OAuth for Privacy

Christopher Peplin (peplin@cmu.edu)

May 29, 2011

Abstract

The rise of Facebook, Twitter and their associated ecosystems of third-party applications was accompanied by a new authentication protocol, OAuth. The protocol enables single sign-on (SSO) as well as limited information sharing between web services at the behest of users without revealing credentials. It also provides an opportune hook for confirming privacy policies with users during authentication, but because of the difficulties of enforcement, an extended OAuth is no more or less effective than other efforts to increase user awareness. The most effective approach will be the education of web developers and information platform operators.

Contents

1 Introduction

The era of Web 2.0 may have reached buzz-word status, but the changes in Internet-enabled applications it ushered in have had an enourmous impact on the quantity and flow of personal information online. With web applications, more user activities are done entirely online, and the services are becoming increasingly interconnected. The rise of Facebook, Twitter and their associated ecosystems of third-party applications was accompanied by a new authentication protocol, OAuth [14]. OAuth enables single sign-on (SSO) as well as limited information sharing between web services at the behest of users without revealing credentials. Unlike the last major SSO contender, OpenID [16], OAuth has gained widespread adoption. For example, applications taking advantage of Facebook's OAuth access include the New York Times and Farmville.

The procedure to grant a third-party client access to data stored in a centralized personal information store, e.g. Facebook, is now greatly simplified and users are comfortable with the process. The ease and familiarity means that more people are sharing more data with more sites, and understanding the privacy policies of each party involved is even more of a daunting task.

Authorizing access to a resource through OAuth is inherently different than visiting a single website with a browser. The authorization is done under the banner of an already trusted website. The OAuth process provides an opportunity to better inform the user of the privacy they can expect for their data and to warn them when data leaves the safety of the host application. One of the hurdles for the standard privacy enhancing protocol P3P was that it required adding an additional step to a user's workflow: setting up privacy policy preferences, paying attention to a warning icon at each new website, using a different search engine, etc. Instead, P3P-style privacy negotiation can be bundled with authentication and shown to the user on the same OAuth permission page they expect. This paper describes one such possible extension to the OAuth specification.

OAuth provides an opportune hook for confirming privacy policies with users during authentication, but because of the difficulties of enforcement, an extended OAuth is no more or less effective than other efforts to increase user awareness. The most effective approach will be the education of web developers and information platform operators.

1.1 Definitions

This project intentionally uses some of the same language as the OAuth2 specification. The published definitions are provided verbatim, with some additional comments for clarity [18].

protected resource "An access-restricted resource which can be obtained using an OAuth-authenticated request." E.g. an e-mail address or phone number.

resource server "A server capable of accepting and responding to protected resource requests." E.g. Facebook or Twitter.

client "An application obtaining authorization and making protected resource requests." E.g. a third-party such as the New York Times or a Zynga.

resource owner "An entity capable of granting access to a protected resource."

end-user "A human resource owner."

token “A string representing an access authorization issued to the client. The string is usually opaque to the client. Tokens represent specific scopes and durations of access, granted by the resource owner, and enforced by the resource server and authorization servers.”

access token “A token used by the client to make authenticated requests on behalf of the resource owner.”

authorization server “A server capable of issuing tokens after successfully authenticating the resource owner and obtaining authorization. The authorization server may be the same server as the resource server, or a separate entity.” E.g. Facebook or Twitter, which act as both resource servers and authentication servers.

2 Web Services

A web service, loosely defined, is an application hosted on the Internet that exposes its functionality and data not only through a graphical user interface in a browser, but through a computer-readable interface (commonly referred to as an application programming interface or API). A web application with an API can be combined with other data sources by developers to create unique combinations of features or entirely new applications based on data previously collected or processed. For example, third-parties can offer additional insight into a user's social network by tapping into the connections they've already made in an application like Facebook. Such data sharing significantly lowers the barrier of entry for new applications, which can now piggyback off of the success of other platforms instead of building an entirely new user base. Users appreciate not needing to sign up for as many accounts, and developers appreciate the deferring of authentication work to other servers.

As users begin connecting the applications they use together, they have started thinking (often subconsciously) about their online identity. Users are concerned with who has access to their identity and to which elements of their personal information and activities online. Application developers continue to need a way to uniquely identify users to offer services, and of course want to share the work of authentication if possible. The interests of the two parties can often seem at odds.

2.1 Identity Management

Identity management is the process or system by which users control the contents of and access to the pieces of their digital identity. There are two approaches to identity management - user-centric and organization-centric.

Organization-centric management is the approach taken by many companies today, where user data is increasingly linked with a unique identifier to try and create relationships between an individual's accounts in different business entities.

User-centric management gives the individual more control. This system avoids using globally unique identifiers for users, instead creating service-specific identifiers that uniquely identify a user within an area but are insufficient for cross-referencing between databases. In some proposed systems, access control resides in a "permission hub," where a user identifies and authenticates, and third-party clients connect to request access to various resources [9]. While not conceived as such, OAuth's capabilities and usage patterns are in some ways an embodiment of this idea. Users are consolidating their personal information into a few large data silos (e.g. Facebook), using OAuth to control access.

2.2 Personal Data Stores

A further evolution of user-centric identity management is the Personal Data Store (PDS). A PDS is a server — either a user's own computer or a hosted solution – that stores personal information and releases it as needed to third-parties with the user's consent. It combines personal information storage, authentication, validation and permissions into one system [10]. This idea shares many features with a "permission hub," and thus the existing social networks.

