SudoWeb: Minimizing Information Disclosure to Third Parties in Single Sign-On Platforms

Georgios Kontaxis[1], Michalis Polychronakis[1], and Evangelos P. Markatos[2]

[1] Computer Science Department, Columbia University, USA
{kontaxis, mikepo}@cs.columbia.edu
[2] Institute of Computer Science,
Foundation for Research and Technology – Hellas, Greece
markatos@ics.forth.gr

Abstract. Over the past few months we are seeing a large and ever increasing number of Web sites encouraging users to log in with their Facebook, Twitter, or Gmail identity, or personalize their browsing experience through a set of plug-ins that interact with the users' social profile. Research results suggest that more than two million Web sites have already adopted Facebook's social plug-ins, and the number is increasing sharply. Although one might theoretically refrain from such single sign-on platforms 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 the 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 across third-party sites with single sign-on interactions. Our example case is Facebook, which combines a very popular single sign-on platform with information-rich social networking profiles. When a user wants to browse a Web site that requires authentication or social interaction with his Facebook identity, our system employs, by default, a Facebook session that reveals the minimum amount of information necessary. The user has the option to explicitly elevate that Facebook session in a manner that reveals more or all of the information tied to his social identity. This enables users to disclose the minimum possible amount of personal information during their browsing experience on third-party Web sites.

1 Introduction

An emerging trend on the Web is "single sign-on" initiatives where users register and log on in multiple Web sites using a single account and an OAuth-like protocol [4]. Social networking sites, such as Facebook and Twitter, have been in the front lines of this initiative, allowing their users to utilize their social profiles 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) identity. In addition, such sites may add a social dimension to a user's browsing experience by encouraging him to "like," share, or comment on their content using his social network capacity, i.e., automatically post respective favorable messages to his social profile and let his friends know about the site. To enable this social dimension, third-party sites request access and control over the user's information and 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 or whole 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 the Facebook identity sacrifices the anonymous browsing of the user; his social identity usually contains his real name. 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 which request access to a user's "friends." Although having access to a user's friends may improve the user's browsing experience, e.g., 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 to enable a torrent of third-party applications to have access to their personal contacts, they will 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 the data in a case of a data breach.
- **Propagation of advertising information in user's social network.** Several of these third-party sites 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 users' credentials to unauthorized applications.** Once a large number of applications starts receiving credentials to access a user's profile, such credentials may be subject to loss or theft, or accidental leakage. Indeed, recent reports by Symantec suggest that such Facebook applications accidentally leaked access to third parties [7].
- **Reverse Sign-on Semantics**. When a service prompts a user to sign on, he provides his credentials and gains 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 the user, and from that data selects information that may be used to identify or authenticate the user.

Although a user could theoretically deny this single sign-on approach, and the installation of the third-party application, many Web sites respond to this disapproval usually by diminishing the user's browsing experience significantly,

cutting the user off from the largest part of their sophisticated functionality. Although this might be less of a problem if only a handful of third-party Web sites used this single sign on mechanism, recent results suggest that more than two million Web sites have added Facebook social plug-ins [9]. 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 whooping 300% increase compared to May 2010 [1]. If this trend continues, as it appears to be, then it will be very difficult for users to browse a significant percentage of the Web sites without revealing their personal information.

In this paper, we propose a new way for users to interact with single sign-on platforms so as to protect their privacy; we propose that users surf the Web using downgraded sessions with the single sign-on platform, i.e., stripped from 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 which 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 initially log in with their ordinary (i.e., non super-user) accounts and upgrade to super-user status if and when they need to execute privileged operations.

For instance, we propose that users may use two parallel and distinct sessions with the Facebook Connect platform, tied to respective social profiles; their primary profile, and a second "disposable" profile. Their primary session will be associated to a social profile where they will maintain all of their social contacts, photographs, and personal information. The primary profile will be their current profile, if they already have one. The "disposable" session will be associated to a profile that will be a stripped-down version of the primary one. It may contain no personal information, social contacts, or other sensitive information that the user is not comfortable sharing with a plethora of random third-party Web sites. By default, the user's browser will keep the appropriate state, i.e., active sessions and cookies, to maintain the "disposable" session alive. As a result, when the user employs the single sign-on mechanism just to bypass the registration step in various Web sites, he will surrender only a small portion of his information or no actual information at all, as the "disposable" session with Facebook will be used. If at any point the user wishes to activate his actual profile, because he actually wants to associate his identity with a third-party Web site or online application, he is able to elevate his browser session with Facebook, i.e., by switching to the primary session from above.

