ROAuth: Recommendation based open authorization

Article · July 2011

DOI: 10.1145/2078827.2078842

3 authors, including:

Mohamed Shehab
University of North Carolina at Charlotte
74 PUBLICATIONS **981** CITATIONS
SEE PROFILE

Christopher Hudel
University of North Carolina at Charlotte
2 PUBLICATIONS **11** CITATIONS
SEE PROFILE

ROAuth: Recommendation Based Open Authorization

Mohamed Shehab
Department of Software & Information Systems
University of North Carolina
Charlotte, NC, USA
mshehab@uncc.edu

Said Marouf
Department of Software & Information Systems
University of North Carolina
Charlotte, NC, USA
smarouf@uncc.edu

Christopher Hudel
Department of Software & Information Systems
University of North Carolina
Charlotte, NC, USA
chudel@uncc.edu

ABSTRACT

Many major online platforms such as Facebook, Google, and Twitter, provide an open Application Programming Interface which allows third party applications to access user resources. The Open Authorization protocol (OAuth) was introduced as a secure and efficient method for authorizing third party applications without releasing a user's access credentials. However, OAuth implementations don't provide the necessary fine-grained access control, nor any recommendations vis-a-vis which access control decisions are most appropriate. We propose an extension to the OAuth 2.0 authorization that enables the provisioning of fine-grained authorization recommendations to users when granting permissions to third party applications. We propose a mechanism that computes permission ratings based on a multi-criteria recommendation model which utilizes previous user decisions, and application requests to enhance the privacy of the overall site's user population. We implemented our proposed OAuth extension as a browser extension that allows users to easily configure their privacy settings at application installation time, provides recommendations on requested privacy attributes, and collects data regarding user decisions. Experiments on the collected data indicate that the proposed framework efficiently enhanced the user awareness and privacy related to third party application authorizations.

Categories and Subject Descriptors

H.4 [**Computers and Society**]: Public Policy Issues; D.2.8. [**Management of Computing and Information Systems**]: System Management

General Terms

Security, Experimentation

Keywords

Privacy, OAuth, Collaboration, Recommendations, Social networks

Symposium on Usable Privacy and Security (SOUPS) 2011, July 20–22, 2011, Pittsburgh, PA USA

1. INTRODUCTION

Open standards and third party software development have long formed a partnership that affords internet users the tools and capabilities to better manage their own identity, privacy, and confidentiality. Seeing a need for users to better know the privacy policies in force for various websites led the World Wide Web Consortium (W3C) to create an open specification titled the Platform for Privacy Preferences (P3P) and the corresponding Preference Exchange Language (APPEL) which is in-use today by various internet websites to provide, in machine readable format, a policy file specifying that site's particular privacy policies [3]. This machine readable XML file was first available to plug-in or addon software developed by third parties, such as AT&T's Privacy Bird [8] before being incorporated directly into web browser software by such manufacturers as Microsoft (Internet Explorer) and then Netscape (Navigator).

The OAuth open standard protocol is another example of an available standard created to provide users with the ability to share information and resources with third party application components of other, more primary, web applications. For example, the OAuth framework might allow for the sharing of photographs from a primary web-based photo sharing website so that a third party photo printing service may access the permitted photographs [5]. Popular within online social networks, Facebook today represents the largest single OAuth 2.0 implementation permitting a mechanism for third party web based applications to access Facebook user identity and privacy information and resources.

Third-party software developers have led charges to improve user privacy and security all on their own, using extensible frameworks available in the Chrome, Firefox, and (recently) Safari web browsers. These browser extensions protect users, for example from unwanted advertisements, malicious software installations, and compromise of user credential data. Indeed, Joshi et al [22] showcase a browser plugin that attempts to solve man-in-the-middle attacks prevalent in modern Phishing attacks.

While the partnering relationship between standards and browser-based extensions is rich in history and likely to continue, there may exist one gap that needs fulfilling. Appreciating that individual privacy preferences may be just that - individual - how can a single extension reflect the privacy preferences of a unique set of individuals? It is in this vein that we offer our novel research in providing a recommender-based model that enables users to make important privacy-based decisions at the time of third party application installation. Recommendations give users confidence in mak-

ing their decisions, especially that most privacy requests are vague and do not clearly convey the accesses requested. The decisions that users make are their own of course, but our algorithm and model provides a mechanism to *inform* them and provide recommendations based on the collaborative decisions (grant/deny) on similar privacy requests within the user's larger social network.

1.1 Contributions

Our contributions in this space include:

- A browser-based extension that intercepts the default OAuth 2.0 request flow, interprets it, and provides the user with an easy and usable interface to make decisions that provide for the protection of private identity attributes *before* application installation.
- A multi-criteria recommender model approach that provides users with rating measures on requested privacy attributes based on the collaborative effort of users who have historically made grant/deny decisions for similarly requested privacy attributes.
- A recommendation to extend the Open Authorization specification to provide an avenue through which web-browsers (through browser extensions method or otherwise) might assist users in making informed decisions regarding their full privacy attributes before the installation of a third party application.
- A user study that shows the results and effectiveness of using our proposed browser extension.

2. PROBLEM DEFINITION

The OAuth framework provides a mechanism for third-party service providers to request end-user resources from an application without releasing the user's application access credentials to the service provider. However, specific implementations may not provide the user with the necessary fine-grain access control, and does not provide any recommendations vis-a-vis which access control decisions may be the most appropriate.

An example we use throughout this paper is one of the free Facebook online video and voice calling applications available through "friendcameo.com". [1] The FriendCameo Facebook application requests the following extended permissions when a user first installs the application: access to the user's email address, ability to publish status and post messages to the user's wall, the ability to access the Facebook chat application, and the ability to enumerate the online presence status of other users (within the first user's social network).

It becomes quickly clear that several of the extended permissions, once granted, cannot realistically be revoked. For example, once users provide FriendCameo access to their email addresses, they cannot realistically remove that email address from FriendCameo's servers and databases by preventing *further* access to the information through Facebook's application and privacy settings. We find there are several user attributes that are practically irrevocable, once granted since the attributes are generally immutable (i.e.: birthday) or generally change with very little frequency (i.e.: hometown locations, religious and political views). (See Figure 1)

[1]We make no value judgement of the extended permissions requested by the example applications we present in this paper and the example applications are simply that, examples.

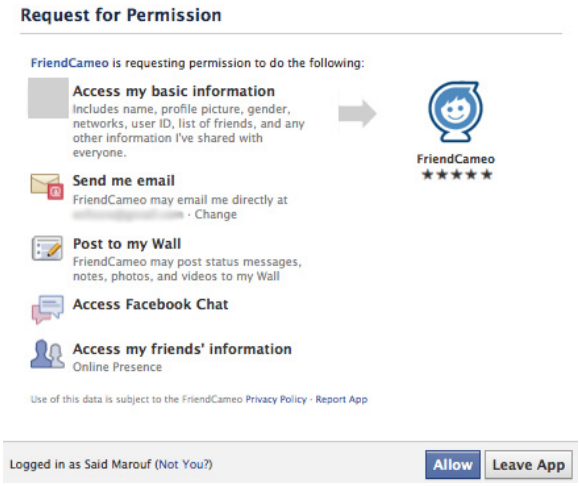


Figure 1: 3rd party application permission request

We view the permanent loss of personal attributes as only one part of the problem; should a method be devised to permit users a "last line of defense" against such information loss, how may they know best what decisions to take. Can users benefit from a community of knowledge to better inform their own decision making?

Our novel approach provides both the aforementioned "last line of defense" mechanism and a recommender model based on the decisions of all users within a system, and the previous decisions of an individual user. Later, we show how such a model might further inform individual privacy decisions through prediction.

