which unnecessarily violate users' privacy. Our goal is to get the best of all worlds: a cookie policy that protects users' privacy while simultaneously retaining the desired functionality and, perhaps most importantly, not pestering users so much that they disable the system.

1.2 A solution: Doppelganger

We introduce Doppelganger, a web browser privacy tool to help each user formulate her ideal cookie policy. Doppelganger is a system that, in effect, simulates a world in which the user has accepted cookies and compares it to the (default) world in which the user has not. If there is no change in the user's experience between the two worlds, then we can fairly say that the cookies are not useful. Thus, Doppelganger essentially creates a hidden twin of the user who is constantly exploring the value of cookies on the sites the user browses and who informs the user when accepting cookies may be a good tradeoff; useless cookies are rejected by default, to preserve privacy. Another key component of Doppelganger is an automated error recovery module, which users may invoke with a single click. Error recovery attempts not only to correct the cookie

Goal	Mechanism
Automatically determine useful cookies	Mirror user session in hidden browser window (the *fork* window) that accepts additional cookies; look for differences in output (see Section 3.2)
Detect differences in pages	Compare page titles; look for user's name/ID in mirrored page; see if a click cannot be mirrored
Determine privacy implications of cookies	Parse and interpret site's P3P policy
Recover from errors	Enable additional cookies and replay user session, using information from the log (see Section 3.3)
Record user session, to enable error recovery	Central log of user's mouse clicks, form field values, and browser state changes (START and STOP events for each page load)

Figure 1: Summary of Doppelganger's cookie management mechanisms.

policy, but also takes action to restore the user's session to a good state, as though cookies had been accepted from the start.

We followed two main principles in designing Doppelganger. The first principle is that users don't like to be constantly interrupted with questions or alerts, and when this happens, they will tend to disable or ignore the offending mechanism [14, 26]. In particular, users should not be asked to do anything manually that can be done automatically. Doppelganger uses *client-side parallelism* to explore alternate policies in the background, trying to find those which result in a positive cost-benefit analysis between privacy loss and functionality gain. In a perfect world, Doppelganger would be able to automatically deduce the ideal cookie policy with no user interaction. In reality, some interaction is needed because users have different privacy preferences and valuations of web site features. Therefore, we assume it is reasonable to expect users to make a small number of high-level privacy decisions.

The second principle we followed in designing Doppelganger is that users don't care about cookies so much as *privacy* and *functionality*. Previous work suggests that: (1) the lack of privacy information in an easily-digestible form may be a significant obstacle to achieving good outcomes for users [1]; (2) users are sensitive to privacy protections, and are more willing to accept a privacy risk if data about them is protected [11]; and (3) users are willing to compromise some amount of privacy if they are offered meaningful incentives to do so [15]. Doppelganger seeks to address all these issues. Instead of requiring users to make uninformed low-level decisions about cookies directly, Doppelganger reformulates cookie decisions as cost-benefit analyses between privacy loss and functionality gains, which are presented to the user. This enables users to make informed decisions regarding their privacy and accept privacy loss when there is commensurate compensation.

1.3 Contributions:

- We introduce Doppelganger, a system for creating and enforcing fine-grained, privacy-preserving cookie policies with low manual effort.
- We show that Doppelganger improves the handling of third-party cookies in Firefox, especially with respect to redirection and inline frames.
- We show how to use client-side parallelism to explore multiple cookie policies simultaneously and find the right balance of privacy and functionality for each user.
- We leverage concepts from Recovery Oriented Computing [5] to implement an automated single-click recovery mechanism.
- We present empirically tuned algorithms for recording and replicating user actions.
- We evaluate the effectiveness of Doppelganger in establishing functional and privacy-preserving cookie policies for typical web browsing habits and compare the results against those obtained with available browser settings.

We summarize Doppelganger's cookie management mechanisms in Figure 1.

2. HTTP COOKIES

HTTP cookies are a general mechanism for web servers to store and retrieve persistent state on web clients [21]. Since HTTP is a stateless protocol, cookies enable web applications to store persistent state over multiple HTTP requests. For example, web shopping applications can use cookies to track which items a user adds to her shopping cart.

When a client makes an HTTP request to a server, the server has the option of including one or more `Set-Cookie` headers in its response. Clients will return these cookies in subsequent HTTP requests using the `Cookie` header. The `Set-Cookie` header has one required field, a name/value pair of the form $NAME = VALUE$. A web server uses this field to encode the state information it wishes to store on the client. There are also four optional fields: `expires`=$DATE$, `domain`=$DOMAIN$, `path`=$PATH\ PREFIX$, and `secure`.

The `expires` field indicates how long the cookie is valid. After that date, the client's web browser should delete the cookie. If the `expires` field is omitted, then the cookie is called a *session cookie* and should be deleted when user closes the web browser. Cookies with an `expires` field are called *persistent cookies*.

The `domain` and `path` fields indicate for which HTTP requests clients should send back cookies. To determine which cookies to include with an HTTP request, the client searches its cookie jar for cookies for domains which suffix-match the domain of the request and paths which prefix-match the path of the request. For example, if the user requests the URL `http://online.foobar.com/store/index.html`, then a cookie with `domain=.foobar.com` and `path=/store` would be included with this request, but a cookie with `domain=pics.foobar.com` would not. The same-origin policy in web browsers prohibits one domain from setting cookies for another. The final optional field, `secure`, indicates whether the cookie should be only sent over encrypted HTTPS connections.

