Int. J. Inf. Secur. (2012) 11:321–332
DOI 10.1007/s10207-012-0173-6

SPECIAL ISSUE PAPER

Minimizing information disclosure to third parties in social login platforms

Georgios Kontaxis · Michalis Polychronakis · Evangelos P. Markatos

Published online: 4 August 2012

Abstract Over the past few years, a large and ever increasing number of Web sites have incorporated one or more social login platforms and have encouraged users to log in with their Facebook, Twitter, Google, or other social networking identities. Research results suggest that more than two million Web sites have already adopted Facebook's social login platform, and the number is increasing sharply. Although one might theoretically refrain from such social login features and cross-site interactions, usage statistics show that more than 250 million people might not fully realize the privacy implications of opting-in. To make matters worse, certain Web sites do not offer even the minimum of their functionality unless users meet their demands for information and social interaction. At the same time, in a large number of cases, it is unclear why these sites require all that personal information for their purposes. In this paper, we mitigate this problem by designing and developing a framework for minimum information disclosure in social login interactions with third-party sites. Our example case is Facebook, which combines a very popular single sign-on platform with information-rich social networking profiles. Whenever users want to browse to a Web site that requires authentication or social interaction using a Facebook identity, our system employs, by default, a Facebook session that reveals the minimum amount of information necessary. Users have the option to explicitly elevate that Facebook session in a manner that reveals more or all of the information tied to their social identity. This enables users to disclose the minimum possible amount of personal information during their browsing experience on third-party Web sites.

Keywords Social login · Privacy · Web

G. Kontaxis (✉) · M. Polychronakis
Columbia University, New York, NY, USA
e-mail: kontaxis@cs.columbia.edu

M. Polychronakis
e-mail: mikepo@cs.columbia.edu

E. P. Markatos
FORTH-ICS, Heraklion, Greece
e-mail: markatos@ics.forth.gr

1 Introduction

An emerging trend on the Web is *single sign-on* platforms that allow users to register and log in on multiple Web sites using a single account through an OAuth-like protocol [1]. Social networking sites such as Facebook and Twitter have been in the front lines of this trend, allowing their users to use their social networking site credentials in a plethora of third-party Web sites. This type of cross-site interaction enables, for instance, third-party Web sites to authenticate users based on their Facebook or Twitter identities. In addition, such sites may add a social dimension to the browsing experience by encouraging users to "like," share, or comment on certain content using their social network capacity, for example, automatically posting respective favorable messages to their social profile for letting their friends know about the site. To enable this social dimension, third-party sites request access and control over a user's information and social networking account.

In other words, these sites request users to authorize Web applications specific to the third-party site, or API calls originating from the third-party site, to access and control part of their social profile. Unfortunately, this process may have several disadvantages, including:

Loss of anonymity. Even the simple act of signing on to a third-party Web site using a Facebook identity sacrifices

the anonymous browsing of users; their social identities usually contain their real names. In most cases, it is unclear how this loss of anonymity is necessary for the site's purposes.

User's social circle revealed. Several of these third-party Web sites install Web applications in the user's social profile or issue API calls that request access to a user's "friends." Although having access to a user's friends may improve the user's browsing experience, for example, for distributed multi-player games, in most cases, it is not clear why third-party Web sites request this information and how based on it they are going to improve the user's browsing experience.

Loss of track. Once users start enabling a torrent of third-party applications to have access to their personal contacts, they will also probably soon lose control of which applications and sites have access to their personal data, and thus, they will not be able to find out which of them may have leaked information in case of a data breach. As a matter of fact, a recent effort [2], which enables one to become aware of third parties with access to his (private) social information, has surprised users with respect to the amount of data being exposed and the type of permissions granted [3,4].

Propagation of advertisements. Third-party Web sites may request permission to access and act upon a user's social profile (e.g., upload content to it) even when the user is not accessing the third-party site. Such actions may frequently take the form of explicit or implicit advertisements, not necessarily approved by the user.

Disclosure of user credentials. Once a large number of applications start receiving credentials to access user profiles, these credentials may be subject to loss, theft, or accidental leakage. Indeed, recent reports by Symantec suggest that some Facebook applications accidentally leaked access to third parties [5].

Reverse Sign-on Semantics. When an online service prompts users to sign on, they provide their credentials and gain access to data offered by that service. However, in the cases described above, the service is the one being given access to the data of its users, and from that data can select information that may be used to identify individuals.

Although users could theoretically deny this social login approach and the installation of third-party applications, they would essentially exclude themselves from the content and functionality that is offered by the Web site only upon login. This might have been less of a problem if only a handful of third-party Web sites used social login mechanisms; however, recent results suggest that more than two million Web sites have incorporated Facebook social plug-ins [6]. To make matters worse, popular Web sites seem to adopt Facebook social plug-ins even more aggressively. Indeed, as of May 2011, as many as 15 % of the top 10,000 most popular Web sites have adopted Facebook social plug-ins, a notable 300 % increase compared to May 2010 [7]. If this trend continues, as it appears to be, then it will be very difficult for users to browse a significant percentage of Web sites without revealing their personal information.

In this paper, we propose a new way for users to interact with social login platforms so as to protect their privacy; we propose that users surf the Web using downgraded sessions with the social login provider, that is, stripped of excessive or personal information, and with a limited set of privileged actions. Thereby, by default, all interactions with third-party Web sites take place under that privacy umbrella. On occasion, users may explicitly elevate that session on-the-fly to a more privileged or information-rich state to facilitate their needs when appropriate.

Our proposed concept is inspired by privilege separation among user accounts in operating systems, and the UNIX `su` command that upgrades the permissions of a user to those of the `super-user`, if and when the ordinary user needs to perform a privileged instruction. In UNIX, even system administrators typically log in with their ordinary (i.e., non-super-user) accounts and upgrade to super-user status only when they need to execute privileged operations. We have implemented our proposed approach as a browser extension called *SudoWeb* [8].