3. PRELIMINARIES

3.1 Third Party Applications and APIs

Most of the major online platforms such as Facebook, Google, and Twitter, provide an open API which allows third party applications to directly interact with their platform. APIs provide a mechanism to read, write, or modify user information on these platforms through other third party applications on behalf of users themselves. An API comes with a set of methods, each representing a certain user interaction executed through a third party application. For example, the FriendCameo [12] Facebook application is able to post content (e.g. messages, photos) to a user's Facebook feed/wall using Facebook's */profile_id/feed* API method, where *profile_id* is the targeted Facebook user ID. It is important to note that third party applications can potentially execute any API call on behalf of a user, relying on the type and scope of permissions granted to these apps. In the previous example, the FriendCameo application could only perform the */profile_id/feed* API call given the user has granted it the "publish_stream" permission. The full set of permissions available to third party apps are defined by the online platforms, and it is up to third party applications to request the proper subset of permissions required. We believe users should have the final decision on whether to grant requested permissions or not.

3.2 OAuth Standard

With an increasing trend towards offering online web services that provide third party applications the ability to in-

teract through open APIs and access user resources, OAuth was introduced as a more secure and efficient method for authorizing third party applications [32]. Traditional authentication models such as the client-server model require third party applications to authenticate with online services using the resource owner's private credentials, typically a username and password. This requires users to present their credentials to third party applications, hence granting them broad access to all their online resources with no restrictions. A user may revoke access from a third party application by changing her credentials, but doing so subsequently revokes access from *all* third party applications that continue to use her previous credentials. These issues are amplified given the high number of third party applications that potentially get access to a user's online resources. OAuth uses an intuitive mechanism where the roles of third party applications and resource owners are separated. OAuth does not require users to share their private credentials with third party applications; instead it issues a new set of credentials for each application. These new set of credentials are per application, and reflect a unique set of permissions to a user's online resources. In OAuth, these new credentials are represented via an *Access Token*. An Access Token is a string which denotes a certain scope of permissions granted to an application, it also denotes other attributes such as the duration the Access Token is considered valid. We are mainly interested in the scope attribute within an Access Token. Access Tokens are issued by an authorization server after the approval of the resource owner. In this paper we extend upon this authorization stage of the OAuth protocol.

When a third party application needs to access a user's protected resources, it presents its Access Token to the service provider hosting the resource (e.g. Facebook, Twitter) which in turn verifies the requested access against the scope of permissions denoted by the Access Token. For example, Alice (resource owner) on Facebook (service provider and resource host) can grant the FriendCameo Facebook application (third party application) access to her email address on her Facebook profile without ever sharing her username & password with FriendCameo. Instead, she authenticates the FriendCameo application with Facebook (authorization server) which in turn provides FriendCameo with a proper Access Token that denotes permission to access Alice's email address.

OAuth provides multiple authorization flows depending on the client (third party application) type (e.g. Web servers, and native applications). In this paper we focus on the Web server authorization flow shown in figure 2 and detailed in the OAuth 2.0 specification [33]. The web server flow is typically used for third party applications that are able to interact with a user's web browser (e.g. Facebook applications). The authorization flow process consists of three parties: 1)The end-user (resource owner) at browser, 2)The web client (third party application), and 3)The authorization server (usually the service provider, e.g. Facebook). Our main focus will be on step "(A)" and "(B)" within the web server authorization flow [33]. Step "(A)" is where third party applications initiate the flow by redirecting a user's browser to the authorization server and passes along the requested scope of permissions. In step "(B)", the authorization server authenticates the end-user, and establishes her decision on whether to grant or deny the third party application's access request.

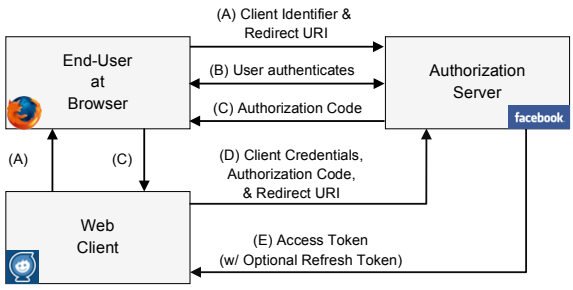


Figure 2: Web Server OAuth Flow

3.3 OAuth and User Privacy

One of the main reasons behind OAuth was to increase user privacy by separating the role of users (resource owners) from that of third party applications. OAuth uses the concept of Access Tokens, where a token denotes a set of credentials granted to third party applications by the resource owners [31]. This avoids the need for users to share their private credentials such as their username & password. It also allows users to revoke access to a specific third party application by revoking its Access Token.

OAuth 2.0 allows third party applications to request a set of permissions via the *scope* attribute, and for users to grant or deny such requests. If a user grants a third party application's request, then an Access Token (denoting the *scope*) is issued for that application, hence granting it the scope of permissions requested. The *scope* attribute represents the set of permissions requested by third party applications, and will is our main focus in this paper. In the web server authorization flow seen in figure 2, the *scope* parameter is part of the request URI that is generated by third party applications (Step "(A)" in figure 2). It is a list of space-delimited strings, each string mapped to a certain permission or access level. For example, the FriendCameo Facebook application requests permission to post to a user's Facebook feed/wall, to log in to Facebook chat, to access her email address, and to check her friend's online/offline presence. FriendCameo requests these permissions with a *scope* attribute value of "*publish_stream,xmpp_login,email,friends_online_presence*". The *scope* value becomes part 'of the OAuth request URI sent to the authorization server (Facebook's OAuth implementation uses comas rather than spaces to separate each requested permission). Step "(B)" of figure 2 is where users grant or deny the requested *scope* value.

We propose an extension to the OAuth 2.0 authorization flow regarding the Web Server client profile detailed in section 1.4.1 of the OAuth 2.0 specification [32]. Before users make their decision on the requested *scope* of permissions, we introduce a new level of awareness and control to the user via an in-house developed browser extension. The extension represents a final line of defense against third party applications and their requested permissions.

3.4 Collaborative Filtering

Recommendation systems are systems that try to assist users in evaluating and making decisions on items by providing them opinions and prediction values as a set of recommendations [35]. These set of recommendations are usually based on other people's opinions and the potential rel-

evance of items to a target user. The first recommender system Tapestry [14], followed the approach of "Collaborative Filtering", in which users collaborate towards filtering documents via their individual reactions after reading certain documents. Since then, collaborative filtering has been widely adopted and is accepted as a highly successful technique in recommender systems [25, 28, 26, 38].
In a context of access control and user privacy, items in a collaborative filtering model can be mapped to individual privacy attributes or permissions. Users make decisions on privacy attributes, i.e. grant them to third party applications or not. This is similar to other recommendation systems in which users make decisions on items, e.g. to rent a movie or not. Users have their own privacy preferences, but may benefit from the community's collaborative privacy decisions to make their own, especially if they lack the knowledge to make good privacy decisions [20]. The effect of community data on user privacy has been investigated by Besmer et al. [4], who explored the effect of community data on user behavior when configuring access control policies. Their work indicated that community data impacts user behavior, when substantial visual cues were provided. Goecks et al. [13] explored the effects of community data in the domain of firewall policy configuration and web browser cookie management. Their results also indicated that users did utilize community data in making their own decisions.

In this paper we propose a collaborative filtering model that utilizes community decisions in providing recommendations to users who install third party applications requesting access to their privacy attributes.