In summary, the contributions of this paper are the following:

– We identify and describe an increasing threat to the users' privacy: a threat which masquerades under the convenience of a single sign-on mechanism and

gives third-party Web sites access to a user's personal information stored in social networks.
- 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 the popular single sign-on mechanism "Facebook Connect" [2] and can be easily extended to support others, such as "Sign in with Twitter" [6].
- We evaluate our implementation and show that (i) it allows users to preserve their privacy when signing on with 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 single sign-on mechanisms.

2 Background

In this section we provide some background on the OAuth protocol [4], which is the primary method for implementing single sign-on functionality across multiple Web sites. We also detail Facebook's single sign-on platform [2], which at the moment is the most popular single sing-on platform with more than 2.5 million Web sites using it [3].

2.1 OAuth Protocol

The OAuth or Open Authentication protocol [4] 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 Web 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 Web site.

2.2 Facebook Authentication

The Facebook authentication or Facebook Connect [2] is an extension to the OAuth protocol that allows third-party sites to identify users by gaining access to their Facebook identity. This is convenient for both the sites and the 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 these third-party sites that, once clicked, directs the user's browser to `http://www.facebook.com/dialog/oauth?client_id=THIRD_PARTY_SITE_ID` where the user's cookies or credentials are validated by Facebook. On successful identity validation, Facebook presents a "request for permission" dialog where the user is prompted to allow or deny the actions of the third-party Web site, i.e., social plug-in actions or access to account information. However, the user is not able to modify or regulate the third-party's request, 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 ID, that conform to what the user has just agreed upon.

2.3 Facebook Social Plug-ins

Facebook has implemented a platform of social plug-ins on top of the OAuth protocol to allow third-party sites to integrate the functionality of Facebook's social experience to their own pages [2]. In addition to authenticating a user, third-party developers are able to add "like" or "comment" Facebook buttons and forms in their site which, once clicked, update the user's Facebook profile with content from that third-party Web site, or allow the user to upload content to the site using his Facebook identity.

3 Related Work

Ardagna et al. [10] highlight the practice of Internet services requiring user information 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 [14] 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. [13] 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 [16] framework addresses threats against the privacy of social network users due to information leaked via the interaction 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 [5] is a platform supporting a federation of single sign-on providers. Its nature of operation has been described in section 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 [12] 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 [8]. 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.

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 the Web 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. Let there be noted 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 that cases she performs a "sudo" on that particular cross-site interaction with Facebook and elevate the by-default downgraded Web session.

In the description of our system we assume the use of a single sign-on mechanism such as Facebook Connect [2, 17]. However, our mechanisms can be extended to cover other single sign-on mechanisms as well.

Figure 1(b) shows the architecture of our system. To understand our approach we will first describe in Figure 1(a) how an ordinary Web browser manages session state. We see that the browser uses a default session store (Session Store [0] (default)) which 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. Indeed, in Figure 1(b) (bottom left) we have added "Session Store [1]" which stores all relevant information, including cookies, for the stripped-down Facebook session. This gives us the opportunity to enable users to surf the web

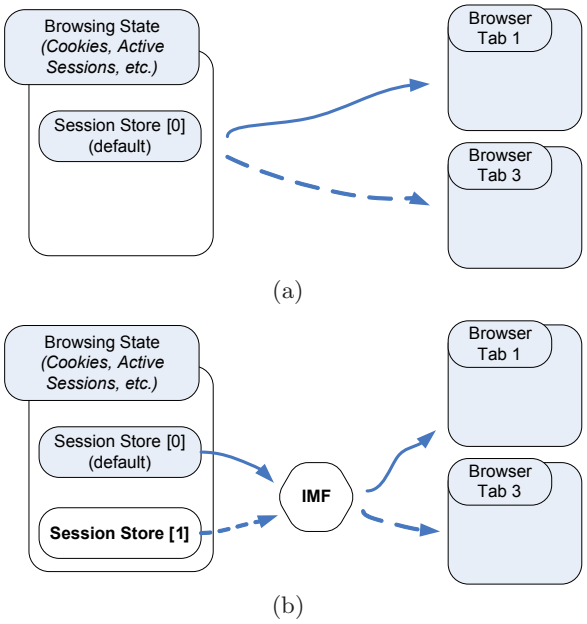

Fig. 1. Typical communication of session state to loaded pages (a), and how *SudoWeb* handles the same communication using multiple session stores (b).