With SudoWeb, users can maintain two parallel and distinct sessions with Facebook's social login platform, called Facebook Connect, tied to two different social profiles: their primary profile and a second "disposable" profile. The primary session is associated with the users' regular social profiles that contain all of their social contacts, pictures, and personal information—the primary profile is their current profile, if they already have one. The "disposable" session is associated with a second profile that is a stripped-down version of the primary one. It should contain no personal information, social contacts, or other sensitive information that the user is not comfortable sharing with third-party Web sites.

By default, the browser keeps the appropriate state, that is, active sessions and cookies, to maintain the "disposable" session alive. As a result, whenever users employ social login just to bypass the registration step in various Web sites, they will surrender only a small portion of their information or no actual information at all, as the "disposable" session with Facebook will be used. If at any point users wish to switch to their actual profile, for example, in case they explicitly wish to associate their real identity with a third-party Web site or online application, they can easily do so by switching to their primary session through SudoWeb's user interface.

This work is an extended version of our previous work that introduced SudoWeb [8], including the results of a real-world study on the permissions requested by third-party Web sites that have integrated Facebook Connect, as well as a more detailed description of SudoWeb's design and

implementation. In summary, the main contributions of this paper are the following:

- We identify and describe a threat to user privacy stemming from the increasing deployment of social login frameworks, which allow third-party Web sites to gain access to personal information stored in user profiles on social networking services.
- We propose a new privacy-preserving framework for users to interact with single sign-on and OAuth-like platforms provided by social networks in their daily activities on the Web.
- We implement a prototype of our framework as a browser extension for the Google Chrome browser. Our prototype supports Facebook's social login architecture [9] and can be easily extended to support others, such as "Sign in with Twitter" [10].
- We evaluate our implementation and show that (i) it allows users to protect their privacy when signing in on third-party Web sites, and (ii) it does not affect any open sessions they might have with other third-party Web sites that use the same social login mechanisms.

The rest of the paper is organized as follows: in Sect. 2, we provide some background information on the OAuth protocol, which is the basis of most single sign-on architectures, and discuss Facebook's social login platform. In Sect. 3, we present the results of a study on the permissions requested by 755 third-party Web sites that have incorporated Facebook's social login platform. In Sect. 4, we outline the design of our architecture and in Sect. 5 detail our proof-of-concept implementation of a browser extension. In Sect. 6, we discuss how existing social login platforms could provide better protections for users' privacy. Finally, in Sect. 7, we present related work in the area and conclude in Sect. 8.

2 Background

In this section, we provide background information on the OAuth protocol [1], which is the primary method for implementing single sign-on functionality across multiple Web sites. We also detail Facebook's single sign-on platform [9], which as of January 2012 is the most popular social login platform with more than 2.5 million Web sites using it [11].

2.1 OAuth protocol

The OAuth or Open Authentication protocol [1] provides a method for clients to access server resources on behalf of a resource owner. In practice, it is a secure way for end users to authorize third-party access to their server resources without sharing their credentials.

As an example, one could consider the usual case in which third-party sites require access to a user's e-mail account so that they can retrieve his contacts in order to enhance the user's experience in their own service. Traditionally, the user has to surrender his username and password to the third-party site so that it can log into his account and retrieve that information. Clearly, this entails the risk of the password being compromised. Using the OAuth protocol, the third party registers with the user's e-mail provider using a unique application identifier. For each user that the third-party requires access to his e-mail account, it redirects the user's browser to an authorization request page located under the e-mail provider's own domain and appends the site's application identifier so that the provider is able to find out which site is asking for the authorization. That authorization request page, located in the e-mail provider's domain, validates the user's identity (e.g., using his account cookies or by prompting him to log in) and subsequently asks the user to allow or deny information access to the third-party site. If the user allows such access, the third-party site is able to use the e-mail provider's API to query for the specific user's e-mail contacts. At no point in this process does the user have to provide his password to the third-party site.

2.2 Facebook authentication

Facebook's social login platform, known as Facebook Connect [9], is an extension to the OAuth protocol that allows third-party sites to authenticate users by gaining access to their Facebook identity. This is convenient for both sites and users; sites do not have to maintain their own accounting system, and users are able to skip yet another account registration and thereby avoid the associated overhead. A "login with Facebook" button is embedded in a third-party Web site and, once clicked, directs the user's browser to a Facebook server where the user's cookies or credentials are validated. Upon successful identity validation, Facebook presents a "request for permission" dialog where the user is prompted to allow or deny the actions requested by the third-party Web site, for example, social plug-in interactions or access to various information in the user's social profile. However, the user is not able to modify or regulate the third-party Web site's requests, for instance to allow access to only a part of the information the site is requesting. If the user grants permissions to the site's request, Facebook will indefinitely honor API requests originating from that third-party site that conform to what the user has just agreed upon.

3 Social login versus user privacy

To gain a better understanding of the type and extent of the permissions requested by third-party Web sites through the