The PDS highlights some security and privacy concerns with data consolidation, especially considering that previously distributed, uncorrelated information is now centralized in a single server and thus a single, vulnerable target. An analysis of the benefits and risks of this approach to identity management is beyond the scope of this project. However, considering the success of Facebook (arguably a proprietary PDS), it is reasonable to assume that for many, these concerns aren't a top priority or that the benefits of such a system outweigh the risks.

The goal of this project is to find a way to provide a basic level of privacy protection and notification to users of these systems while minimizing any additional burden on users and software developers. The success of these so-called “software as a service” applications depends on the availability of a secure but extremely simple method of authenticating and sharing user information between services. OAuth fostered innovation by lowering the barrier of entry for developers to link data and for users to try out new tools, and any additional privacy controls cannot stand in the way of this feature without risking alienating the market [1].

2.3 Personal Value in Linking Data

Beyond single sign-on, OAuth was the first user-friendly protocol for linking accounts together for the purposes of sharing data. The process is initiated by the user, so there is likely some apparent benefit for them to do so. This is not a situation where the resource server is offering user data to third-parties for profit or marketing. The rate of user adoption indicates there is great value to consumers in being able to link their data between services.

Conversely, OpenID has had limited success in gaining popular traction. OAuth and OpenID undoubtedly have different goals. OpenID providers serve primarily for identity management - the user’s OpenID URL is their identity online. OAuth, oppositely, was created to solve security concerns when delegating access to a user’s account with a specific service. Instead of providing a username and password, third-parties can use an access token with limited capabilities that can be revoked at will by the user without having to change their password. For better or worse, OAuth has emerged as the leading protocol in both realms. Web developers found OAuth to be easier to implement than OpenID, and so began using it as their primary SSO tool.

Debating the merits of relying on a limited number of proprietary OAuth servers (e.g. Facebook and Twitter) instead of an unrestricted set of OpenID providers is beyond the scope of this project, but again, we must recognize the higher acceptance of OAuth among users.

2.4 API Keys

Before OAuth, API keys were the most popular way of authenticating requests to or between web services. These are simple persistent access tokens generated once and appended to each HTTP request, which uniquely authenticates and authorizes a user for that request. Their implementation is generally insecure, as requests made over standard HTTP (and not SSL-encrypted HTTPS) are sent in the clear and the token is susceptible to snooping. More importantly, they are too cumbersome for end users, who won’t be bothered with copy and pasting long strings from application to application [6].

OAuth adds an extra layer of security be default (by using signed authentication requests in the original specification, and requiring SSL encryption in the next iteration), and pushes the exchange of access tokens down from the user layer to the server layer. The end result is the same - the third party has some sort of authentication token that allows them to make requests on behalf of the user - but the details of the process are transparent to the user.

2.5 Single Sign-On v.s. Access Delegation

The use of OAuth for single sign-on can unfortunately lead to data creep, as third parties take advantage of the protocol to gather data from users. Users are so comfortable with the OAuth process that they may not fully review each request for access (see figure 1 for an example of an OAuth access request). An application that only uses the protocol for authentication may be requesting access to much more data than is required to perform their business function.

The client application does not always have malicious intent. Hoping to avoid having to reacquire access in the future (and thus bother their users), and also to give themselves as much flexibility as possible, developers have a tendency to ask for complete access to a user's information even if a subset would be sufficient.

Furthermore, there is no standard granularity (either in the specification or in common practice) of control that a user has over the access granted to clients. Some data stores grant access on a per-item basis, while others offer only blunt "read" and "read and write" options. Requests for authentication have a wide range of requirements. Linking to personal information, or even identifying an individual is often unnecessary. The primary risks of broader application of authentication include covert identification, excessive use and excessive aggregation [11]. The authentication protocol should be careful not to encourage excessive data exposure.

3 State of Identification, Authorization & Privacy

3.1 Privacy Policy Troubles

The protections offered to users by first-party information stores are not often made clear during the OAuth authorization workflow. When a user submits personal information to a trusted website, they expect the site to follow its stated privacy policy. It is not clear that they can expect the same level of protection from third-parties to which they grant access. OAuth offers no opinion on the responsibility of clients to abide by the resource server's privacy policy. Furthermore, there is no requirement in the specification for the resource server to verify compliance by the client.

In practice, the protections offered to users differ on a site to site basis. This is no different than without OAuth (where users still must expect different privacy policies on different websites), but the addition of almost trivial data sharing between companies stacks the cards against users. One may say that granting access via OAuth is sufficient consent, but the current user interfaces do not sufficiently explain the risks and conditions. The result is a clean, simple interface for sharing information which encourages users to share more and give up privacy in exchange for promised improvements in service. This protocol could be more mindful of users without significantly complicating the process.

3.2 Pragmatic Privacy Enhancement

Ideas for an overhaul of identity management have been brewing over a period of years, while users and developers are moving ahead with whatever technology is most accessible. It is in the interest of all parties to make smaller, incremental improvements to existing technologies to improve user privacy instead of focusing soley on ideal, complete systems.