4. PROPOSED OAUTH FLOW

We propose an extension to the OAuth 2.0 authorization flow for Web Servers, by introducing two new modules into the flow: 1)A Permission Guide that guides users through the requested permissions, and shows them a set of recommendations on each of the requested permissions, and 2)A Recommendation Service that retrieves a set of recommendations for the requested permissions following a collaborative filtering model as explained in section 4.2. Our OAuth

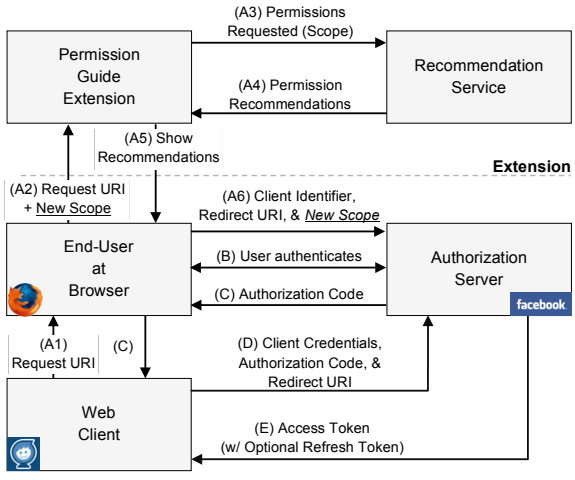


Figure 3: Proposed OAuth Flow

extension focuses on step "(A)" of the Web Server authorization flow in OAuth 2.0 [33]. We revise step "(A)" to become a six stage process as shown in figure 3 and explained in the following steps:

- (A1): The web client (third party application) redirects the browser to the end-user authorization endpoint by initiating a request URI that includes a *scope* parameter.
- (A2): The Permission Guide extension captures the scope value from the request URI and parses the requested permissions. At this step the extension allows users to choose a subset of the permissions requested.
- (A3): The Permission Guide extension requests a set of recommendations on the parsed permissions. This is achieved by passing the set of permissions to the Recommendation Service.
- (A4): The Recommendation Service returns a set of recommendations for the permissions requested by the web client.
- (A5): Using the set of returned recommendations, the extension represents the permissions with their respective recommendations in a user friendly way.
- (A6): The Permission Guide extension redirects the end-user's browser to a new request URI with the new scope ($scope'$), assuming the user chooses to modify the requested permissions.

4.1 Permission Guide

The Permission Guide is represented by a browser extension that integrates into the authorization process by capturing the *scope* parameter value within the request URI generated by a third party application. Once the scope is captured, the extension parses the requested permissions and presents them in a user friendly manner as shown in Figure 7. A readable label of each requested permission is shown to the end-user e.g. it shows "Facebook Chat" rather than "xmpp_login", and a detailed description for each permission is also shown when the user hovers over a certain permission label.

The extension also shows users a set of recommendations for the requested permissions. For each permission there is a thumbs-up and a thumbs-down recommendation value. These recommendations represent prediction values that we calculate following our model in section 4.2. These prediction values represent the likeliness of a user to grant or deny a certain permission based on her previous decisions and on the collaborative decisions of other users. Users who have not made any decisions yet, are shown recommendations based on other user decisions.

The extension allows users to customize the requested permissions by checking or unchecking individual permissions, where a checked permission is one the user wishes to grant to the third party application and an unchecked permission is one she wishes to deny access to.
Once a user decides on the permissions she wishes to grant and deny, she simply needs to click a *Set Permissions* button on the extension (blue button in Figure 7). This will trigger the extension to generate a new request URI with a new scope $scope'$, and forward the user's browser to this new request URI. $scope'$ will always be a subset of the original requested scope, i.e. $scope' \subseteq scope$.
An example $scope'$ from the FriendCameo application we saw earlier could be as follows:

$$scope' = publish\_stream$$

reflecting the user's desire to allow FriendCameo to post to her feed/wall, but deny it access to her email, Facebook chat and friend's online/offline presence. Note that using a subset of the permissions requested could potentially hinder the functionality of a third party application once installed. Investigating such consequence is out of the scope of this paper, but we include it as part of our future work.

Our Permission Guide extension also collects the user's decisions on the requested permissions, hence allows us to generate a data set of decisions to be used in our recommendation model explained in section 4.2. These decisions are uploaded to our servers once a user sets her desired permissions within the extension, i.e. clicks the *Set Permissions* button. The data uploaded to our servers includes: *app_id, requested_perms, decisions, recommendations*, where the *app_id* is the application's unique id which is assigned by the service provider (e.g. Facebook), the *requested_perms* are the scope of permissions requested by the third party application, the *decisions* are the individual user decisions (grant or deny) on each of the requested permissions, and the *recommendations* are the recommendation values at the time the user made her decision.

Our goal is to provide a simple and usable user interface for interacting with permission requests, hence increasing user awareness and providing an easy mechanism for guiding users in making their decisions.

4.2 Recommendation Model

We propose a Recommendation Service component that extends upon our Permission Guide extension. Let $\mathcal{A}$, $\mathcal{U}$ and $\mathcal{P}$ represent the set of applications, users and permissions respectively. A user $u_i \in \mathcal{U}$ can make a decision $d_i \in \{grant, deny\}$ on a permission $p_j \in \mathcal{P}$ for an application $a_k \in \mathcal{A}$. An application a_k which requests permissions $p_1, \cdots, p_m$ is mapped to a set of decisions $d_1, \cdots, d_m$ made by the user installing a_k.

Our model follows the multi-criteria recommendation model where user recommendations are calculated per criterion [25, 2]. The model utilizes the set of permissions $\mathcal{P}$ as a set of criteria, i.e. each permission $p_j \in \mathcal{P}$ represents an individual criterion within the model. The multi-criteria approach fits our model as decisions are made per permission (criteria) rather than an application as a whole. We model a user's utility for a given application with the user's decisions $d_1, \cdots, d_m$ on each individual permission $p_1, \cdots, p_m$ using Function 1.

$$D : Users \times Applications \rightarrow d_1 \times \cdots \times d_m \qquad (1)$$

Function 1 represents a user's overall decision on a certain application via the set of decisions made on each individually requested permission. That is, a user u_i makes a decision d_i on an application a_k with respect to an individual permission. For each permission p_j, there exists a matrix $\mathcal{C}_{p_j}$ representing user decisions on p_j for each application $a_k \in \mathcal{A}$, see Figure 4. A matrix entry d_i with a value of 1 denotes a user has *granted* a_k the permission p_j, whereas a 0 denotes a *deny*. Entries with "?" values denote the user is yet to make a decision on permission p_j for application a_k. Our model provides recommendations to users that guide them in making these future decisions. Applications that do not request a permission p_j have an empty entry in $\mathcal{C}_{p_j}$ and are handled properly in our implementation.

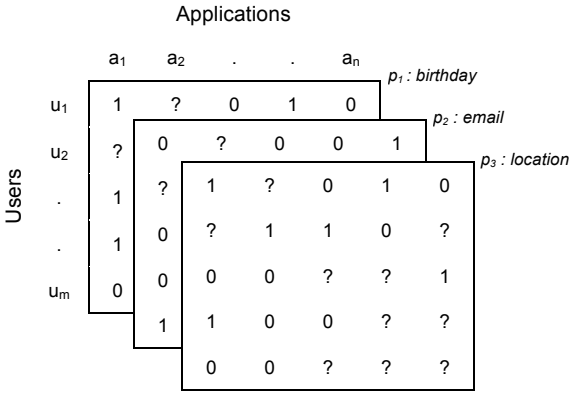


Figure 4: A three-permission (criteria) model with $\mathcal{C}_{p_1}$, $\mathcal{C}_{p_2}$, & $\mathcal{C}_{p_3}$. User decisions on applications made per-permission

For example, let $p_1 = birthday$, $p_2 = email$, and $p_3 = location$, where each represents a single criterion within a three-criteria model. Let $u_1 = Alice$ who installed application a_1 that requests access to the permissions *birthday*, *email*, and *location*. As illustrated in Figure 4, Alice has granted a_1 the permissions *birthday* and *location* ($d_1 = grant, d_3 = grant$), whereas denied *email* ($d_3 = deny$). Alice has yet to make a decision on a_2 i.e. a single decision on each requested permission $\in \{birthday, email, location\}$. Our proposed model utilizes the set of decisions for each $\mathcal{C}_{p_j}$, hence providing a recommendation that fits each criterion.