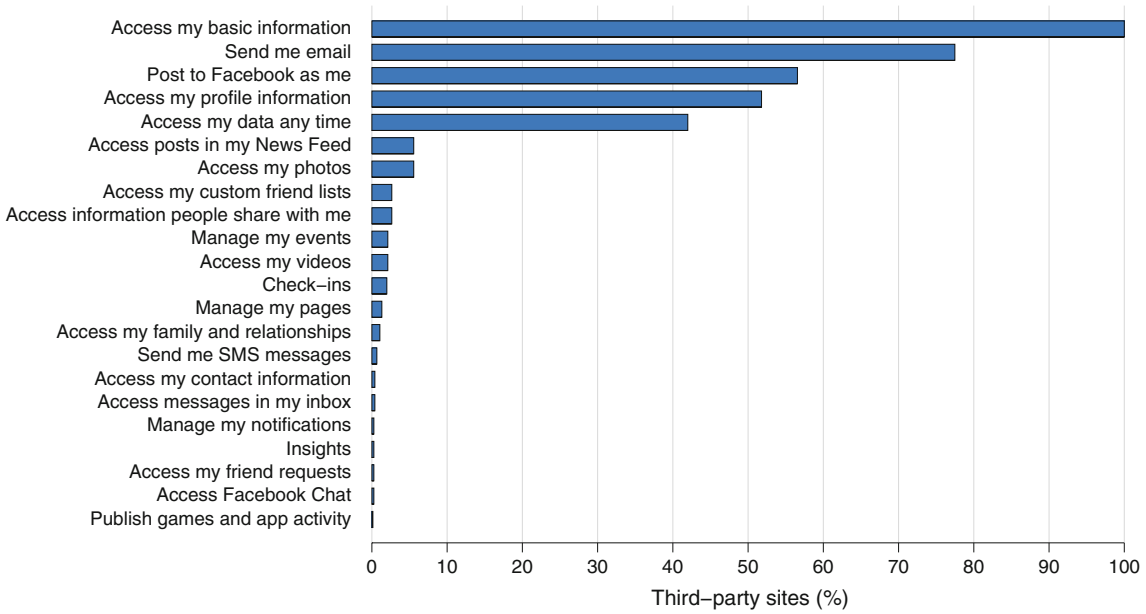

Fig. 1 Distribution of requested permissions for a set of 755 Web sites that have integrated Facebook's single sign-on platform

Facebook Connect mechanism, also known as "login with Facebook," we studied a random sample of 755 sites that have incorporated Facebook's social login platform. Figure 1 presents the frequency distribution of the different permissions requested by these Web sites. A full list of the available permissions can be found through Facebook's developer page [12].

One may notice that all sites request access to a user's basic information. This is the minimum amount of private information a user must disclose, even if a third-party Web site does not really need all that information. According to the description of this type of permission, the basic information includes the "user id, name, profile picture, gender, age range, locale, networks, list of friends, and any other information they have made public."

Besides basic profile information, the administrator of a third-party Web site may explicitly ask for additional permissions to access more user information or perform certain actions on behalf of the user. For example, 77 % of the studied sites request access to the user's e-mail address, 57 % are able to post content on behalf of the user, and more than 42 % require to be able to indefinitely access user information even when a user is not using the application.

Moreover, permission to manage Facebook notifications could enable malicious third parties to hide the misuse of other permissions granted to them. What is more, access to direct messages sent or received and Facebook's real-time chat system, could seriously compromise a user's private communications. Finally, special consideration should be given to permissions that may result in real-world consequences for the user, for example, the ability of a third-party to access information about the user's physical location ("*Check-ins*") or send SMS messages, which may result in monetary charges.

We argue that in most cases the type of permissions and the amount of information requested from the user during social login are more than necessary. Even with benign third parties, the more personal data being shared, the greater the damage in case of leaks either accidental or as a result of an attack. To give an example, one of the cases in our study is a music band that urges its fans to perform a social login when visiting its site. Although we could not confirm the presence of functionality dependent upon social login, we will give the site the benefit of the doubt. However, its requirements are over the top. It requests access to basic, contact and profile information, photos and videos, and even to the user's e-mail address and Facebook chat. Moreover, access is requested even when the user is not using the site. Finally, it requests the ability to upload content on Facebook on behalf of the user, and to read and manage the user's events and reports on his physical location.

Figure 2 shows a screenshot of the social login dialog for that site (its name has been anonymized). Access to all of the user's photos and videos seems unjustified, as is access to the user's private conversations. Furthermore, the ability to impersonate the user on Facebook is in no way restricted to purposes related to the nature of the third party. Finally, managing all events and physical location information so it can, for example, generate activity related to the band clearly demonstrates the need for fine-grained permissions. Ideally, the third party would request access to photos tagged with a certain keyword related to the band, manage

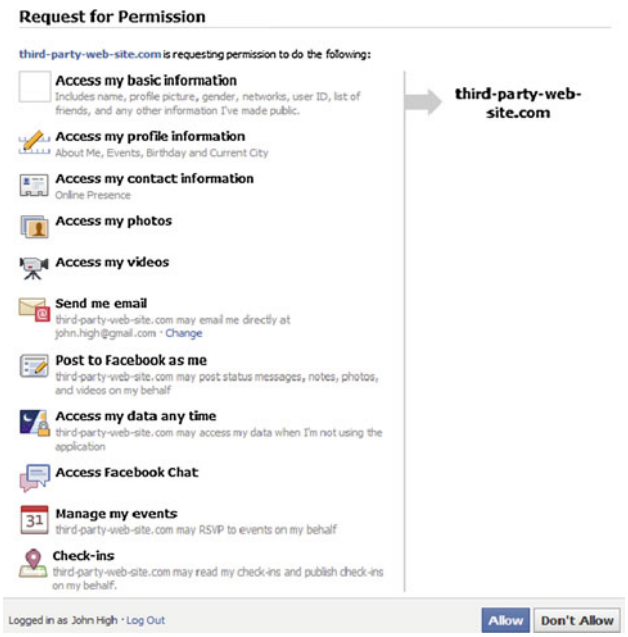

Fig. 2 Example case of a third-party Web site requesting permission to access and manage an excessive amount of personal user information. The user can only allow everything or nothing (thus aborting social login), and any kind of fine-grained control over the permissions is absent

events and locations with specific prefixes in their names, and add a "uploaded by 'X' on behalf of user 'A' " label to content uploaded on Facebook.

Overall, the above study confirms our intuition that the amount, type, and combination of permissions requested by third-party sites can potentially put users at risk. At the same time, user reactions to a recent effort [2] that enables users to become aware of third-party applications and Web sites with access to their (private) social information confirm the general demand for improved control and better protection over the data one uploads to a social network. Facebook itself acknowledges the issue and, in a small effort to remedy the problem, offers users the option to anonymize the e-mail address they surrender to third parties. This option is unfortunately opt-in and is enabled by default only on rare occasions driven by abuse-related heuristics.