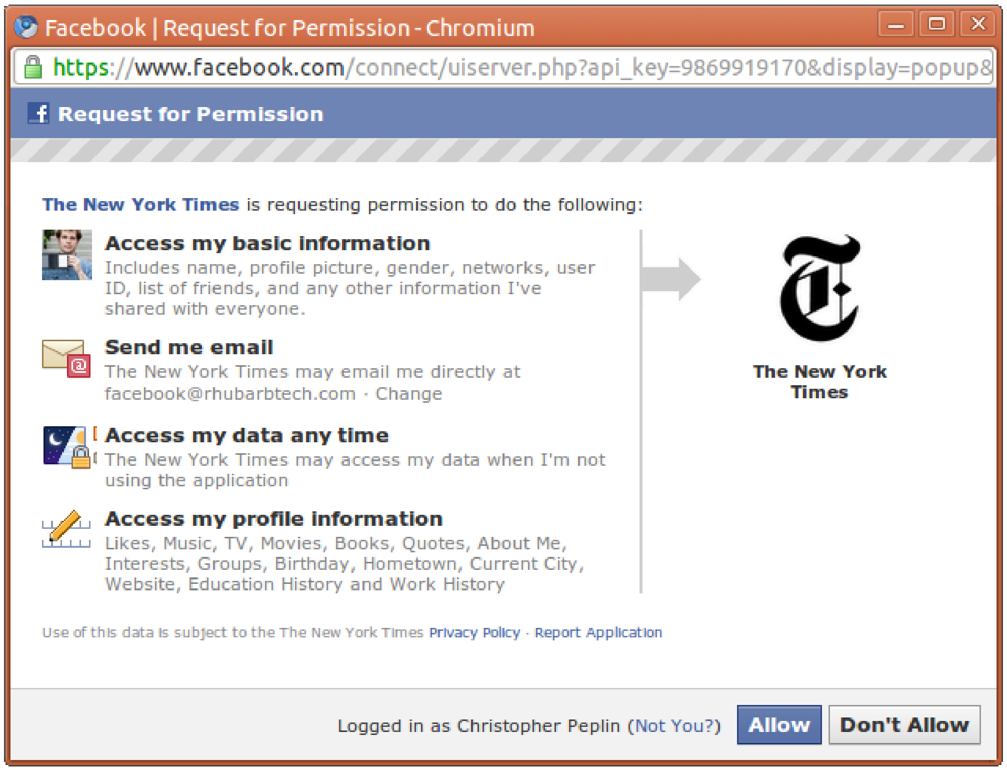
In a keynote address at the Identity 2.0 conference [8], Dick Hardt questioned which sector will spearhead such an identity overhaul. The success of OAuth proves the power of small companies and individual developers in shaping the technologies used online. Some of the most technically up-to-date, standards-compliant and accessible websites are from these small players, not the government, banks or large corporations (who are in fact notoriously behind the curve online). Many of the disruptive software technologies of the past five years have started as grassroots efforts led by developers and not the result of any corporate-backed strategy.

3.3 In Practice

There are many projects in the identity management space that attempt (or have the capability) to integrate at least one aspect of privacy control with single sign-on — fine-grained information release, permissions delegation, personal information storage, etc. OAuth is a slim protocol that has only one parameter specifically aimed at access control — `scope`.

When a client makes a request for a resource on behalf of the resource owner, it can optionally set the value of the `scope` parameter. This parameter is a space separated, unordered list of values that describes the types of data or permissions this application is requesting. The OAuth specification leaves the details of the possible values up to the authorizing and resource servers, meaning that each OAuth provider has a different set of permitted scope values and valid combinations. For example, Facebook's implementation of the OAuth2 draft describes a set of "extended permissions" [5] that can be requested via the `scope` parameter. MySpace made their own proprietary modifications to the original OAuth protocol to implement a similar feature (see figure 2) [13].

Typically, the user will see a human-readable description of the scope requested on the authorization screen (see figure 1). If they accept, the scope is stored alongside the access token that is generated — together, a contract between the two applications. The client's subsequent requests for protected resources with this access token can only access the data or perform the actions described by the stored scope. There is no way for the third-party to access a resource to which the user did not specifically grant access (unless there are security holes in the resource server).


Facebook | Request for Permission - Chromium

https://www.facebook.com/connect/uiserver.php?api_key=9869919170&display=popup&

Request for Permission

The New York Times is requesting permission to do the following:

Access my basic information
Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've shared with everyone.

Send me email
The New York Times may email me directly at facebook@rhubarbtech.com · Change

Access my data any time
The New York Times may access my data when I'm not using the application

Access my profile information
Likes, Music, TV, Movies, Books, Quotes, About Me, Interests, Groups, Birthday, Hometown, Current City, Website, Education History and Work History

The New York Times

Use of this data is subject to the The New York Times Privacy Policy · Report Application

Logged in as Christopher Peplin (Not You?) Allow Don't Allow

Figure 1: Screenshot of OAuth access grant for the New York Times Facebook application. Facebook informs users of the resources requested by the application, but does not allow fine grained control. The page includes a subdued notice that use of the data is subject to the application's privacy policy, not Facebook's. [12]

