USER CENTRIC POLICY MANAGEMENT

by

Gorrell P. Cheek

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2013

Approved by:

Dr. Mohamed Shehab

Dr. Bill Chu

Dr. Cem Saydam

Dr. Weichao Wang

Dr. Craig Depken

ABSTRACT

GORRELL P. CHEEK. User centric policy management. (Under the direction of DR. MOHAMED SHEHAB)

Internet use, in general, and online social networking sites, in particular, are experiencing tremendous growth with hundreds of millions of active users. As a result, there is a tremendous amount of privacy information and content online. Protecting this information is a challenge. Access control policy composition is complex, laborious and tedious for the average user. Usable access control frameworks have lagged. Acceptance / use of available frameworks is low. As a result, policies are only partially configured and maintained. Or, they may be all together ignored. This leads to privacy information and content not being properly protected and potentially unknowingly made available to unintended recipients.

We overcome these limitations by introducing User Centric Policy Management – a new paradigm of semi-automated tools that aid users in building, recommending and maintaining their online access control policies. We introduce six user centric policy management assistance tools: *Policy Manager* is a supervised learning based mechanism that leverages user provided example policy settings to build classifiers that are the basis for auto-generated policies. *Assisted Friend Grouping* leverages proven clustering techniques to assist users in grouping their friends for policy management purposes. *Same-As Subject Management* leverages a user's memory and opinion of their friends to set policies for other similar friends. *Example Friend Selection* provides different techniques for aiding users in selecting friends used in the development

of access control policies. *Same-As Object Management* leverages a user's memory and perception of their objects for setting policies for other similar objects. *iLayer* is a least privilege based access control model for web and social networking sites that builds, recommends and enforces access control policies for third party developed applications.

To demonstrate the effectiveness of these policy management assistance tools, we implemented a suite of prototype applications, conducted numerous experiments and completed a number of extensive user studies. The results show that User Centric Policy Management is a more usable access control framework that is effective, efficient and satisfying to the user, which ultimately improves online security and privacy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Online social networking sites are experiencing tremendous adoption and growth. The internet and online social networks, in particular, are a part of most people's lives in the US. Although, growth is expected to slow; eMarketer reports that over 150 million US internet users will interface with at least one social networking site per month [16]. That is approximately 64% of all internet users – see Figure 1.

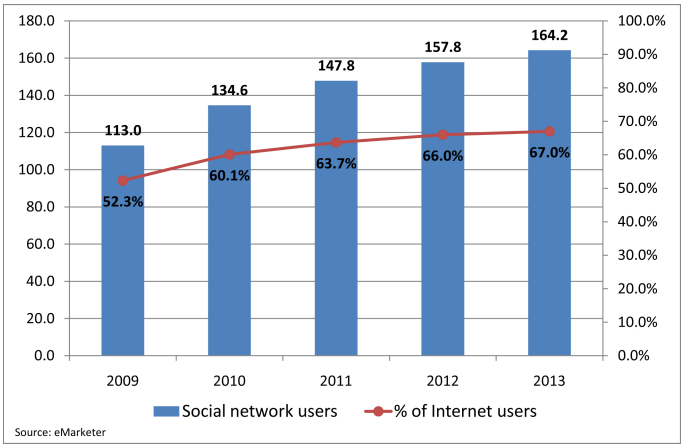


Figure 1: US social network users and penetration

eMarketer also reports that approximately 90% of internet users ages 18-24 and 82% of internet users ages 25-34 will interact with at least one social networking site per month. This trend is increasing for all age groups – see Figure 2. As the young population ages, they will continue to leverage social media in their daily lives. In addition, new generations will come to adopt the internet and online social networks. These technologies have become and will continue to be a vital component of our

social fabric which we depend on to communicate, interact and socialize.