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 a single sign-on platform (such as 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 towards 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, i.e., 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 2 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

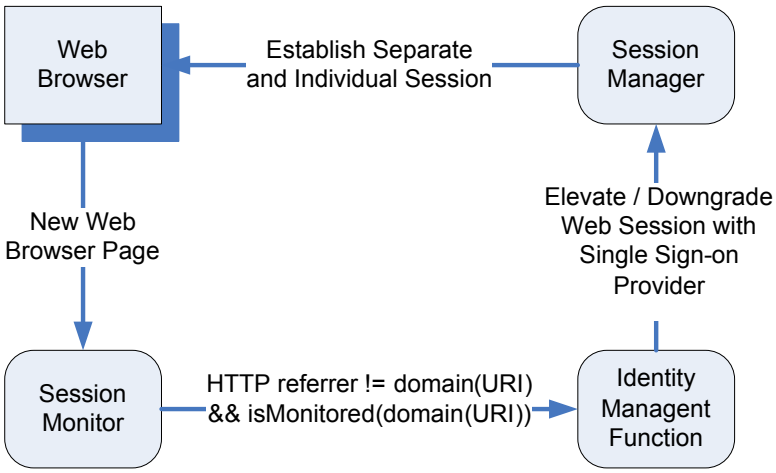

Fig. 2. *SudoWeb* extension modules.

(i.e., tab or window) is created (New Web Browser Page), the Session Monitor kicks in to find whether this is an attempt to log in a single sign-on mechanism[3]. 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) which 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 3 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 Google Chrome[4] with support for the "Facebook Connect" single sign-on mechanism. We find that, due to its popularity, our proof of concept application covers a great part of single sign-on interactions on the Web. Moreover, we implicitly support Facebook's social plug-ins, such as *share site content* or *comment on site content*, described in section 2, that require the "connect" mechanism as a first step. Our browser extension can be seamlessly configured to support a greater variety of such cross-site single sign-on interactions.

[3] *SudoWeb* keeps a list with all single sign-on domain mechanisms monitored. If such a domain is monitored the isMonitorred(domain(URI)) function returns TRUE.

[4] 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.

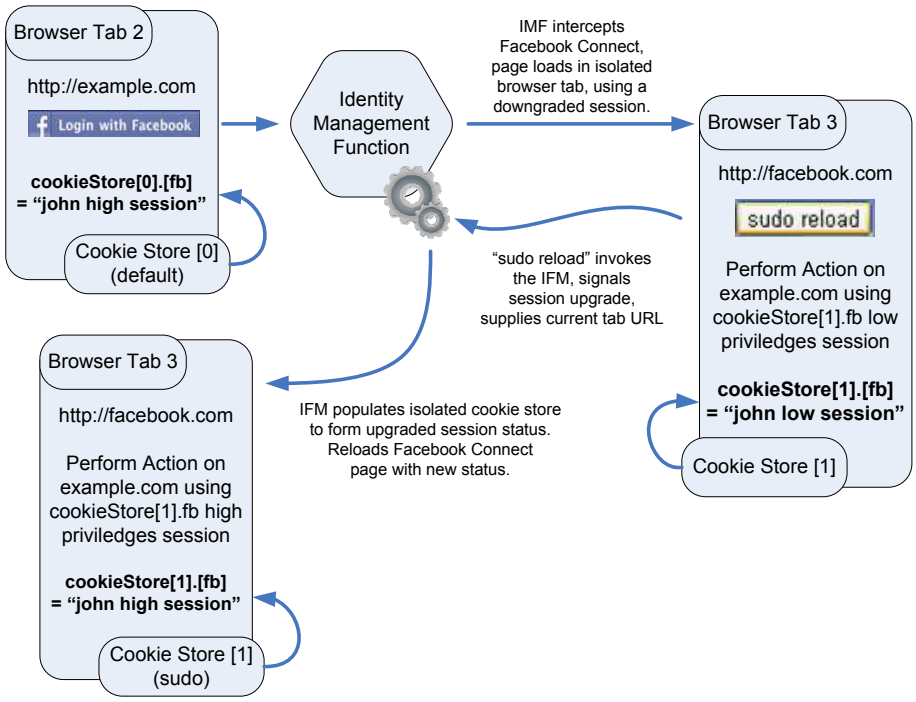

Fig. 3. Example Workflow of *SudoWeb*.