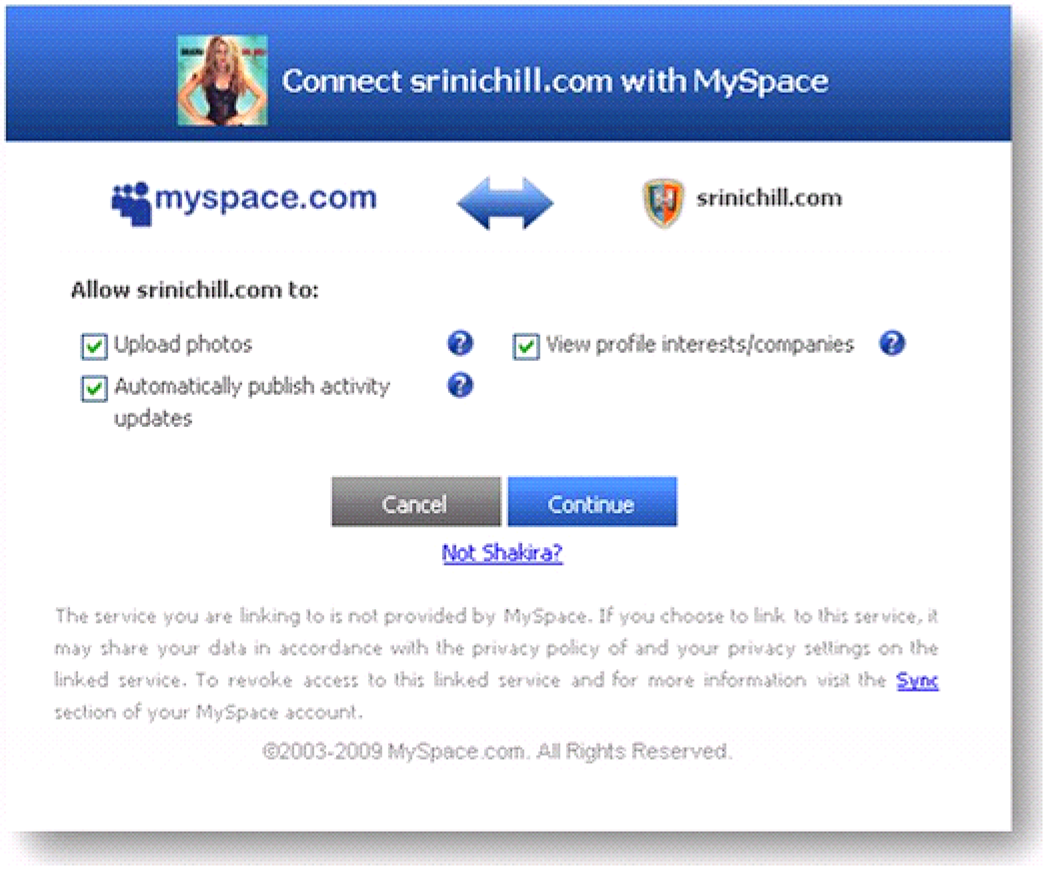
In common practice, the scope describes only what can be accessed, not what can be done with the information after it has been shared. There is no notice for a user that by granting access to their e-mail address (stored with the content provider), they may be unintentionally providing it to advertisers as well.

Two examples of OAuth authorization pages, Facebook (figure 1) and MySpace (figure 2), illustrate the lack of standardization for informing users of data exposure. MySpace includes a descriptive notice in small, light gray text at the bottom of the screen:

> "The service you are linking to is not provided by MySpace. If you choose to link to this service, it may share your data in accordance with the privacy policy of and your privacy settings on the linked service" [13].

It also includes a note about and a direct link to the account page for revoking access for previously authorized applications:

> "To revoke access to this linked service and for more information visit the Sync section of your MySpace account" [13].


Connect srinichill.com with MySpace
myspace.com
srinichill.com
Allow srinichill.com to:
Upload photos
Automatically publish activity updates
View profile interests/companies
Cancel
Continue
Not Shakira?

The service you are linking to is not provided by MySpace. If you choose to link to this service, it may share your data in accordance with the privacy policy of and your privacy settings on the linked service. To revoke access to this linked service and for more information visit the Sync section of your MySpace account.

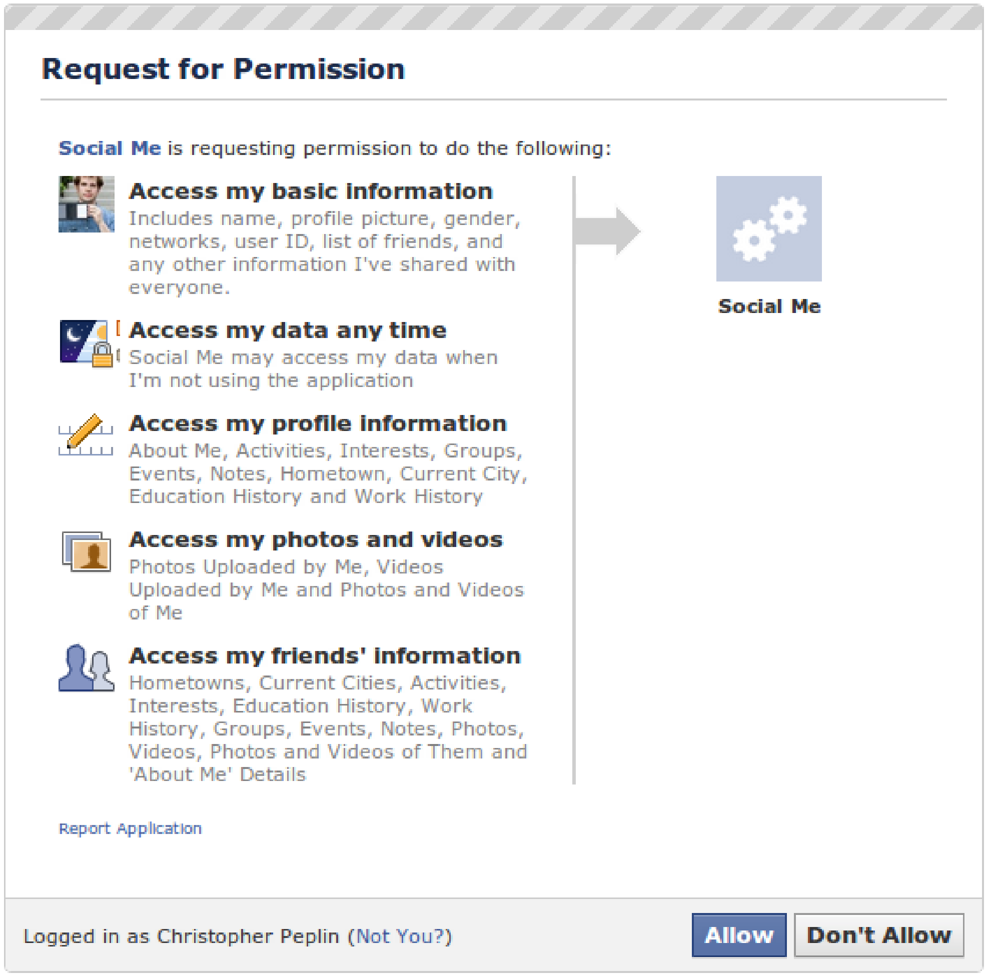
©2003-2009 MySpace.com. All Rights Reserved.