4.3 Collaborative Application Filtering

We define Pr_{p_j,a_k} as the overall probability of permission p_j being granted to application a_k. The various probabilities can be represented as a $Permissions \times Applications$ matrix as seen in Figure 5. Each entry P_{jk} in the matrix maps to the Pr_{p_j,a_k} for permission p_j and application a_k. Lower Pr values indicate less user willingness towards granting certain permissions to an application. These probability values are based on our collected data set as explained in detail in section 6. Note that these values are driven by user decisions within our framework and are frequently updated. Figure 6 shows an example set of applications $\{a_1, a_2, a_3, a_4, a_5\}$, permissions *(birthday,email,location,sms,photos)* and their corresponding Pr_{p_j,a_k} values. For example, $Pr_{32} = Pr_{location,a_2} = 0.15$, which denotes a low probability of the permission *location* being granted to application a_2 by users who installed a_2.

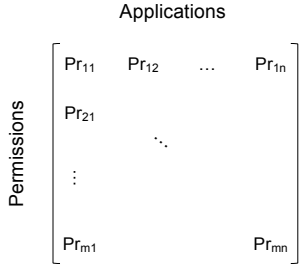


Figure 5: Permissions × Applications matrix

In our system, users collaborate towards increasing or de-

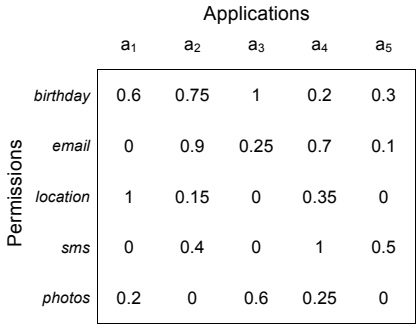


Figure 6: Example Pr_{p_j,a_k} values.

creasing the Pr_{p_j,a_k} values of various applications, hence filtering applications according to the willingness of users to grant them certain permissions. Filtering applications is based on similarity values calculated using the Pr_{p_j,a_k} of each application. We also use the similarity value to predict recommendation values within our model.

Collaborative filtering algorithms have mainly been based on one of two popular similarity measures, 1)Pearson Correlation and 2)Cosine-similarity [18, 28]. Our model measures similarities between applications using Pearson Correlation based on the $Permissions \times Applications$ matrix. We use an application-based similarity measure similar to item-based similarity in Collaborative Filtering algorithms. Item-based similarity proved to be more accurate than others such as user-based similarity [37], hence we follow a similar approach where items in our context map to applications.

Equation 2 represents our application-based similarity measure, which in terms is the Pearson correlation value between applications a_i and a_j, where $\mathcal{P}$ is the set of all permissions in our system and $\overline{Pr}_{a_i}$ is the average probability for application a_i being granted a permission in $\mathcal{P}$.

$$sim(a_i, a_j) = \frac{\sum_{\forall p \in \mathcal{P}} (Pr_{p,a_i} - \overline{Pr}_{a_i})(Pr_{p,a_j} - \overline{Pr}_{a_j})}{\sqrt{\sum_{\forall p \in \mathcal{P}} (Pr_{p,a_i} - \overline{Pr}_{a_i})^2 \sum_{\forall p \in \mathcal{P}} (Pr_{p,a_j} - \overline{Pr}_{a_j})^2}} \quad (2)$$

Applications that don't request a certain permission p_j have a Pr_{p_j,a_i} of zero. Also note that $-1 \leqslant sim(a_i, a_j) \leqslant 1$ where -1 denotes a reverse correlation between a_i and a_j, 0 denotes no correlation, and 1 denotes a exact correlation. Applications which are similar and highly correlated, are those which request a similar set of permissions, and have similar Pr_{p_j,a_k} values for each of their requested permissions. For example, if both applications a_1 & a_2 requested the same set of permissions $\{p1, p_2\}$, and they have a $Pr_{p_1,a_1} = Pr_{p_1,a_2}$ and a $Pr_{p_2,a_1} = Pr_{p_2,a_2}$, then a_1 and a_2 are considered highly correlated and their application-similarity value $sim(a_i, a_j)$ will be close to 1.

When predicting recommendation values for permissions of application a_i, we make sure they are based on a_i's "Nearest Neighbors", that is, the set of applications where $sim(a_i, a_j)$ is highest and $sim(a_i, a_j) \geqslant 0$ [35].

Prediction Model.

When a user u_i, say Alice, wants to install application a_i, we calculate a set $\mathcal{R}_{a_i}$, where $r_{u_i,p_j} \in \mathcal{R}_{a_i}$ is a prediction value for permission p_j requested by a_i. $r_{u_i,p_j} \in \mathcal{R}_{a_i}$ is a prediction of how likely Alice would be willing to grant p_j to a_i. Equation 3 is how we calculate r_{u_i,p_j} for application a_i. Note that we calculate r_{u_i,p_j} for each p_j requested by an application a_i.

$$r_{u_i,p_j} = \overline{Pr}_{p_j} + \frac{\sum_{a \in \mathcal{N}} sim(a_i, a) * d_{u_i,a}}{\sum_{a \in \mathcal{N}} |sim(a_i, a)|} \quad (3)$$

r_{u_i,p_j} is based on multiple factors: First, $\overline{Pr}_{p_j}$, which is the average probability permission p_j is granted over all applications in $\mathcal{A}$. $\overline{Pr}_{p_j}$ can easily be calculated via it's corresponding row in the $Permissions \times Applications$ matrix (see Figure 5). Note that $\overline{Pr}_{p_j}$ is driven by all users in our system. Second, $sim(a_i, a)$, which is the application-similarity value between a_i and $a \in \mathcal{N}$, where $\mathcal{N}$represents a_i's "Nearest Neighbors". The number of applications in $\mathcal{N}$ depends on the application-similarity values calculated by $sim(a_i, a_j)$ earlier. Third and finally, $d_{u_i,a}$ which is u_i's previous decisions on the applications within $\mathcal{N}$ in regards to permission p_j. $d_{u_i,a}$ values are captured via the $\mathcal{C}_{p_j}$ matrix explained earlier (see Figure 4).

Notice that the prediction values calculated are based on both a user's previous decisions and on the decisions of other users, hence capturing the essence of Collaborative Application Filtering.

5. BENEFITS AND ATTACKS

In presenting our browser extension, we must also acknowledge the various attacks against which it might be purported. We believe the most likely attack scenarios all center around abusing the extension to manipulate the recommender model into recommending decisions that most favor the attacker. These attacks might take the form of:

a) Frequent use of the extension for the same application. Attackers may use the extension to load the same application repeatedly in the hopes to increase the frequency of their own decision in the underlying database. For example, an attacker may want to direct users to reveal their email address and so may attempt to load the OAuth request repeatedly, choosing that option.

We can mitigate this risk considerably by linking the user's request to their Facebook user id (available to the plug-in through the DOM model of the browser page) and ensuring that only the most recent permission decisions are stored in the database.

b) Decoding and manipulation of the API. Attackers may decode the extension and exploit its internal programming interface to make direct calls to the underlying web service in an attempt to inflate user statistics with an eye to influencing decisions. Our previous mitigation method will be useless here since any information sent to the web-service may be captured and replayed, perhaps varying slightly the unique user information in an appearance of legitimacy. Further complicating any countermeasures is the fact that the entire code resides on the client computer and can reasonably be suspect to successful reverse engineering.