Motivated by this issue, in the rest of this paper, we present the design and implementation of a framework for minimum information disclosure across third parties in social login platforms.

4 Design

The modus operandi we assume in our approach is the following:

1. The user browses the Web having opened several tabs in her browser.
2. Then, the user logs in her ordinary Facebook account so as to interact with friends and colleagues.
3. While browsing at some other tab of the same browser, the user encounters a third-party Web site asking her to log in with her Facebook credentials. At this point in time, our system kicks in and establishes a new and separate downgraded session with Facebook for that cross-site interaction. That session is tied to a stripped-down version of her account which reveals little, if any at all, personal information. Now:
   (a) The user may choose to follow our "advice" and log in with this downgraded Facebook session. Observe that this stripped-down mode does not affect the browsing experience of the user in the tabs opened at step 2 above: the user remains logged in with her normal Facebook account in the tabs of step 2, while in the tabs of this step she logs in with the stripped-down version of her account. Effectively, the user maintains two sessions with Facebook:
      (i) One session logged in with her normal Facebook account, and
      (ii) One session logged in with the stripped-down version of her account.
   (b) Alternatively, the user may want to override our system's logic and log in with her normal Facebook account revealing her personal information; in such cases, she performs a "sudo" on that particular cross-site interaction with Facebook and elevates the by-default downgraded Web session.

In the description of our design, we assume the use of Facebook Connect [9,13]; however, SudoWeb can be extended to cover other social login platforms as well.

Figure 3b shows the architecture of our system. To understand our approach, we will first describe in Fig. 3a how an ordinary Web browser manages session state. We see that the browser uses a default session store (Session Store [0] (default)) that stores all relevant state information, including cookies. Thus, when the user logs into Facebook (or any other site for that matter) using her ordinary Facebook account, the browser stores the relevant cookie in this default session store. When the browser tries to access Facebook from another tab (Tab 3 in the figure), the cookie is retrieved from the default session store and the page is accessed using the same state as before.

In our design, we extend this architecture by including more than one session stores. As shown in Fig. 3b (bottom left), we have added "Session Store [1]" that stores all relevant information, including cookies, for the stripped-down Face-

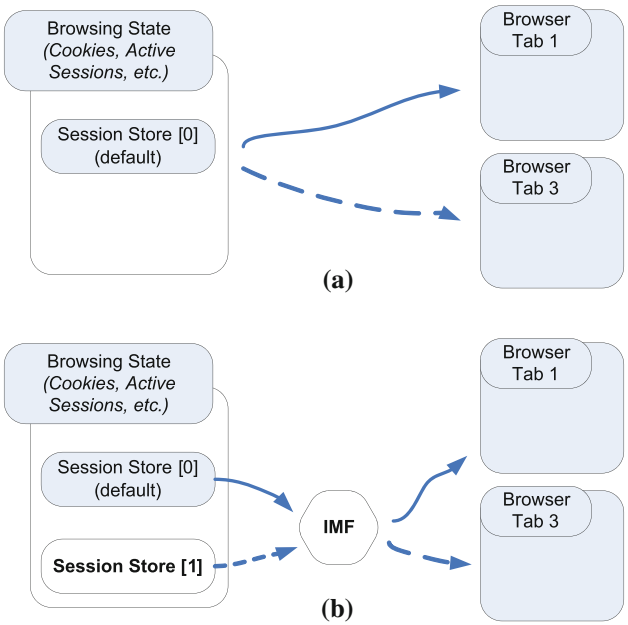

Fig. 3 Typical communication of session state to loaded pages (**a**), and how SudoWeb handles the same communication using multiple session stores (**b**)

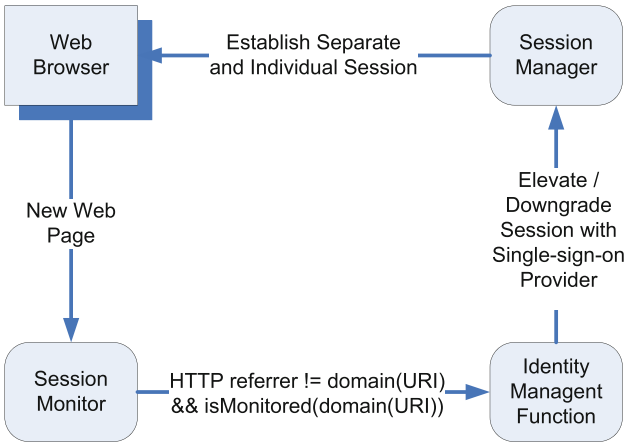

Fig. 4 SudoWeb modules

book session. This gives us the opportunity to enable users to surf the Web using two distinct and isolated sessions with Facebook at the same time: a session tied to the "normal" account is enabled in Tab 1 while a stripped-down session is in effect in Tab 3. To select the appropriate account, our system (IMF) intercepts all URL accesses and checks their HTTP referrer field. If the URL points to Facebook Connect but the HTTP referrer field belongs to a different domain name, then our system suspects that this is probably an attempt from a third-party Web site to authenticate the user with her Facebook credentials.

Therefore, as it stands inline between the loading page and the browser's state store(s), it supplies the appropriate state (from Session Store [1]) for the stripped-down Facebook session to be employed. This is an implicit privacy suggestion toward the user. If the user disagrees, she may choose to authenticate with her ordinary Facebook account, in which case, Tab 3 will receive all cookies from Session Store [0].

We consider the proposed concept as analogous to privilege separation in operating systems, that is, different accounts with different privileges, such as root and user accounts. Our design can scale and evolve so that it accommodates different privacy-preserving scenarios in interaction with third-party Web sites.