Figure 2: Screenshot of extended permissions on MySpace OAuth access grant page. MySpace added proprietary permissions parameters to the original OAuth specification [15], as well as OpenID. Users can choose whether or not to grant permission for each resource individually with check boxes. Similar to Facebook, the page includes a subdued (but more descriptive) notice that the shared data is subject to a new privacy policy. [13]

The authorization page for Facebook applications provides much less information (see figure 1). If the client application provided one, Facebook will display a link to a privacy policy with a note about possible extended use. In the case of the New York Times Facebook client:

> "Use of this data is subject to the The New York Times Privacy Policy" [12].

Again, the text is small, gray and likely to be missed by users. Compared to MySpace, this version at least provides a link to the client application's website. Disturbingly, however, if the application omits a privacy policy link from their application settings, not only is no link displayed but there is no longer any mention of possible extended data use (see figure 3). In this case, it would be reasonable (but dangerously incorrect) for a user to assume that their data is still covered by the Facebook privacy policy, and at no greater risk of exposure after authorizing the application for access.



Request for Permission

Social Me is requesting permission to do the following:

Access my basic information
Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've shared with everyone.

Access my data any time
Social Me may access my data when I'm not using the application

Access my profile information
About Me, Activities, Interests, Groups, Events, Notes, Hometown, Current City, Education History and Work History

Access my photos and videos
Photos Uploaded by Me, Videos Uploaded by Me and Photos and Videos of Me

Access my friends' information
Hometowns, Current Cities, Activities, Interests, Education History, Work History, Groups, Events, Notes, Photos, Videos, Photos and Videos of Them and 'About Me' Details

Figure 3: Screenshot of OAuth access grant for a Facebook application without a privacy policy. If the application requesting access does not specify a privacy policy, not only does the user not get a link to the application's website, but there is no notice that the data is no longer protected by Facebook's privacy policy. [12]

This is especially important since once a user's data is transferred, it could be stored indefinitely on the third-party servers. By granting OAuth access to one's Facebook profile, for example, their entire social network history could be replicated on an insecure server with no policy for data protection or breach handling.

Most application developers implicitly agree to platform policies written by the resource servers when an application is first created. The policies range in their limitations, but for example, Facebook's developer agreement requires that data not be sold to advertisers. The privacy policy and develop policy are typically not in the same location on the resource server's website, and end-users are not expected to navigate to the developer area.

4 Proposed OAuth Extension

OAuth is an interesting candidate for integrating privacy controls because it is an open standard and in the set of popular authentication systems, a relatively simple protocol. OAuth is also a standard in flux - the next version of the standard, OAuth2 [18], is current being drafted. The draft is already implemented by Facebook, while the first version [15] is in use elsewhere. The standard is at a late enough stage that such privacy extensions will not likely be incorporated, but the activity does suggest that the market for an authentication standard is active and willing to adapt quickly.

No existing authentication system with wide deployment has ever met all of the criteria for a complete identity management system. Rather than trying to build a complete system from scratch, or even modify an existing one to cover all areas, OAuth should be extended only in ways that are most natural. One problem with privacy enhancing technologies is that they typically add to or change the workflow of a user. This approach instead augments something they are already used to doing (OAuth) with privacy controls.

The proposed extension adds additional user control of and consent to information release and an element of minimal disclosure (pseudonyms). The goal of the extension is not to implement all aspects of identity management, as others have tried in the past, but to embolden OAuth to become a partner in a mixed "identity metasystem" [2] (discussed further in 6.4). Compared to other existing or proposed systems, the extended OAuth specification does not attempt to be as feature complete or secure. It represents a pragmatic approach to identity management, and attempt to create an incremental improvement on the way to a better system.

Viewing an OAuth access request as a pseudo-P3P policy [4], the protocol is current missing the "usage" section which would define the intended use of the data. OAuth can be combined with P3P-style privacy summaries to allow users to simultaneously authenticate and approve privacy policies. The extension registers a `usage` parameter (similar to `scope`) that uses P3P compact policies to describe how the information will be used. Additionally, the specification recommends that only a pseudonym is exposed to the client by default, unless more detailed identification is specifically requested.

Parameter Name `usage`

Parameter Usage Location The end-user authorization endpoint request, the end-user authorization endpoint response, the token endpoint request, the token endpoint response, and the "WWW-Authenticate" header field.

4.1 Access Grant

When a client is requesting an access grant (the process of prompting a user to grant permissions) they can optionally provide the `usage` parameter.

`usage` The intended use of the data in the scope of the access request expressed as a P3P compact policy. This parameter is optional.

Errors `invalid_usage` The requested usage is invalid, unknown, or malformed.

4.2 Access Token

After the access grant, the client can request an access token.

4.2.1 Request

`usage` The intended use of the data in the scope of the access request expressed as a P3P compact policy. If the access grant being used already represents an approved usage, the requested usage MUST be equal or lesser than the usage previously granted. This parameter is optional.

4.2.2 Response

`usage` The allowed use of the data in the scope of the access request expressed as a P3P compact policy. The authorization server SHOULD include the parameter if the requested usage is different from the one requested by the client. This parameter is optional.

Errors `invalid_usage` The requested usage is invalid, unknown, malformed, or exceeds the previously granted usage.