In this case, we propose to use out-of-band methods to attempt to deter and detect fraud. We can use the source IP address as one method to further link an individual web service call to a unique user. This has an admittedly loss of fidelity when user requests are routed through proxies (such as those found in institutional use, or at very large internet service providers). Combining an IP address with a minimum time interval between requests may provide us

adequate protection in this regard. Limiting unique web-service calls to calls within a 15 minute window from the same source IP address may serve as a sufficient preventative control; post-request log file analysis may further provide some detective measures towards preventing fraud.

Our web-service must also be resilient against known web attacks and our servers undergo web application and server scanning to prevent the most common and known attack vectors such as buffer overflow, SQL injection, and cross-site scripting attacks.

6. EXPERIMENTS

We evaluate our proposed OAuth 2.0 extension using Facebook as our target platform, keeping in mind that our extension is also applicable to other OAuth 2.0 platforms. Facebook is an ideal target given its large user base of over 500 million active users, and its extensive directory of third party applications (over 550,000 active applications) [9]. Facebook is also one of the major platforms to adopt the OAuth 2.0 protocol, which makes it a good fit for our evaluation process.

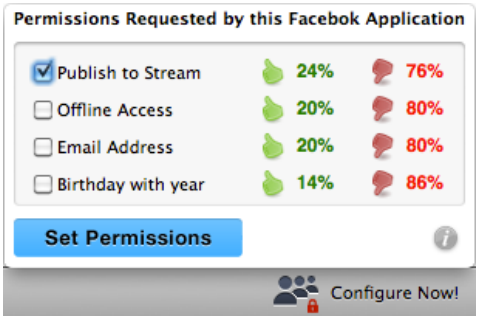


Figure 7: Permission Guide Plugin UI

To evaluate our proposed OAuth extension, we implemented two main components: a permission guide, and a recommendation service.

Permission Guide.

Our proposed Permission Guide in section 4.1 was implemented as a Firefox plugin using a combination of Mozilla's XML User Interface Language (XUL) [29] and Javascript. XUL was used to build the user interface of the plugin as seen in figure 7 whereas Javascript was used to interact with our back-end recommendation service API. The plugin was tested on Firefox versions 3.6 and 3.5 running on Mac OS X 10.6, Linux CentOS and Windows (XP, Vista) machines.

Once installed, the extension resides within the browser's bottom status bar and begins monitoring, waiting for a Facebook application installation process to commence. The extension does not otherwise interfere with a user's browsing experience. Once a Facebook application installation process is detected, the extension is activated and instantly appears in the bottom right corner of the browser's window. An installation process is detected by parsing the URLs a user visits and searching for the following:

1. **Permission Request:** The URL for permission requests by Facebook applications can be identified by locating the substrings *permissions.request* and *facebook.com/connect/uiserver*, denoting a Facebook application permission request.
2. **Request Type:** If a permission request is detected, the plugin looks for the type of request issued, i.e. Basic permission access, or Extended permission access. A basic permission access request is identified by a missing or empty *scope* attribute within the permission request URL. Otherwise, if the *scope* attribute is located, the extension recognizes that an extended permission access request in progress.

Recommendation Service.

The service is a PHP based solution running on Apache 2.2.14 with MySQL 5.1.5 as the data store solution. We run the service on a desktop machine running Linux CentOS, with 2GB RAM and a 2.0 GHz Intel Xeon CPU. The recommendation service applies the recommendation based schema explained in section 4.2 by providing two private API methods which are used by our Firefox plugin. The first API method is the *getRecommendations* method which accepts an *app_id* and a set of requested permissions. It then returns a set of recommendations in a JSON [23] format which maps a recommendation value to each permission. The second API method provided is the *postDecisions* method which is invoked by our plugin when a user makes her decision on the requested permissions. This API method takes an *app_id*, a set of requested permissions, a set of user decisions on these permissions, and the set of recommendation values displayed at decision time. These values are stored onto our recommendation back-end server and used later in our recommendation based schema.

For our evaluation purposes, we are primarily focused on extended permission requests because those are the permissions which are customizable by users on the targeted platform (Facebook), whereas basic permissions are mandatorily granted to each installing application. In the case of basic permission access requests, our extension notifies users that basic access is requested, and no customization is possible. In the case of extended permission requests for application a_k, the plugin performs the following :

1. Extracts the extended permissions requested by parsing the *scope* value from within the request URI. For Facebook, the *scope* value is a list of comma-delimited strings, each string representing a certain requested permission.
2. Asynchronously retrieves recommendations for the set of requested permissions by invoking our *getRecommendations* API. Once the recommendations are retrieved, the plugin user interface is updated properly.
3. Dynamically generates the user interface to be shown to the user based on the requested permissions and their respective recommendation values. Figure 7 is an example interface for

   scope=publish_stream,offline_access,user_photos,email

4. Attracts the user's attention by changing the plugin status to "Customize Now!" and revealing the user interface at the bottom right corner of Firefox.

Figure 8 illustrates the overall process of handling and detecting a Facebook application permission request.

Once the user makes a decision on the permissions she would like to grant or deny by clicking the "Set Permissions" button, the plugin will perform two actions: 1)Invoke

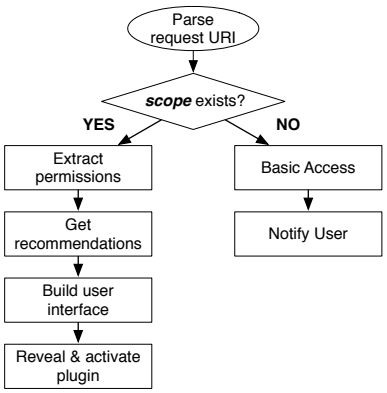


Figure 8: Handling Permission Requests

our *postDecisions* API method passing along the user decisions. 2)Generate a new *scope* value using the permissions granted by the user. Using this new scope $scope'$ the user is then redirected to a customized application request URI, resulting in a new Facebook application permission request page. At this point the user has defended herself against unnecessary accesses by applications. It is important to note that our approach does not allow applications to get access to private data (e.g. email) at all, because we prevent this before the application is installed. The default approach by Facebook is to remove access to private data after the application is installed, which is realistically not sufficient because applications have already gotten access to the data they need.

6.1 Results

Our proposed browser extension is hosted on the Mozilla Add-Ons website under the add-on name (FBSecure)[2], in addition to our own website[3]. These two websites, in addition to using Twitter [39], were used as means of recruiting participants for this study which is approved by our university's IRB. The extension was installed by 1286 Facebook and Firefox users who installed 1561 unique Facebook applications. The results summarized in this section are based on the population of users who installed our Firefox extension, use Facebook, and sought out privacy extensions. This user sample is mainly biased towards privacy aware users, but also includes regular users recruited via Twitter, whom did not specifically seek out privacy extensions.

We gathered over 4700 user decisions on 56 different Facebook extended permissions. In this section we present the evaluation of our recommendation model based on the user decisions collected during the usage of the plugin. For every application permission request, our extension enabled the collection of the details of the requested permission, the generated recommendation, and the user selected permission settings. Figure 9(a), shows the probability of applications requesting different permissions, for example we found that the most popular requested permission is the "publish_stream" permission, which enables apps to post messages on a user's wall, and is requested by 45% of the Facebook apps. Other popular permissions include email, offline_access and user_birthday.