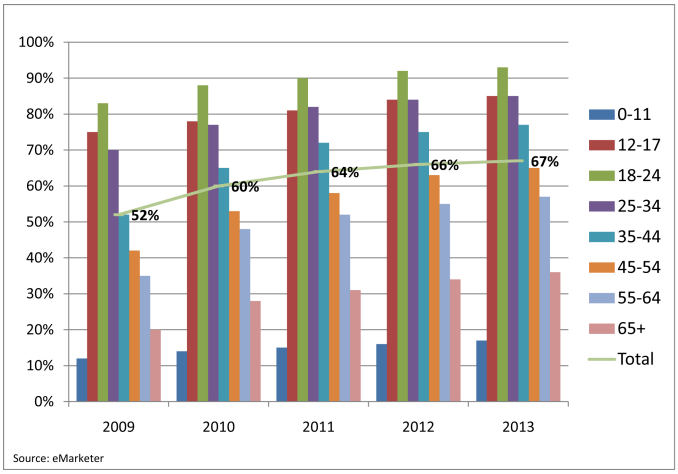


Figure 2: US social network user penetration, by age

Not only are there a tremendous amount of users online, there is also a tremendous amount of user profile data (e.g., name, birthday, education, work history, etc.) and content (e.g., web links, notes, photos, etc.) online. For example, Facebook touts that they have over one billion active monthly user [17]. Each of these users has an average of 130 friends and creates over 90 pieces of new content each month. Over 30 billion pieces of content are shared each month between users. This large amount of content coupled with the significant number of users online makes maintaining appropriate levels of privacy very challenging.

There have been numerous studies concerning privacy in the online world [9, 33, 39]. A number of conclusions can be drawn from these studies. First, there are varying levels of privacy controls, depending on the online site. For example, some sites make available user profile data to the internet with no ability to restrict access. While other sites limit user profile viewing to just trusted friends. Other studies introduce the notion of the privacy paradox, the relationship between individual privacy intentions

to disclose their personal information and their actual behavior [50]. Individuals voice concerns over the lack of adequate controls around their privacy information while freely providing their personal data. Other research concludes that individuals lack appropriate information to make informed privacy decisions [6]. More over, when there is adequate information, short-term benefits are often opted over long-term privacy. However, contrary to common belief, people are concerned about privacy [5, 14]. But, managing ones privacy can be challenging. This can be attributed to many things, e.g., the lack of privacy controls available to the user, the complexity of using the controls and the burden associated with managing these controls for large sets of users.

1.1 Viewer Policy Management

As stated previously, online social networks have grown tremendously in recent years. Managing access to one's privacy information and content for all these viewers / friends (consumers of user information / content) is quite challenging. Online social networks provide access controls frameworks. However, these frameworks vary in capability from very basic coarse-grained sets of access controls with limited flexibility to more fine-grained sets of access controls that are overly cumbersome and difficult to use. Research has found that managing access to online information (both privacy and content) is traditionally manual, complex, difficult and time consuming [15, 33, 39]. As a result, access control frameworks are not widely adopted and, therefore, users' information is potentially exposed in unwanted ways [34, 62, 9]. For example, unintended sharing of content occurs over a third of the time as a result of relying on default privacy settings [41]. In addition, almost two thirds of the time,

users' expectations don't align with their privacy settings because of the difficulty in managing them – potentially resulting in unintended information leakage [42].

Additional research points to the long sought after goal and importance of usable security [8, 36, 66]. But, *usability* and *security* often have competing objectives. According to ISO 9241-11 (1998), *usability* is the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." ISO 17799 (2005) states that "*security* is achieved by implementing a suitable set of controls... to ensure that the specific security and business objectives... are met." How do we build suitable access control frameworks that are effective, efficient and satisfying to the end user? These frameworks must adhere to all three criteria. If they are suitable to the end user but don't provide effective controls, then security is not achieved. Conversely, access control frameworks not only have to be effective and efficient, they must also be satisfying. If they are not, they are unlikely to be used and therefore become ineffective. Saltzer and Schroeder [56] emphasize the importance of *psychological acceptability* as a key protection mechanism design principle. "It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly. Also, to the extent that the user's mental image of his protection goals matches the mechanisms he must use, mistakes will be minimized."

Access control frameworks must be easy to use. Users must be able to manage access to their online information in a simple and intuitive way that aligns with their intentions. In addition, these frameworks must be designed such that they adhere to a user's mental image or model for managing and controlling access to their

online information. Jones et al. [24] describe mental models as "personal, internal representations of external reality that people use to interact with the world around them; [they] are used to reason and make decisions and can be the basis of individual behaviors." Specific to access control frameworks, a user's mental model is their understanding of how access to their online information is managed and controlled. It is not necessarily derived from formal instruction or training of the framework. Even so, the framework's capabilities should align, as much as possible, to the user's mental model. For example, a user's mental model of an access control policy should align with how the online social network evaluates that policy. If the user's intent is to limit access to a set of sensitive pictures to just family members, the online social network should evaluate and enforce that policy accordingly. The more alignment of an access control framework with a user's mental model, the less likely for policy errors and unintended information leakage. We believe that *User Centric Viewer Policy Management Assistance Tools* need to be placed in the hands of the average user to assist them in effectively managing access to their online information.