Figure 4 shows the modules of our system. Initially, in the upper left corner, the user browses ordinary Web pages (Web Browser). When a new browser page (i.e., tab or window) is created (New Web Browser Page), the Session Monitor kicks in to find whether this is a social login attempt.[1] If (i) it is such an attempt (i.e., isMonitorred(domain(URI)) is TRUE) and (ii) the attempt is from a third-party Web site (i.e., HTTP referrer != domain(URI)) then our system calls the Identity Management Function (IMF) that employs a downgraded, stripped-down from all personal information, session for the user. From that point onwards, the session manager manages all the active sessions of the user, in some cases different sessions with different credentials for the same single sign-on domain. Figure 5 shows the workflow of our system in more detail.

5 Implementation

We have implemented our proposed architecture as a browser extension for the latest version of the Google Chrome browser[2] with support for the Facebook Connect social login platform. Due to the platform's popularity, our proof-of-concept implementation covers a great part of social login interactions on the Web [7,14]. Nevertheless, our browser extension can be seamlessly configured to support a greater variety of such cross-site social login interactions.

5.1 SudoWeb modules

Here, we describe the modules that comprise our extension to the Google Chrome browser, in support of our proposed architecture.

[1] SudoWeb keeps a list with all social login domains currently supported and thus monitored. If such a domain is monitored the isMonitorred(domain(URI)) function returns TRUE.

[2] As we take advantage of generic functionality in the extension-browser communication API, we find it feasible to also port the extension to Mozilla Firefox. It is noted that Chrome and Firefox account for almost 60 % of the Web browser users [15].

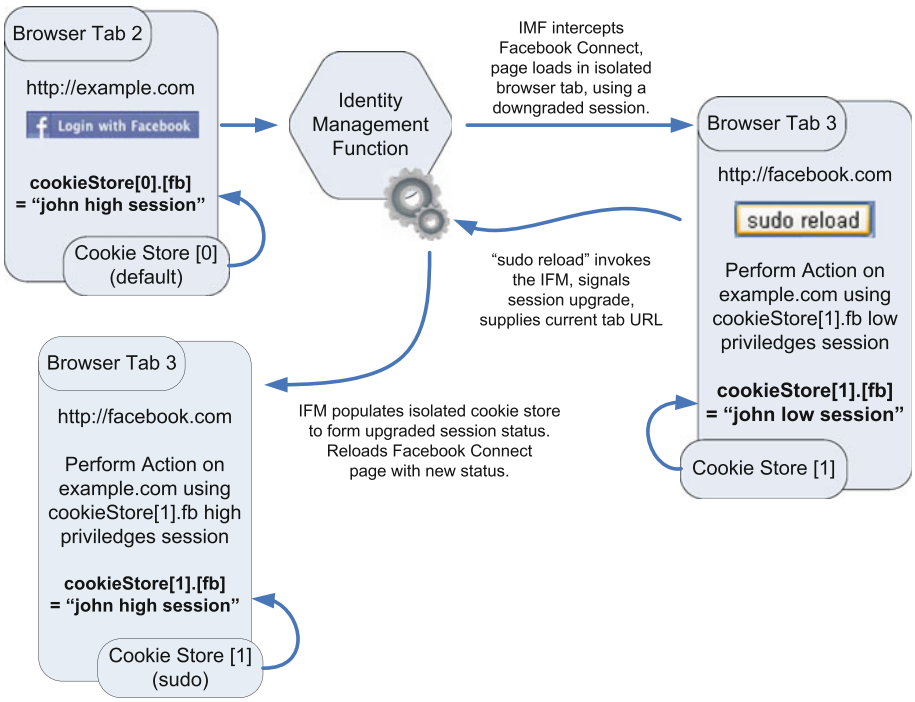

Fig. 5 Outline of SudoWeb's workflow

5.1.1 Identity management function (IMF)

In the heart of the extension lies the logic module offering the identity management function, or IMF. This function is responsible for detecting the possible need for elevating or downgrading a current session with a social login provider (here: Facebook).

Such a need is detected by identifying differences in the HTTP referrer domain and the URL domain of pages to be loaded. That is, when the user navigates away from a third-party Web site (identified by the HTTP referrer field) toward a social login Web site (we keep a configuration file with all social login sites supported), IMF steps in, instantiates a new, isolated, and independent session store in the browser, and instructs the *session manager module* to initialize it. This allows the browser to receive state that establishes a downgraded or stripped-down session with the social login provider. Furthermore, it places a "sudo reload" HTML button on that page giving the user the opportunity to reload that page using an elevated session instead.

5.1.2 Session monitor

The session monitor module plays a supporting role to the IMF. If one considers our extension as a black box, the session monitor stands at its input. It inspects new pages opening in the browser and looks for cases where the page URL belongs to a monitored domain of a social login provider (here: Facebook) but the page has been invoked through a different, third-party domain. It does so by comparing that URL with the HTTP referrer. The referrer is an HTTP parameter supplied by the browser itself based on the URL of the parent tab or window that resulted in a child tab or window being spawned.

The session monitor notifies the IMF of such incidents and supplies the respective page URL. We should note that recent research has revealed that the HTTP referrer field can be empty or even spoofed [16,17] undermining all mechanisms based on it. Although it is true at the network elements may remove or spoof the HTTP referrer field so that it will be invalid when it reaches the destination Web server, our work with the HTTP referrer field is at the Web *client* side, not at the Web server side. That is, the HTTP referrer field is provided to SudoWeb by the Web browser *before* it reaches any network elements that may remove or spoof it.

5.1.3 Session manager

The session manager module also plays a supporting role to the IMF. If one considers our extension as a black box, the session manager stands at its output. Upon the installation of our extension, the session manager prompts the user of the Web browser to fill in his ordinary social login (here: Facebook) account username and password, as well as the stripped-down version that is to be used for downgraded integrations with third-party Web sites.

The session manager stores the necessary state, for example, cookies, required to establish the two distinct sessions

Fig. 6 Screenshot of the configuration page for the *session manager module*. The user is required to provide the necessary information once so that SudoWeb can alternate between downgraded and elevated privilege sessions with the social login provider. Here, the user declares the two Facebook identities that will be used for that purpose. At the bottom of the configuration page, a session test retrieves the names of the two identities from Facebook in real time, demonstrating how two parallel Facebook sessions can be seamlessly maintained

with the social login provider and is responsible for populating the browser's cookie store once instructed by the IMF. As a result, it stands at the output of our extension and between the browser's session store and the rendered pages that reside in tabs or windows. It affects the state upon which a resulting page relies.