4.3 Sample P3P Compact Policy

A social network user may have the following privacy preferences, expressed and stored as a P3P compact policy:

```
ALL DSP CURa ADMa DEVa IVAa IVDa OUR NOR ONL DEM CNT PRE
```

This (optimistic) policy will match applications that use the data requested only for completing the current action (e.g. searching a friends list), individual analysis or individual decision making (e.g. personalized service based on a social network profile). The only recipient of the data can be the client application itself, and the data cannot be retained beyond an active user session (meaning that the application must re-request the information it requires from the resource server each time the user logs in). The user must be able to access all information stored with the application about themselves. Finally, the application must have a dispute resolution plan.

If the user begins the authorization process for an application with the following non-compliant policy, they will be warned of the specific differences:

```
ALL DSP CURa ADMa DEVa IVAa IVDa TEL OUR NOR ONL DEM CNT PRE
```

In this case, the application plans to use the data for telemarketing - something the user's privacy preferences do not allow. Technically, P3P compact policies are only intended for use with HTTP cookies. Using them here is a slight stretch of the P3P specification, but it reflects their spirit. One limitation of compact policies (addressed in the P3P 1.1 draft specification [19]) is the lack of granularity - the usage described in the policy applies to all data collected from the user. The 1.1 specification introduced compact statements, which group together a set of compact policy elements to describe one or more types of data. This allows the application or user to specify different intended usage for each type of data mentioned in the `scope` parameter, a reasonable request for both parties when considering the wide range of information shared via OAuth (from first name to geotagged photographs).

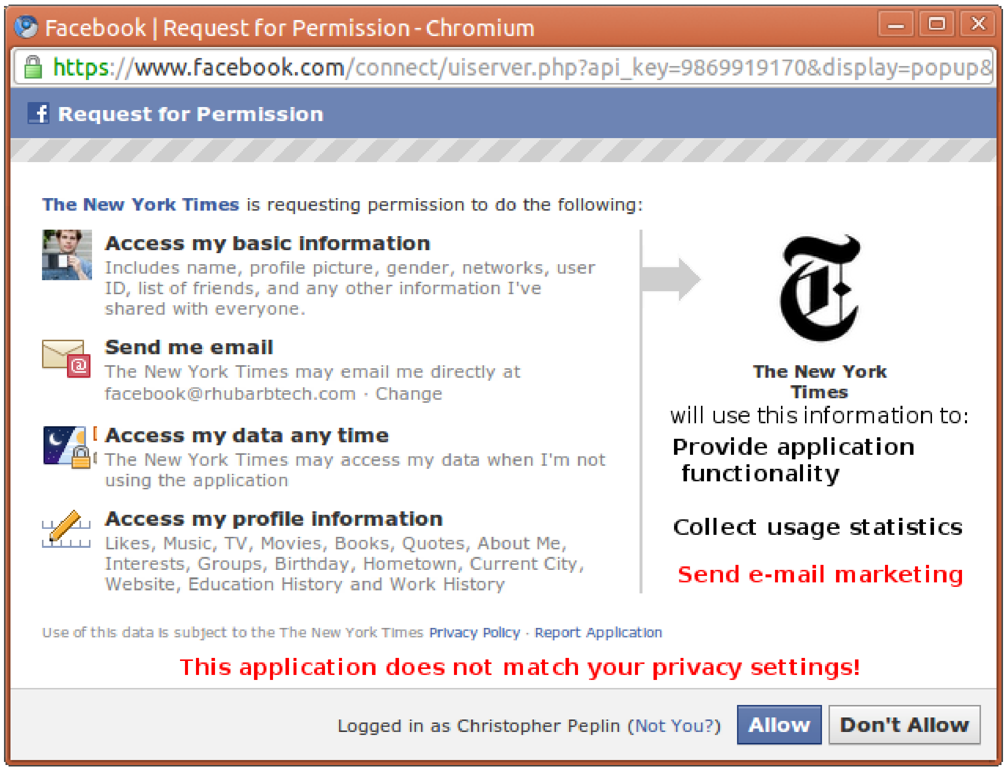
5 Use Cases

5.1 No Privacy Settings

If an authorization server does not allow users to predefine their minimum privacy requirements or the user does not have any set, the site must assume the highest level of privacy. The user should be prompted on each access grant to confirm the privacy settings manually.

5.2 Insufficiently Limited Use

If a user has their minimum privacy preferences set at the authorization server and the client is requesting usage beyond what is allowed by those preferences, the user should be shown a prominent warning and a description of the specific discrepancies. The user should be able to manually grant access even in this case.


Facebook | Request for Permission - Chromium

https://www.facebook.com/connect/uiserver.php?api_key=9869919170&display=popup&

Request for Permission

The New York Times is requesting permission to do the following:

Access my basic information
Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've shared with everyone.

Send me email
The New York Times may email me directly at facebook@rhubarbtech.com · Change

Access my data any time
The New York Times may access my data when I'm not using the application

Access my profile information
Likes, Music, TV, Movies, Books, Quotes, About Me, Interests, Groups, Birthday, Hometown, Current City, Website, Education History and Work History

The New York Times
will use this information to:
Provide application functionality
Collect usage statistics
Send e-mail marketing

Use of this data is subject to the The New York Times Privacy Policy · Report Application

This application does not match your privacy settings!

Logged in as Christopher Peplin (Not You?) Allow Don't Allow

Figure 4: Mockup of extended OAuth access page with a privacy policy mismatch. If the user's privacy settings do not match the usage requested by the application, they should be warned but allowed to proceed.