5.1 *SudoWeb* Modules

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

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 single sign-on provider (here: Facebook). Such 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) towards a single sign-on Web site (we keep a configuration file with all single sign-on 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 so that the browser receives such state that establishes a downgraded or stripped-down session with the single sign-on 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.

Session Monitor Supporting role to the IMF plays the session monitoring module. If one considers our extension as a black box, the session monitor stands at its input. It inspects new pages opening in the Web browser and looks for cases where the page URL belongs to a monitored single-sign-on provider domain (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 in several cases can be empty or even spoofed [11, 15] 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 which may remove it or spoof it.

Session Manager This 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 single sign-on (here: Facebook) account username and password, as well as his stripped-down one that is to be used for the downgraded integration with third-party Web sites. The session manager maintains in store the necessary state, e.g., cookies, required to establish the two distinct sessions with the single sign-on 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 rely on.

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 of 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

Following the use-case presented at the beginning of section 4, a user browsing the Web will eventually come across a third-party that wishes to interact with his Facebook identity via the cross-site single sign-on mechanism. As soon as the user clicks 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

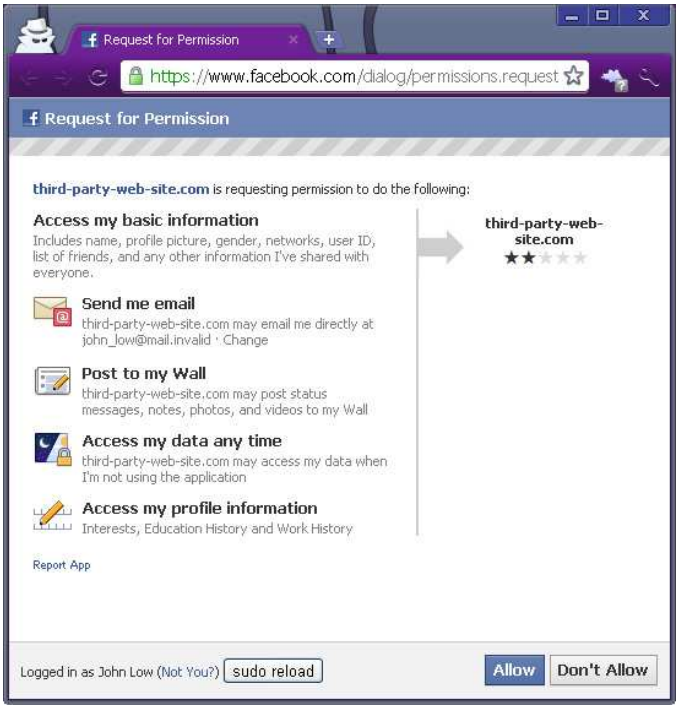

Fig. 4. Example screenshot of a Facebook "Request for Permission" page that has been invoked by fictional site `third-party-web-site.com` so that the Web user may authorize that site to access his Facebook account information. By default, a downgraded-status session is maintained with Facebook, using the appropriate state to be logged in with the disposable account "John Low." There is the option to switch to an elevated-status session via the "sudo reload" button at the bottom of the window.

and loaded in an incognito window, i.e., 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 browser window similar to figure 4. In this figure, we have used `third-party-web-site.com` as the name of the third part Web site which wants to authenticate the user using her Facebook account. We see that in addition to authenticate the user, the third-party Web site asks for permission to (i) send the user email, (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