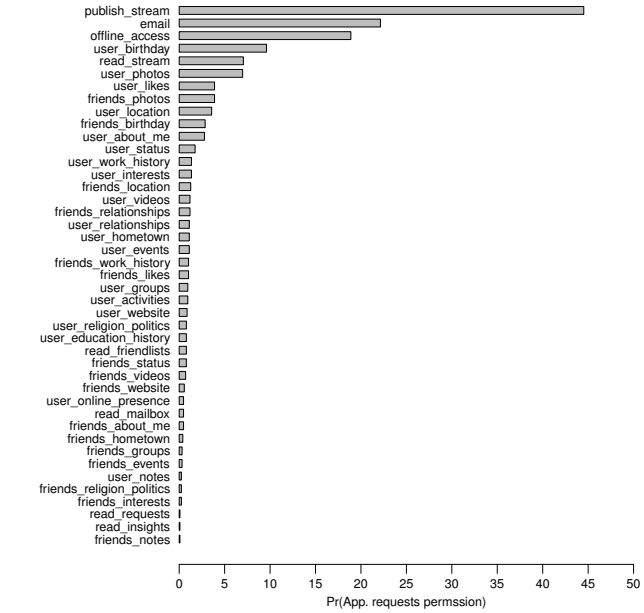
Over all our user population, Figure 9(b) shows how likely users were willing to grant different permissions. Our results show that users have varying willingness towards different permissions, for example only 31% of the users shared their email will apps, while 58% shared their online status. Note that some of the permissions requested from a user to give application access to his friends' information, for example "friends_location". Users were on average 12.3% more willing to share their friends' data compared to their own data. Figure 9(c), summarizes distribution of the number of permissions requested by applications, with an average of 3.1 permissions requested per application. Figure 9(d), shows the average number of granted permissions for apps requesting permissions, it can be noted that on average applications are granted around 44.7% of the permissions that are requested. Figure 9(e), shows the distribution of number of applications by users who installed the plugin, on average the plugin was used to install 5.2 applications. The extension provides users with recommendations for each of the application requested permissions. The recommendation is presented to the user as thumbs up and thumbs down with their associated recommendation values based on the the recommender model presented in previous sections. We are interested in evaluating whether the recommender system properly predicts the user's decision. Also we are interested in evaluating what is the lowest (highest) recommendation value that will influence users into granting (denying) a requested permission, we refer to this value as the threshold T. Where users said to be encouraged to grant the permission if the recommendation is higher than T and to deny otherwise. In this case we have four possible outcomes for the recommended and decided value, as shown in Table 6.1. In literature there are several proposed metrics for evaluating recommender system performance, we focus on the most adopted metrics in literature which are based on three measures namely accuracy, precision and recall [19]. Accuracy of the recommender system is the degree of closeness of the recommender system to the actual decision taken by the user, which is calculated as $\frac{TP+TN}{TP+TN+FP+FN}$. The precision or the repeatability of the recommender system, is a measure of the degree to which repeated recommendations under the similar conditions generate the same results, which is computed as $\frac{TP}{TP+FP}$. The recall or sensitivity is a measure of the ability of the recommender system to select instances of either to recommend or not, which is computed as, $\frac{TP}{TP+FN}$. Figure 10, shows the accuracy, precision and recall calculated for different threshold values. Note that the system maintained an accuracy about 90% over all threshold values. The precision and recall are inversely proportional with a break even region around the threshold value of 45%, which could explain that the recommendation value of 45% or higher is an indication that the system is recommending to grant the requested permission, and lower than 45% is recommending to deny the permission. Also note that the system achieves a precision and recall values of 92% and 85% around this threshold.

	Recommended	**Not Recommended**
Used	True Positive (TP)	False Negative (FN)
Not Used	False Positive (FP)	True Negative (TN)

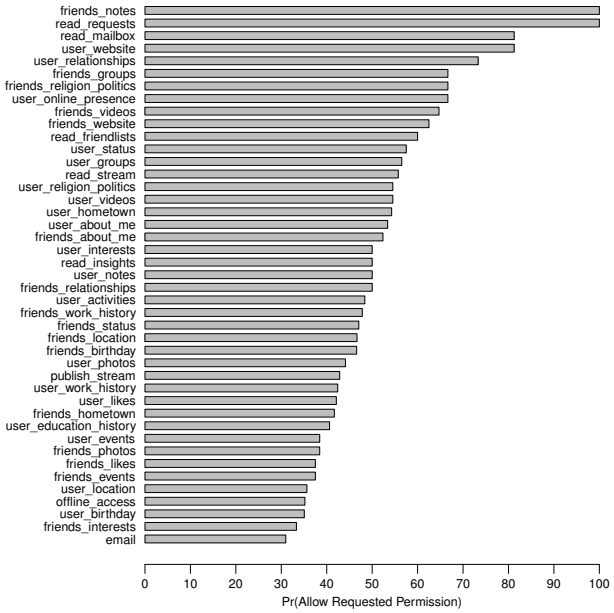
Table 1: Classification based on user decisions

In addition to investigating the accuracy, precision and recall measures we further investigated the causality of our

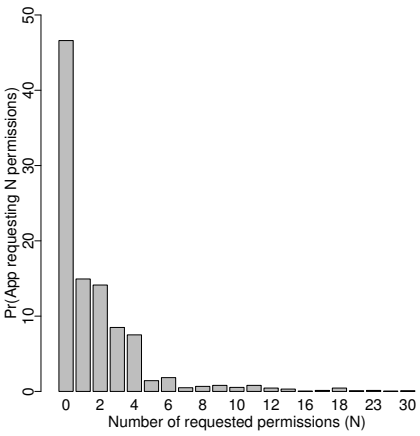
[2]`https://addons.mozilla.org/firefox/addon/fbsecure`
[3]`http://liisp.uncc.edu/fbs`



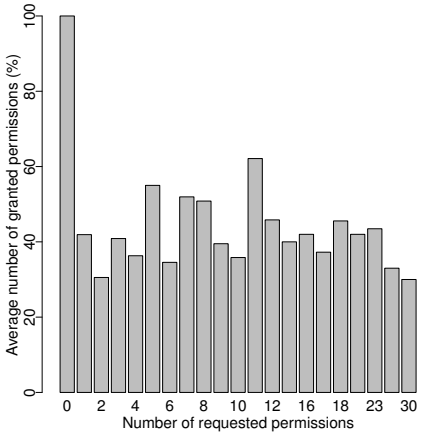
(a) Probability of Requesting a Permission



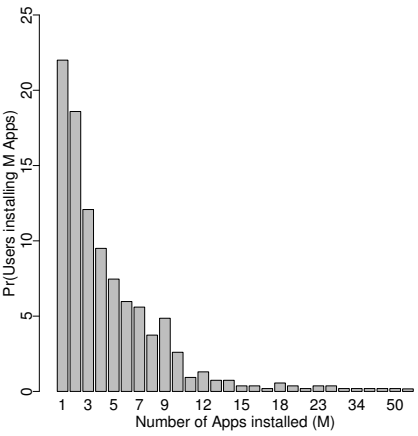
(b) Probability of Allowing a Requested Permission