5.3 Sufficiently Limited Use

If a user has their minimum privacy preferences set at the authorization server and the client is requesting usage less than or equal to what is allowed by those preferences, the user not be shown anything specific regarding the privacy settings (beyond perhaps a small icon indicating that there are no issues). Since the majority of cases will likely fall into this use case, users should not be constantly bothered with notifications that the privacy settings are acceptable. They should only be notified when there is an issue. In this case, the user just needs to decide if they wish to grant access to the specific application, confident that the third party's privacy policy matches their own personal privacy requirements.

5.4 Usage Upgrade

If a client has an access token for a user, but wishes to expand the usage of their personal information, they must reacquire the token from the authorization server either by prompting the user out-of-band (e.g. e-mail) of their new desired usage or by requiring the user's permission at the next application launch. They must reacquire the token with the new usage settings, and they must be properly stored with the authorization server.

5.5 Policy Violation

There is no technical way to enforce a usage policy. The authorization server must be vigilant in confirming the practices of their permitted client applications, and follow up with complaints from their customers. In the case of a breach of policy, all users of that application (easy to find from the content provider's access token registry) must be notified and given the option to continue or terminate their relationship with that client.

6 Alternatives

There are numerous alternative approaches to integrating privacy with authentication and identity management. This section describes some of the efforts of other projects, as well as ideas considered and dropped in favor of the OAuth extension described in this paper.

6.1 Web2ID

Current single sign-on solutions rely on a trusted, centralized identity provider to authenticate users at various websites. This violates the user's privacy because the websites they authenticate with are exposed to the identity provider. One example of a decentralized single-sign on system is Web2ID, an SSO framework that uses public & private key encryption in place of an identity provider [21].

Tailored specifically for in-browser web service mashups, Web2ID avoids some of the problems associated with the redirects and page refreshes required by OAuth. The added complexity is of debatable value for end-users, however. Web2ID also introduces a broader identity management framework (closer to OpenID) and is not backwards compatible with any existing protocols.

6.2 User-centric Federated SSO

Centralized SSO also violates user privacy by allowing identity managers and service providers to exchange and link personal information and web traffic. The User-centric Federated Single Sign-On System (UFed) adopts the principles of user-centric identity management to protect privacy. UFed relies on existing protocols, but not the most widely used ones. It also sacrifices simplicity for security, something which many service providers do not require and which developers may resent [17].

6.3 P3P

The existing machine-readable privacy protocol, P3P, could be used without modification during the authentication process for clients. The primary use case for P3P requires the user's browser or other end-user application acting as a user-agent to access a website's P3P policy. Instead, the authorization server could act as the user agent and communicate directly with the client application. Users no longer need to install any additional software for policy checking and the rollout can be a resource server-led effort.

Short of implementing dynamic per-user full P3P policies, the authorization server and client could use standard P3P compact policies in their HTTP headers when requesting token access. The workflow would be identical to that described in section 4, but with the policies sent via HTTP headers instead of URL parameters.

This approach would require the modification of neither the P3P or OAuth specifications, so can be implemented by any resource server immediately. That said, without being explicitly described in a standard protocol, it must rely on becoming common practice through other means. Another problem is that the definition of the `scope` and `usage` fields would be less similar in the code. For example, typical open-source OAuth libraries accept URLs, user IDs and access tokens as parameters. This approach would require them to also accept the HTTP request itself, to access the P3P headers. There is an increased chance that the scope and usage would fall out of sync as their data structures diverge.

6.4 Identity Metasystem

Personal identity on the Internet is based on a fractured landscape of incompatible systems and has seemingly been on the verge of its next phase (some say "Identity 2.0" [8]) for multiple years. No single system has been created that completely satisfies developers and users alike, and those that have come close never saw widespread deployment. Identity advocates propose an "identity metasystem" [2] that combines the many existing implementations of identity management into a unified platform.

The "identity metasystem" attempts to unify the interface for online authentication for both developers and consumers. The task is not impossible, as such abstraction layers have been created for other areas of computing (e.g. video, networking). The inspiration for the system is the Laws of Identity [2], a set of principles for identity management set collaboratively by identity advocates. They cover user consent, disclosure, information user justification and usability.

The most promising of the alternatives, it is also one of the biggest. This system may eventually mitigate the privacy concerns described in section 3.1, but widespread deployment is remote compared to the availability of OAuth today.

7 Conclusion

Despite their similarity in implementation, there is a problematic difference between the proposed `usage` parameter and its counterpart `scope`. If a protected resource is not in the scope of an access token (e.g. e-mail address), there is no technical way for the third party to access that data. If the application doesn't abide by the policy they were granted by the `usage` parameter, there is no technical recourse. There is no way for the user to detect that the information is being misused, and no way to revoke access once it has been transferred off of the resource server. This fact makes the extended OAuth specification no better and no worse than existing privacy policy tools when it comes to protecting users. What it does provide is better notification and a clearer contractual agreement between parties.

In the end, any new identity system has to be sold first to developers. Developers will accept and implement a relatively complicated identification protocol if and only if the information behind the authentication wall is of sufficient value. Despite complaining loudly, developers using the Twitter API all migrated from Basic Authentication to OAuth because the API was compelling enough to do so. Other service providers with a lower uptake in OAuth usage suffer from a lack of quality offerings [7].

Without much trouble, OAuth can be extended to incorporate existing approaches to communicating privacy preferences. Unfortunately, it is susceptible to the same issues - namely a lack of enforcement and thus a lack of incentive for client compliance. Unless the resource servers required it (unlikely), developers would be unlikely to adapt the privacy extension.