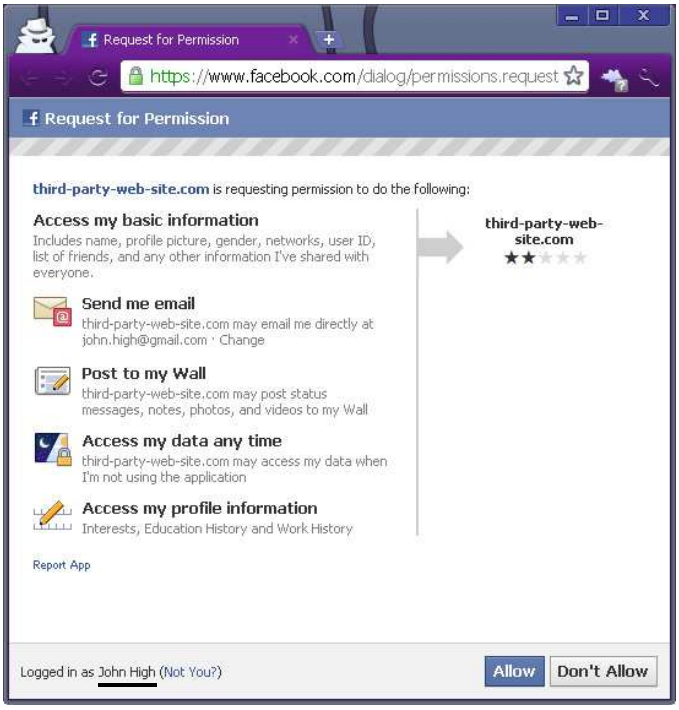

Fig. 5. Example screenshot of the previous Facebook page, after the "sudo reload" option has been selected by the Web user; the page has been reloaded using on-the-fly the necessary session state to maintain an elevated-status Facebook session using the account "John High."

not allow this access, the entire authentication session is over 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 access to his information by the third-party site, 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.

Figure 5 presents an example screenshot of the browser window the user will see if he chooses to elevate his session. One may notice at the bottom of the browser's page that 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."

Fine-grained privacy settings. Inspired by the privilege separation principles of UNIX, *SudoWeb* presents a step towards 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 single sign-on 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 single sign-on providers.

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.[5] We believe that single sign-on 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 single sign-on mechanisms should continue to function and provide their services to the users.

Terms of Use. It may seem that our approach may conflict 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 that some sign-on mechanisms have not yet caught up with the changing needs of the 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

[5] Unless the user explicitly uninstalls the application.

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, e.g., the professional one, to another social network, such as LinkedIn. We believe that sooner or later most successful single sign-on sites will catch up with the changing user needs and will adapt their terms of use to suit the users. Otherwise, the users might adopt single sign-on sites which are closer to their needs.

7 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 to the user's personal information, as well as (ii) the request to act upon a user's social network on behalf of the user, e.g., for advertisement. Unfortunately, the user can not 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 on 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 single sign-on platform, i.e., stripped from 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 framework in the Chrome browser with current support for the popular single sign-on mechanism Facebook Connect. Our results suggest that our framework is able to intercept attempts for third-party Web site authentication and handle them in a way to protect the user's privacy, while not affecting other ongoing Web sessions that the user may concurrently have.

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.

Availability

The source code of *SudoWeb* is available at `https://code.google.com/p/sudoweb/`.

References

1. BuiltWith - Facebook for Websites Usage Trends. http://trends.builtwith.com/javascript/Facebook-for-Websites.
2. Facebook for Websites. https://developers.facebook.com/docs/guides/web/.
3. Facebook Statistics. https://www.facebook.com/press/info.php?statistics.
4. OAuth. http://oauth.net/.
5. OpenID Foundation - OpenID Authentication 2.0 Specifications. http://openid.net/specs/openid-authentication-2_0.html.
6. Sign in with Twitter. http://dev.twitter.com/pages/sign_in_with_twitter.
7. Symantec Official Blog - Facebook Applications Accidentally Leaking Access to Third Parties. http://www.symantec.com/connect/blogs/facebook-applications-accidentally-leaking-access-third-parties.
8. The Chromium Projects - Multiple Profiles. http://www.chromium.org/user-experience/multi-profiles.
9. 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.
10. Claudio A. Ardagna, Sabrina De Capitani di Vimercati, Sara Foresti, Stefano Paraboschi, and Pierangela Samarati. Supporting privacy preferences in credential-based interactions. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, 2010.
11. Adam Barth, Collin Jackson, and John C. Mitchell. Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security*, 2008.
12. Arkajit Dey and Stephen Weis. PseudoID: Enhancing privacy in federated login. In *Hot Topics in Privacy Enhancing Technologies*, 2010.
13. Adrienne Felt and David Evans. Privacy protection for social networking platforms. In *Proceedings of the Workshop on Web 2.0 Security and Privacy*, 2008.
14. Wanying Luo, Qi Xie, and Urs Hengartner. Facecloak: An architecture for user privacy on social networking sites. In *Proceedings of the International Conference on Computational Science and Engineering*, 2009.
15. Mark Meiss, John Duncan, Bruno Gonçalves, José J. Ramasco, and Filippo Menczer. What's in a session: tracking individual behavior on the web. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, 2009.
16. Kapil Singh, Sumeer Bhola, and Wenke Lee. xbook: redesigning privacy control in social networking platforms. In *Proceedings of the 18th conference on USENIX security symposium*, 2009.
17. Brad Stone. Facebook aims to extend its reach across the web. *New York Times*, 2008.