Our extension takes advantage of the incognito mode in Google Chrome to launch a separate browser process with isolated cookie store and session state so that when the session manager pushes the new state in the cookie store, the user is not logged off from the existing elevated session (here: with Facebook) that may be actively used in a different browser window.

5.2 Operation and interaction of SudoWeb modules

In the spirit of the use case presented at the beginning of Sect. 4, a user of SudoWeb will configure the *session manager* once, continue browsing the Web, and eventually come across a third-party that wishes to interact with his Facebook identity via a social login.

The configuration of the *session manager* provides the information necessary to seamlessly alternate between downgraded and elevated privileges with one or more social login providers. Figure 6 presents an example screenshot of such a configuration for Facebook, where the user declares the two identities that will be used for that purpose; two buttons are used to indicate to the *session manager* that the current session the user has with Facebook is to be treated either as a primary (elevated) or secondary (downgraded) session. The user logs in to Facebook with one identity, assigns one of the two characterizations, is then briefly logged out of that identity so that he can repeat the process for the second identity, and finally his original session with Facebook is restored, making the entire process minimally disrupting. At the bottom of the configuration page, a session test, retrieving the names of the two identities from Facebook in real time, demonstrates how two parallel Facebook sessions can be seamlessly maintained.

After it has been configured, SudoWeb monitors the Web browser for social login events. Upon the user reaching a third-party page and clicking on the "login with Facebook" button, our system kicks in:

1. The *session monitor* detects the launch of a new Facebook page from a page under the domain of the third-party Web site. The *session monitor* notifies the *IMF* module of our extension and so the page launch is intercepted and loaded in an incognito window, that is, an isolated browser process with a separate and individual session store.
2. The *IMF* coordinates with the *session manager module* so that this isolated environment is populated with the necessary state for a downgraded Facebook session to exist.

The entire process happens in an instant and the user is presented with a window similar to the one shown in Fig. 7—we use third-party-web-site.com as the name of the third-party Web site that wants to authenticate the user using her Facebook account. We see that in addition to authenticating the user, the third-party Web site asks for permission to (i) send email to the user, (ii) post on the user's wall, (iii) access the user's data any time, and (iv) access the user's profile information. Although Facebook enables users to "Allow" or "Don't Allow" access to this information (bottom right corner), if the user chooses not to allow this access, the entire authentication session will terminate and the user will not gain access to the content of third-party-web-site.com.

Having intercepted this third-party authentication operation, SudoWeb brings the stripped-down account (i.e., John Low) forward, on behalf of the user. Therefore, if the user chooses at this point to allow the third-party site access to his information, only a small subset of his actual information will be surrendered. Note that a "sudo reload" button has been placed at the bottom of the page, allowing the user to elevate this session to the one tied to his actual, or a more privileged, Facebook identity.

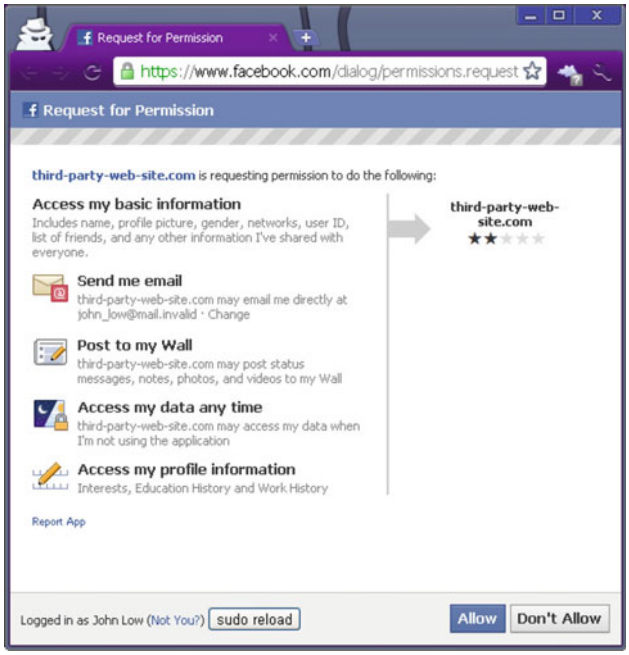

Fig. 7 Example screenshot of a Facebook "Request for Permission" page that has been invoked by the fictional site third-party-web-site.com so that the user may authorize that site to access his Facebook account information. By default, SudoWeb maintains a downgraded-status session with Facebook, using the appropriate state to be logged in with the disposable account "John Low." The user has the option to switch to an elevated-status session via the "sudo reload" button at the bottom of the window

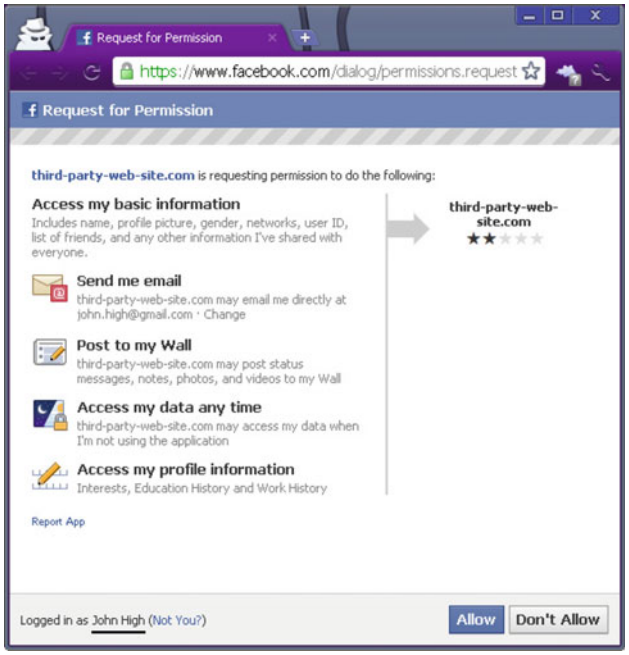

Fig. 8 Example screenshot of the previous Facebook page, after the "sudo reload" option has been selected by the user; the page has been reloaded using transparently the necessary session state to maintain an elevated-status Facebook session using the account "John High"

Figure 8 shows a screenshot of the browser window; the user will see whether he chooses to elevate the session. Note that, at the bottom of the browser's page, the user is no longer considered to be logged in as "John Low" but as "John High."