7.1 Recommendations

OAuth is not a traditional single sign-on solution in that the identity has stronger ties to the identity provider [3]. The resource servers & identity providers should take advantage of this close relationship to make a safer user experience. Within the bounds of the existing OAuth2 specification, the follow recommendations would improve user notification and expand awareness of the risks of data exposure:

- In the interest of their customers, resource servers are encouraged to hold their client applications to the terms of the host application's privacy policy. To maintain the faith of their users, they must aggressively pursue those who violate it.

- Resource servers should hold third-party applications to higher standards when granting access to the ecosystem. The requirements at registration for Facebook and Twitter clients are minimal, with little verification of the legitimacy of any proposed application. This has certainly helped with the rapid expansion of applications, but at the cost of user security and privacy.

- Authorization servers are encouraged to support only SSO by default. A user's unique identifier should be considered a protected resource, and a pseudonym should be provided in its absence. For example, with a blank `scope` field, Facebook applications can currently access "all public data in a user's profile, including her name, profile picture, gender, and friends" [5] as well as the user's unique identifier (i.e. Facebook ID). For simple single sign-on, absolutely none of this is required - only a valid, permission-less access token that sufficiently identifies the user. Clients should be required to explicitly request each protected resource in the `scope` parameter beyond this pseudonym, to make it clear to users and developers what information is accessible to each party.

- Users should be allowed to modify the valid lifetime of access tokens. They should be able to to grant single use or other time-limited access tokens instead of the default permanent ones [20].

References

[1] Wang Bin et al. "Open Identity Management Framework for SaaS Ecosystem." In: *ICEBE '09: Proceedings of the 2009 IEEE International Conference on e-Business Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 512–517. ISBN: 978-0-7695-3842-6. DOI: `http://dx.doi.org/10.1109/ICEBE.2009.82`.

[2] Kim Cameron. *The Laws of Identity*. Kim Cameron's Identity Weblog. Jan. 2006. URL: `http://www.identityblog.com/stories/2004/12/09/thelaws.html`.

[3] P.J. Connolly. *OAuth Is the New Hotness in Identity Management*. eWeek. Apr. 24, 2010. URL: `http://www.eweek.com/c/a/Security/OAuth-Is-the-New-Hotness-In-Identity-Management-572745/`.

[4] Lorrie Faith Cranor. *Web Privacy with P3P*. Beijing, China: O'Reilly, 2002.

[5] *Extended Permissions*. Facebook. Oct. 10, 2010. URL: `http://developers.facebook.com/docs/authentication/permissions`.

[6] Stephen Farrell. "API Keys to the Kingdom." In: *IEEE Internet Computing* 13.5 (2009), pp. 91–93. ISSN: 1089-7801. DOI: `http://dx.doi.org/10.1109/MIC.2009.100`.

[7] Eran Hammer-Lahav. *Twitter a Hot Princess, Google an Empty Castle*. Hueniverse. Sept. 15, 2010. URL: `http://hueniverse.com/2010/09/twitter-a-hot-princess-google-an-empty-castle/`.

[8] Dick Hardt. *Identity 2.0 Keynote*. Aug. 4, 2005. URL: `http://www.youtube.com/watch?gl=US&hl=uk&v=RrpajcAgR1E`.

[9] John Harrison and Pete Bramhall. *New approaches to identity management and privacy*. Dec. 2007. URL: `http://www.ico.gov.uk/upload/documents/library/data_protection/detailed_specialist_guides/edentity_hp_idm_paper_for_web.pdf`.

[10] *Higgins - Personal Data Store Overview*. Eclipse Foundation. Oct. 10, 2010. URL: `http://wiki.eclipse.org/Personal_Data_Store_Overview`.

[11] Stephen T. Kent and Lynette I. Millett. *Who Goes There?: Authentication Through the Lens of Privacy*. Washington, D.C.: National Academies Press, 2003. Chap. 1,2.

[12] *"Log In With Facebook" Pop-up Screenshot*. Facebook. Oct. 10, 2010. URL: `http://facebook.com`.

[13] MySpace. *Extended Permissions for Offsite Apps*. Oct. 10, 2010. URL: `http://wiki.developer.myspace.com/index.php?title=Extended_Permissions`.

[14] *OAuth Community Site*. Oct. 10, 2010. URL: `http://oauth.net`.

[15] *OAuth Core 1.0*. Oct. 10, 2010. URL: `http://oauth.net/core/1.0/`.

[16] *OpenID Foundation*. Oct. 10, 2010. URL: `http://openid.net/`.

[17] Suriadi Suriadi, Ernest Foo, and Audun Josang. "A User-centric Federated Single Sign-on System." In: *NPC '07: Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 99–106. ISBN: 0-7695-2943-7.

[18] *The OAuth 2.0 Protocol (Draft)*. Oct. 10, 2010. URL: `http://tools.ietf.org/html/draft-ietf-oauth-v2-10`.

[19] *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*. Nov. 2006. URL: `http://www.w3.org/TR/P3P11`.

[20] Yuan Wang and Julita Vassileva. “A User-Centric Authentication and Privacy Control Mechanism for User Model Interoperability in Social Networking Sites.” In: *International Workshop on Adaptation and Personalization for Web 2.0*. Trento, Italy, 2009, pp. 110–119. URL: `http://sole.dimi.uniud.it/~antonina.dattolo/papers/2009/book/Dattolo-apweb2009.pdf#page=120`.

[21] Saman Zarandioon, Danfeng Yao, and Vinod Ganapathy. “Privacy-aware identity management for client-side mashup applications.” In: *DIM ’09: Proceedings of the 5th ACM workshop on Digital identity management*. Chicago, Illinois, USA: ACM, 2009, pp. 21–30. ISBN: 978-1-60558-786-8. DOI: `http://doi.acm.org/10.1145/1655028.1655036`.