(c) App. Requested Permissions



(d) App. Granted Permissions



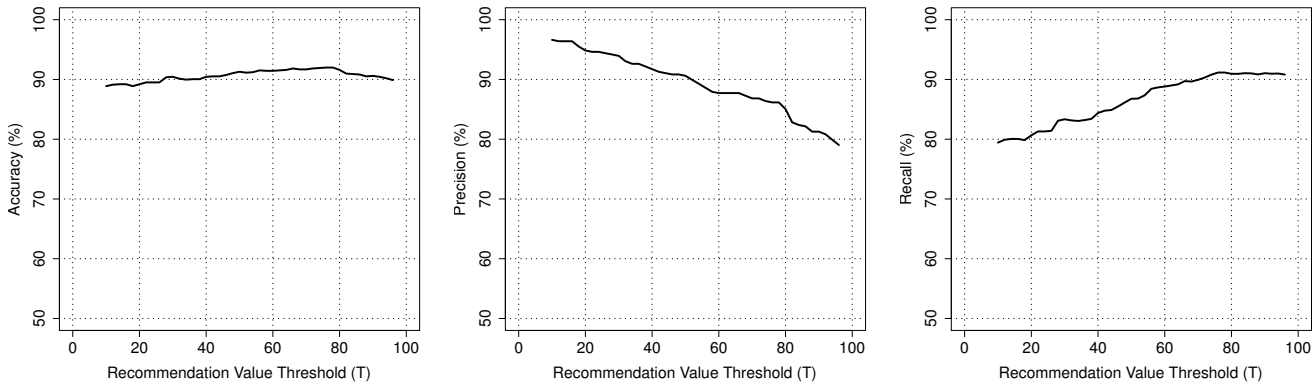
(e) User Plugin Usage Stats

Figure 9: Application and User Permission Probability Profiles

recommendation scheme. That is, do users who use the recommendation based scheme less likely to grant permissions to applications. To investigate, our browser extension was designed to accommodate two groups of users. The first group (G1), are users who were not shown the recommendation values (see Figure 11(a)). The second group (G2), are users who were shown the recommendation values generated by the recommendation system (see Figure 11(b)). The plugin randomly selected users who belonged in each of the groups. For each group we recorded the users' *openness*, which is the percentage of granted permissions for each application installed. The average user openness of G1 and G2 were 57.4% and 33.2% respectively, which indicates that users who were not presented with the recommendation were more likely to grant permissions to applications. To compare the two groups we performed a T-test of the hypothesis to investigate the following question, "on average, are users in G2 less open than users in G1?". Using the collected data, with a significance level of 5% this hypothesis was accepted (P-Value of 0.0001). These results show that the users who were presented with the recommendation values were less open to granting permissions to applications. Note that, in this experiment there was no control over specific apps within each group. The results presented in this experiment are based on the average openness values calculated over all installed apps in both groups. This experiment can be further enhanced by controlling specific apps across both groups, which will be addressed in future experiments.

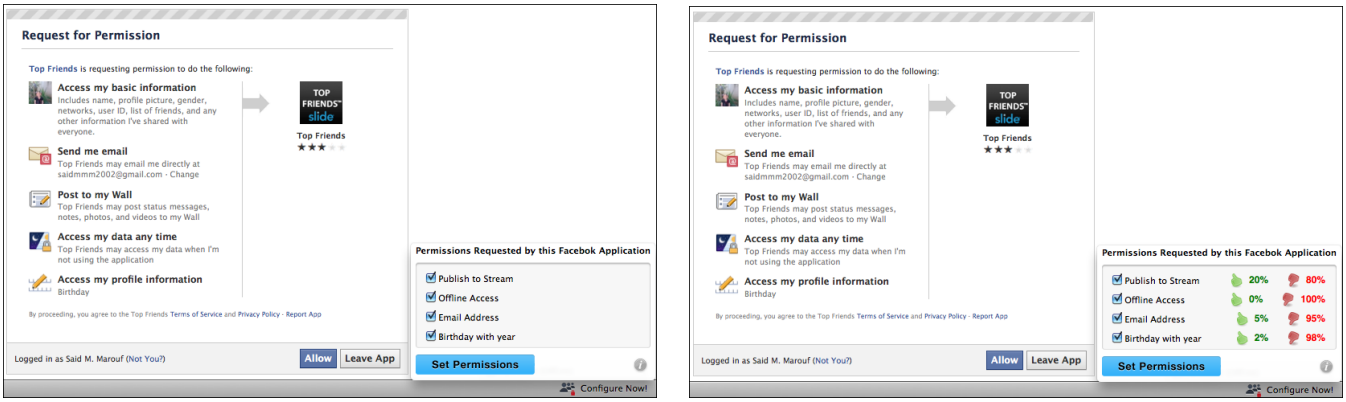
7. RELATED WORK

Developing usable tools that provide fine-grained control over user private data is a highly emerging problem in online platforms especially within social networks (e.g. Facebook, Twitter, MySpace) [15, 1, 6, 17]. Studies such as the one by Acquisti and Gross [16, 1] indicate user concern over



(a) Accuracy (b) Precision (c) Recall

Figure 10: Recommendation system accuracy, precision and recall evaluation



(a) Group (G1) no recommendation shown (b) Group (G2) recommendation shown

Figure 11: Recommendation experimental setup

their privacy on social networks while most users did not apply strict privacy settings on their online social profiles. This was mostly due to the lack or poor understanding of what privacy controls are available to them. Our work in this paper focuses on providing a usable tool via a browser extension which allows for users to easily understand and customize their privacy settings at application installation time. We also increase user awareness by providing them a set of recommendations on the requested privacy attributes.

Using browser-based plugins and extensions is another widely used approach in aiding information privacy. Jenkin et al. [21] originally identified this approach to address the problem of "an *information provider* wanting to serve *secrets* embedded within regular web pages to *authorized users*". The authors' original italicized elements hold true for our model, substituting *users* for information providers, *identity attributes* for secrets, and *authorized third party applications* for authorized users. Today the "open source" model is routinely used to enable community-contributed plug-ins for web browsers that aid in some aspect of security; as one example, the Mozilla Firefox browser boasts over 500 security and privacy plug-ins available at their add-ons website [30]. As part of this community we made our browser extension available to the public via Mozilla's website.

Felt et al. [11] detail a novel solution for protecting privacy within social networking platforms through the use of an application programming interface to which independent application owners would agree to adhere to. Our approach requires no such agreement and, barring a wholesale adoption of a privacy proxy such as the one which Felt proposes, still enables the user to protect their information attributes. We achieve this by utilizing the already popular OAuth 2.0 authorization flow and providing a seamless experience to users for customizing and protecting their private information attributes.

Fang and LeFevre's work asserts the value in providing highly accurate privacy settings with reduced user input [10]. Using real user input, they infer a set of privacy-preferences using a machine learning approach. While the authors' study is based on real users, i.e collecting data from real users, they do not provided a technique that applies the inferred privacy settings onto a user's real online profile. Our implemented browser extension is not only based on real user data, but also capable of applying a user's desired privacy decisions to their real online profile, in this case their Facebook profile. They also focus on privacy settings related to a user's friends network, where our work focuses on third party applications which we believe represent a big-

ger threat to user privacy, especially given the huge catalog of third party online social applications which request access to a user's private data and are also able to execute actions on behalf of users (e.g. posting images to a user's profile, or sending an SMS message). We also note that using a machine learning technique is not ideal in situations where instant privacy suggestions are required, which is the case when installing third party applications within social networks such as Facebook.

Besmer et al. [4] demonstrated in their research the value of social navigation cues in prompting users to make informed privacy decisions; where that research was not concerned with the type of data and arbitrarily assigned a recommended positive or negative cue for each item, our research is very specifically tied to data types and our recommender model provides cues that are based on real user privacy decisions. Our work is also significantly different in that we provide a real life user study through a real world implementation of a browser extension that integrates with a user's actual Facebook profile, where the research by Besmer et al. was experimental. We also note that our browser extension applies a user's privacy decisions onto their real privacy settings within Facebook.

While much has been researched about the privacy impacts of recommender systems themselves [36, 34, 7], little research appears to be available for the use of recommender systems in aiding privacy and security systems. One notable exception is in the research of Kelly et al. [24] where the authors demonstrated the benefit of combining collaboration among a user population in the suggestion of an individual user's privacy policy. They also propose an incremental model for optimizing a user's policy over time. We find this approach not optimal when dealing with third party applications, that once installed, can harvest a user's private social network data. Optimal and instant privacy protection should be provided to users at installation time, which we achieve through our browser extension.