The Facebook session with which the user was surfing prior to engaging in this cross-site Facebook interaction remains intact in the other open browser windows since, as mentioned earlier, we take advantage of the browser's incognito mode to initiate an isolated session store in which we manage the escalation and de-escalation of user sessions. All the user has to do is close this new window to return to his previous surfing activity.

6 Discussion

Here, we discuss how social networks providing single sign-on interaction could evolve to facilitate user needs and better protect their privacy. We also propose a series of requirements from third-party Web applications in terms of "fair play."

6.1 Fine-grained privacy settings

Inspired by the privilege separation principles of UNIX, SudoWeb presents a step toward surfing the Web using several distinct sessions: each session with different privileges. We have implemented the philosophy of our system using parallel Web sessions tied to distinct Facebook accounts; each account revealing a different amount of information. We believe that the increasing privacy concerns of users will motivate social login providers to offer more fine-grained disclosure of user information and more control over the user's privacy in a single account. If that happens, the concept of our system will still be valid, but implemented closer to the mechanics of social login providers.

6.2 Fairness

Current single sign-on mechanisms in social networks are especially unfair to people with rich social circles. For example, if a third-party Web site wants to install an application that has access to all of a user's friends in return for a service, this is unfair to people who have lots of friends, compared to people who have (or have declared) no friends. Both types of users will get the same kind of service at a different price: the first category will reveal the names of lots of friends while the second will reveal none. To make matters worse, this cost (and unfairness) seems to increase with time: as the user accumulates friends, the installed third-party application will continue to have access to all of them.[3]

[3] Unless the user explicitly uninstalls the application.

We believe that social login mechanisms should: (i) Restrict themselves only to authentication and refrain from asking access to more personal information, such as friends and photos. (ii) If they do ask for more personal information, they should make clear how they are going to use it and how this will benefit the user. (iii) If the user denies the provision of more personal information, the social login mechanisms should continue to function and provide their services to the users.

6.3 Terms of use

It may seem that our approach conflicts with the terms of use for some sites. For example, maintaining multiple accounts is a violation of the terms of use of Facebook, while it appears not to be a violation of the terms of use of Google. We believe that this conflict stems from the fact they have not yet caught up with the changing needs of users. For example, several users maintain two Facebook profiles: one personal profile with all their personal contacts, friends, and relatives, and one professional profile where their "friends" are their colleagues and business contacts. The postings that run on their personal profiles are quite different from the postings than run on their professional profiles. Even the language of these postings may be different.

Forcing those users to have a single Facebook account will make their social interactions more difficult or will force them to move one of their profiles, for example, the professional one, to another social network, such as LinkedIn. We believe that sooner or later successful social networking sites will catch up with changing user needs and will adapt their terms of use to suit users. Otherwise, users might shift to social login providers that are closer to their needs.

7 Related work

Ardagna et al. [18] highlight the practice of Internet services requiring user information in return for accessing their digital resources. They coin the concept of a user portfolio containing personal data and propose the use of sensitivity labels that express how much the user values different pieces of information. Furthermore, they assume scenarios where an atypical negotiation takes place between the user and the server, in which the server prompts the user to choose among disclosing alternative pieces of information. The user decides on the type and amount of information disclosed in relation to the type and amount of digital resources being offered.

Facecloak [19] shields a user's personal information from a social networking site and any third-party interaction, by providing fake information to the social networking site and storing actual, sensitive information in an encrypted form on a separate server. At the same time, social functions are maintained.

Felt et al. [20] studied the 150 most popular Facebook applications and found that almost all of them required too much user information for their purposes. They propose the use of a proxy to improve social networking APIs such that third-party applications are prevented from accessing real user data while social functions are not affected.

The xBook [21] framework addresses threats against the privacy of social network users due to information leaked via interactions with third-party applications. It provides a trusted hosting environment where untrusted applications are split into components with a manifest to detail security permissions in terms of user data access and communication between components or remote locations. xBook takes up the role of enforcing that manifest at run-time.

OpenID [22] is a platform supporting a federation of single sign-on providers. Its nature of operation has been described in Sect. 2. An interesting feature of OpenID is the support for multiple identities per user; upon receiving a user-identification request from a third-party site and after authenticating with the user, it may decide to return a different identity for the same user to different third-party sites.

PseudoID [23] is a privacy enhancement for single- sign-on systems like OpenID or Facebook Connect. As third-party sites interact with the single sign-on provider to acquire access to a user's identity, that provider is able to correlate a user's identity with the sites she logs into. In PseudoID users set up the profiles or identities they wish to use with third-party sites and employ the PseudoID's blind signature service to cryptographically blindly sign such tokens of information. When they need to identify themselves to a third-party site, just as before, that site interacts with PseudoID to retrieve the user's identity. Contrary to the traditional model, the user does not log in to PseudoID, thereby allowing the service to associate her with that particular third-party site request. The user presents to PseudoID a blindly signed identity and PseudoID, after checking the validity of the cryptographic signature, forwards that identity to the third-party site.

Concurrently and independently to our work, a user-friendly mechanism for users to switch between Google Chrome profiles is being developed [24]. At the moment, one is able to use multiple browser profiles by adding a data-directory flag when invoking the browser. Browser profiles contain their own cookie store, browser settings, and installed extensions. By using different profiles, among other things, one is able to switch between cookie stores and therefore between Web site identities.