Cookies are also characterized by the context in which they are sent or received. Suppose a user clicks on a link for a particular document, and then the web browser issues a request for that

document. After the browser receives the HTML page from the web server, it parses the page for references to elements needed to render the page, and issues additional HTTP request for these elements. Examples of additional elements include images, Javascript files, stylesheets, Flash objects, and sub-documents. Some of these requests may be to the same domain of the requested document, but some requests may be to different domains. The latter is often the case with advertisements. Content whose URL matches the domain of the main page (i.e., the one in the URL bar) is considered to be *first-party*. All other elements are in *third-party* context. For example, if a user is visiting `www.x.com`, then content served from *.x.com is first-party, whereas content on the page from `www.y.com`, such as an ad, is third-party.

2.1 Uses of cookies

Cookies have many purposes: session state, personalization, authentication, and tracking. Web sites use cookies for personalization to remember users' preferences and settings. For example, Google allows users to customize the format of their search results and uses cookies to remember these preferences. Web sites with user accounts also use cookies to authenticate users' sessions [12]: after a user logs in, a web site can set an session cookie on the user's machine to authenticate her subsequent requests. Web sites can set persistent cookies to remember users and not require a login on subsequent visits. Lastly, web sites can use cookies to track users and their actions. For example, e-commerce sites can track customers' browsing history to make purchase suggestions, and advertising sites can track users to conduct targeted advertising. However, tracking cookies have troubling privacy implications. By tracking the pages a web surfer visits, the web searches she makes, and the items she browses and purchases, web site operators and Internet advertisers can construct sophisticated profiles of users for targeted advertising, data mining, and information sharing with other companies.

Tracking cookies also make cookie management difficult. Many users might prefer not to accept tracking cookies due to the privacy risks; recent studies [19] have found that about 58% of users have deleted their cookies at some point. To prevent her web surfing habits from being tracked, a privacy-conscious user might decide not to accept or send any cookies, but blocking all cookies causes a significant loss in functionality on the web. Most web mail services, e-commerce, and banking sites require users to accept and send cookies for authentication, and blocking cookies also denies users personalization features. Blocking all cookies is consequently impractical for most users.

2.2 Web browser cookie management

Rather than blocking all cookies, the average privacy-conscious user would probably be willing to accept some cookies from the web services she derives some benefit from, but would like to block cookies that compromise her privacy "too much" or provide her no value. Sadly, web browsers provide few useful options to users who wish to customize their cookie settings to this end. Users can configure their browsers to accept only first-party cookies, accept only session cookies, prompt for a decision, and combinations of the above policies.

These options are inadequate. Accepting only first-party cookies is a good start; most web sites do not require clients to accept third-party cookies to operate correctly and advertising companies such as DoubleClick use advertisements and web bugs [3] in conjunction with third-party tracking cookies to correlate users' web browsing across multiple sites. However, current web browsers' implementations of a "first-party only" policy fall short of expectations. For example, Firefox misclassifies IFRAME content as first-party, so advertisers embed ads in IFRAMEs [23] to trick browsers into accepting and sending their otherwise third-party cookies. Also, click-tracking services and advertisers use HTTP redirection [22] to evade third-party cookie blockers. Suppose `www.xyz.com` hires a click-tracking service `www.trackyou.com` to record statistics about its site usage. As a user navigates `www.xyz.com`, say by clicking on a link that seems to point to news articles on `www.xyz.com`, the target of the link may actually be something like `www.trackyou.com/redirect?target=www.xyz.com/news.html`. The user's request first visits `www.trackyou.com`, enabling `www.trackyou.com` to record the request and then redirect the browser to the real target, `www.xyz.com/news.html`. However, since the first request is for `www.trackyou.com`, a browser with a first-party only cookie policy will allow `www.trackyou.com` to set a cookie on the user's machine. The danger here is that if a third site `www.abc.com` and `www.xyz.com` both use the same click-tracker `www.trackyou.com`, this enables `www.abc.com` and `www.xyz.com` to collude with `www.trackyou.com` to determine their common users and track their browsing habits. Furthermore, if a user has an account on either `www.xyz.com` or `www.abc.com` that reveals her real name, this enables both sites to associate her browsing history with her real identity.

Accepting only session cookies also seems like a good idea, since it limits the ability of web sites to track users across browsing sessions. However, blocking all persistent cookies denies users the option of web site personalization and authentication without logging in or another more heavyweight solution. In addition, broadband connections and more effective computer power management make it convenient for users to leave their computers on and browsers open for longer time periods. We anticipate these factors will increase the length of users' average browsing session. A session cookie used over the course of a long browsing session (say, a week) could violate a user's privacy as much as a persistent cookie.

The only existing option for users who want a fine-grained cookie policy is for the web browser to prompt the user for every decision. With this policy, when the browser receives a cookie from a web site `foo.com`, it opens a dialog notifying the user it has received a cookie from `foo.com`, and asks the user whether it should accept the cookie, accept the cookie for each session only, or block it. The dialog also offers the option to apply the decision to every cookie from the same domain. Although in theory this mechanism enables the user to tailor her cookie policy at a fine level of granularity, the usability costs are severe [18]. First, despite the option for the browser to remember her decisions for each domain, a user will often receive a barrage of these interruptive dialogs in a browsing session. Second, although the dialog informs the user that a web site is trying to set a cookie, the user is given no information on how the cookie will be used by the web site and must often make policy decisions before she has even viewed the site's home page. If a user makes a mistake in her policy (e.g., deciding to block cookies at a site she later needs authenticator cookies to login), she must navigate several confusing browser menus (up to three levels deep) to correct her decision. Also, choosing which cookies to accept is non-obvious. She may know she needs to enable cookies for a particular domain to make it "work", but should she enable session cookies or persistent cookies? A user may discover she must enable cookies after she has already taken a series of actions on the web site. In the worst case, she must repeat all these actions after she