Liu and Terzi offer a framework for deriving a "privacy score" to inform the user of the potential risk to their privacy created by their activities within the social network [27]. However, such research does not account for the discrete control over attributes which our research enables.

Goecks et al. [13] explore the effects of community data in the domains of firewall policy configuration and web browser cookie management. Their research indicates that users utilized community data in making their privacy decisions. They also investigate the effects of *informational cascades* and the possible misuse of community data within social navigation systems. They present two approaches for mitigating the effects, and we believe these can complement our work.

8. CONCLUSION AND FUTURE WORK

Usable privacy configuration tools are essential in providing user privacy and protecting their data from third party applications in social networks. We proposed an extension to the web server authorization flow of OAuth, and implemented a browser-based extension that integrates into the existing authorization flow, and provides users the ability to easily configure their privacy settings for applications at installation time. We also proposed a multi-criteria recommendation model based on collaborative filtering which incorporates the decisions of the community and previous decisions made by an individual user. Based on this model our browser extension provides users with recommendations on permissions requested by applications. We successfully demonstrate that our extension, combined with our multi-criteria recommendation model leads to the preservation of irrevocable, immutable private identity attributes and the preventing of their uninformed disclosure during application installation. Among popularly requested extended permissions, individuals - when given the choice - will, in the majority of cases, deny the request. We demonstrate the effectiveness of the recommendations through a causal group of users who were not shown any recommendations, and we found them to be more willing to grant permissions to third party applications than those who were provided with recommendations.

In the future, we will investigate cases where over time, users might setup multiple privacy configurations for the same application. We will investigate possible reasons for this, such as: a user's increased privacy awareness, or misconfigurations that hinder an application's ability to function properly. In the later case, we can build a feedback mechanism that handles such misconfigurations. We also plan on investigating the effect of using different visual styles of showing recommendations. Instead of thumbs up and down, we could explore using a horizontal bar indicating the recommendation value and adapting its color to the value itself. We will also investigate clustering applications based on their functionality before composing the recommendations and incorporate this the similarity measure. Finally, we want to test our proposed framework in OAuth 2.0 environments other than Facebook, plus provide various versions of our browser extension that are compatible with other browsers such as Google Chrome, and Safari.

9. ACKNOWLEDGEMENTS

This work was funded by the National Science Foundation (NSF-CNS-0831360).

10. REFERENCES

[1] A. Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Privacy Enhancing Technologies*, pages 36–58, 2006.

[2] G. Adomavicius and Y. Kwon. *In Recommender Systems Handbook: A Complete Guide for Research Scientists and Practitioners*, chapter Multi-Criteria Recommender Systems - Forthcoming. Springer, 2010.

[3] G.-J. Ahn, M. Ko, and M. Shehab. Privacy-enhanced user-centric identity management. In *ICC*, pages 1–5, 2009.

[4] A. Besmer, J. Watson, and H. R. Lipford. The impact of social navigation on privacy policy configuration. In *Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS 2010, Redmond, Washington, USA, July 14-16, 2010*.

[5] W. Bin, H. H. Yuan, L. X. Xi, and X. J. Min. Open identity management framework for saas ecosystem. In *e-Business Engineering, 2009. ICEBE '09. IEEE International Conference on*, pages 512 –517, 2009.

[6] D. Carrie and E. Gates. Access control requirements for web 2.0 security and privacy. In *Proc. of Workshop on Web 2.0 Security & Privacy (W2SP 2007)*, 2007.

[7] S. Chen and M.-A. Williams. Towards a comprehensive requirements architecture for privacy-aware social recommender systems. In *APCCM '10: Proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling*, pages 33–42, 2010.

[8] L. F. Cranor. P3p: Making privacy policies more useful. *IEEE Security and Privacy*, 1:50–55, 2003.

[9] Facebook. Facebook Press Room. *http://www.facebook.com/press/info.php?statistics*, 2010.

[10] L. Fang and K. LeFevre. Privacy wizards for social networking sites. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *WWW*, pages 351–360. ACM, 2010.

[11] A. Felt and D. Evans. Workshop on web 2.0 security and privacy. oakland, ca. 22 may 2008. privacy protection for social networking platforms, 2008.

[12] FriendCameo, Inc. FriendCameo. *http://friendcameo.com*, 2010.

[13] J. Goecks, W. K. Edwards, and E. D. Mynatt. Challenges in supporting end-user privacy and security management with social navigation. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, pages 5:1–5:12, New York, NY, USA, 2009. ACM.

[14] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[15] K. K. Gollu, S. Saroiu, and A. Wolman. A social networking-based access control scheme for personal content. *Proc. 21st ACM Symposium on Operating Systems Principles (SOSP '07). Work in progress.*

[16] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, WPES '05, pages 71–80, New York, NY, USA, 2005. ACM.

[17] M. Hart, R. Johnson, and A. Stent. More content - less control: Access control in the Web 2.0. *Web 2.0 Security & Privacy*, 2003.

[18] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the international ACM SIGIR conference*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[19] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53, January 2004.

[20] A. Herzog and N. Shahmehri. User help techniques for usable security. In *Proceedings of the 2007 symposium on Computer human interaction for the management of information technology*, CHIMIT '07, New York, NY, USA, 2007. ACM.

[21] M. Jenkin and P. Dymond. A plugin-based privacy scheme for world-wide web file distribution. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 7, pages 621 –627 vol.7, Jan. 1998.

[22] Y. Joshi, D. Das, and S. Saha. Mitigating man in the middle attack over secure sockets layer. In *2009 IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA)*, pages 1–5. IEEE, December 2009.

[23] JSON. JSON. *http://tools.ietf.org/html/rfc4627*, 2006.

[24] P. G. Kelley, P. Hankes Drielsma, N. Sadeh, and L. F. Cranor. User-controllable learning of security and privacy policies. In *AISec '08: Proceedings of the 1st ACM workshop on Workshop on AISec*, pages 11–18, New York, NY, USA, 2008. ACM.

[25] H.-H. Lee and W.-G. Teng. Incorporating multi-criteria ratings in recommendation systems. In *IRI'07*, pages 273–278, 2007.

[26] M. Li, B. Dias, W. El-Deredy, and P. J. G. Lisboa. A probabilistic model for item-based recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 129–132, New York, NY, USA, 2007. ACM.

[27] K. Liu and E. Terzi. A framework for computing the privacy scores of users in online social networks. In W. Wang, H. Kargupta, S. Ranka, P. S. Yu, and X. Wu, editors, *ICDM*, pages 288–297. IEEE Computer Society, 2009.

[28] M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 329–336, New York, NY, USA, 2004. ACM.

[29] Mozilla. Mozilla XML User Interface Language (XUL). *https://developer.mozilla.org/En/XUL*.

[30] Mozilla Firefox. Mozilla Firefox Privacy and Security Extensions. *https://addons.mozilla.org/en-US/firefox/extensions/privacy-security*.

[31] OAuth 2.0. Access Token, OAuth 2.0. `http://tools.ietf.org/html/draft-ietf-oauth-v2-10\#page-5`.

[32] OAuth 2.0. The OAuth 2.0 Protocol. *http://tools.ietf.org/html/draft-ietf-oauth-v2-10*, 2010.

[33] OAuth 2.0. Web server client profile, oauth 2.0. `http://tools.ietf.org/html/draft-ietf-oauth-v2-10\#section-1.4.1`, 2010.

[34] N. Ramakrishnan, B. Keller, B. Mirza, A. Grama, and G. Karypis. Privacy risks in recommender systems. *Internet Computing, IEEE*, 5(6):54 –63, 2001.

[35] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.

[36] J. Riedl. Personalization and privacy. *Internet Computing, IEEE*, 5(6):29 –31, 2001.

[37] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[38] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.

[39] Twitter, Inc. Twitter. *http://twitter.com*.