Our approach is similar to that feature of Chrome but at the same time bears significant differences. While Google is building a profile manager for browsers, we design a more generic privacy-preserving framework that describes

information and privilege separation in Web sessions involving cross-site interaction. While Chrome's profiles bundle sessions with different sites in a single browser profile, we operate on a more flexible basis where we populate an isolated and distinct browser instance with the state of only those sessions that the user has explicitly activated. Moreover, while in the Chrome feature the user is responsible for switching between the different identities and profiles, we employ heuristics that automatically detect the need to switch to a downgraded Web session. Therefore, we do not have to rely on the user's alertness to protect his information.

8 Conclusion

Recent results suggest that hundreds of thousands of Web sites have already employed single sign-on mechanisms provided by social networks such as Facebook and Twitter. Unfortunately, this convenient authentication usually comes bundled (i) with a request for the user's personal information, as well as (ii) a request to act upon a user's social network on behalf of the user, for example, for advertisement. Unfortunately, the user cannot deny these requests, if she wants to proceed with the authentication.

In this paper, we explore this problem and propose a framework to enable users to authenticate to third-party Web sites using single sign-on mechanisms provided by popular social networks while protecting their privacy; we propose that users surf the Web using downgraded sessions with the social login platform, that is, stripped of excessive or personal information and with a limited set of privileged actions. Thereby, by default, all interactions with third-party Web sites take place under that privacy umbrella. On occasion, users may explicitly elevate that session on-the-fly to a more privileged or information-rich state to facilitate their needs when appropriate.

We have implemented our approach in SudoWeb, a Chrome browser extension with current support for Facebook's social login platform. Our results suggest that SudoWeb is able to intercept attempts for third-party Web site authentication and handle them in a way that protects user privacy, while not affecting any other ongoing Web sessions that a user may concurrently have.

Availability

The source code of SudoWeb is available at http://www.cs.columbia.edu/~kontaxis/sudoweb/.

Acknowledgments This work was supported in part by the FP7-PEOPLE-2009-IOF project MALCODE and the FP7 project SysSec, funded by the European Commission under Grant Agreements No. 254116 and No. 257007. Evangelos Markatos is also with the University of Crete. Most of the work of Georgios Kontaxis was done while at FORTH-ICS.

References

1. OAuth. http://oauth.net/
2. Start 2012 by taking 2 minutes to clean your apps permissions. http://mypermissions.org/
3. Sophos—one-stop shop to clean up social media permissions. http://nakedsecurity.sophos.com/2012/01/05/mypermissions-clean-up-social-media-permissions/
4. Mashable—Check out who has access to your social media accounts. http://mashable.com/2012/01/04/mypermissions/
5. Symantec Official Blog—Facebook applications accidentally leaking access to third parties. http://www.symantec.com/connect/blogs/facebook-applications-accidentally-leaking-access-third-parties
6. WebProNews—Million sites have added Facebook's social plugins since f8. http://www.webpronews.com/2-million-sites-have-added-facebooks-social-plugins-since-f8-2010-09
7. BuiltWith—Facebook for Websites Usage Trends. http://trends.builtwith.com/javascript/Facebook-for-Websites
8. Kontaxis, G., Polychronakis, M., Markatos E.P.: SudoWeb: Minimizing information disclosure to third parties in single sign-on platforms. In: Proceedings of the 14th Information Security Conference (ISC), pp. 197–212. Springer, Berlin October (2011)
9. Facebook for Websites. https://developers.facebook.com/docs/guides/web/
10. Sign in with Twitter. http://dev.twitter.com/pages/sign_in_with_twitter
11. Facebook Statistics. https://www.facebook.com/press/info.php?statistics
12. Facebook Developers—Permissions. https://developers.facebook.com/docs/reference/api/permissions/
13. Stone, B.: Facebook aims to extend its reach across the web. New York Times (2008) http://www.nytimes.com/2008/12/01/technology/internet/01facebook.html
14. BuiltWith—OpenID usage statistics. http://trends.builtwith.com/docinfo/OpenID
15. StatCounter—Top 5 browsers (2011–2012). http://gs.statcounter.com/#browser-ww-weekly-201120-201220
16. Barth, A., Jackson, C., Mitchell, J.C.: Robust defenses for cross-site request forgery. In Proceedings of the 15th ACM conference on computer and communications security, pp. 75–88. ACM (2008)
17. Meiss, M., Duncan, J., Gonçalves, B., Ramasco, J.J., Menczer, F.: What's in a session: tracking individual behavior on the web. In: Proceedings of the 20th ACM conference on hypertext and hypermedia, pp. 173–182. ACM (2009)
18. Ardagna, C.A., De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Samarati, P.: Supporting privacy preferences in credential-based interactions. In: Proceedings of the 9th annual ACM workshop on Privacy in the electronic society, pp. 83–92. ACM (2010)
19. Luo, W., Xie, Q., Hengartner, U.: Facecloak: An architecture for user privacy on social networking sites. In Proceedings of the international conference on computational science and engineering, pp. 26–33. IEEE Computer Society (2009)
20. Felt, A., Evans, D.: Privacy protection for social networking platforms. In: Proceedings of the workshop on web 2.0 security and privacy (2008)

21. Singh, K., Bhola, S., Lee, W.: xbook: redesigning privacy control in social networking platforms. In: Proceedings of the 18th conference on USENIX security symposium, pp. 249–266. USENIX Association, Berkeley (2009)
22. OpenID Foundation—OpenID authentication 2.0 specifications. http://openid.net/specs/openid-authentication-2_0.html
23. Dey, A., Weis, S.: PseudoID: Enhancing privacy in federated login. In: Hot topics in privacy enhancing technologies, pp. 95–107. Springer, Berlin (2010)
24. The chromium projects—multiple profiles. http://www.chromium.org/user-experience/multi-profiles