1.2 Application Policy Management

Unfettered and unchecked access of third party applications is another security vulnerability that puts social networks and their online information at risk. For example, a hypothetical social networking web site called *Social123.com* is powered by a content management system. The administrative user has installed several third party applications to customize and enhance the social networking site. One installed application, *BdayCalApp*, has full access to the database, yet only requires a subset

of access, e.g., users' birthdays to be displayed on a calendar. An attacker can target *Social123.com* via *BdayCalApp* to gain access to the login credentials of all of the users on the social networking site. These credentials are stored in the *users* table within the database. A poorly written third party application may find itself vulnerable to a SQL injection attack which could expose all the tables, including the *users* table. Even though *BdayCalApp* only requires access to the table containing users' birthdays, all tables are available to the application and thus increase the risk of compromise and misuse.

On a popular content management system used to power social networking sites called Drupal [13], we conducted a study of 412 third party applications in order to analyze the database calls made by them. Through static analysis, we extracted the database table accesses made by these applications. In Drupal, database calls are made by using the *db_query()* function. For every database call, we parsed through the SQL statement to find the database tables and recorded the results. We found that applications accessed tables they created and the core tables provided by Drupal.

We also found that third party applications have significantly more access to the Drupal core tables than what is required. Only 2% of the applications required access to the *sessions* table. That leaves 98% who didn't require access and yet had access thus leaving the web site vulnerable to session hijacking – see Table 1. Only 7% of the modules required access to the *node_revisions* and *permissions* tables leaving roughly 382 applications (or modules) access to these tables and thus vulnerable to potential privilege escalation and content compromise attacks. The *users* table is a default table that holds basic user information such as user names and passwords.

Approximately 77% of the applications didn't need access to this table but did have access. These are clearly not examples of least privilege [57] and therefore the risk of compromise increases.

Table 1: Sample of Drupal core database tables

Table Name	**Table Description**	**Potential Impact**	**% of Modules That Require Access**
sessions	Contains user session information, e.g., userID, sessionID, user IP address, etc.	Session hijacking	2%
users_roles	Lists the assignments between users and roles	Privilege escalation	5%
node_revisions	Contains edits / revisions of node content	Content compromise	7%
permissions	Lists each user role's permissions	Privilege escalation	7%
users	Contains usernames, passwords, profile information, etc.	Account compromise	23%

Least privilege based access control for third party developed applications for online social networks is not available. Third party application access control is limited in capability and the capability that does exist is hard to configure for the average administrative user. Least privilege based access control frameworks for third party developed applications are needed. Applications need only be given access to the objects and resources they require. We believe that administrators need additional *User Centric Application Policy Management Assistance Tools* to protect online information from attacks via third party applications.

CHAPTER 2: RESEARCH PROBLEM AND HYPOTHESIS

With the ever increasing number of internet users and content available online, it a challenging effort in managing access to information. Maintaining an effective access control policy can be a very complex, laborious and tedious task. As a result, policies are only partially configured and maintained. Or, they may be all together ignored. This leads to user data and content not being properly protected and potentially unknowingly made available to unintended recipients, e.g., viewers or "friends", third party applications, etc. In general, acceptance and use of access control policy management mechanisms designed to protect online information is low.

Our research problem is as follows:

Problem Statement: Users, to include administrative users, are not effectively setting / managing access control policies in online systems and therefore profile data and content can potentially be exposed and compromised.

Our hypothesis is as follows:

Hypothesis Statement: Enhanced user centric policy management mechanisms are more effective, efficient and satisfying to the user, ultimately improving user security and privacy over more traditional online policy management approaches.

CHAPTER 3: SOLUTION OVERVIEW

3.1 Overview

There are two primary consumers of online information: viewers and third party applications – see Figure 3. Viewers are traditionally users of the web / social networking site, e.g., "friends". Viewers access the target user's (also called the focus user's) profile data and content. For example, you are the focus user and your friend Bob (viewer) reads (consumes) your Facebook user data profile containing your education and work history.

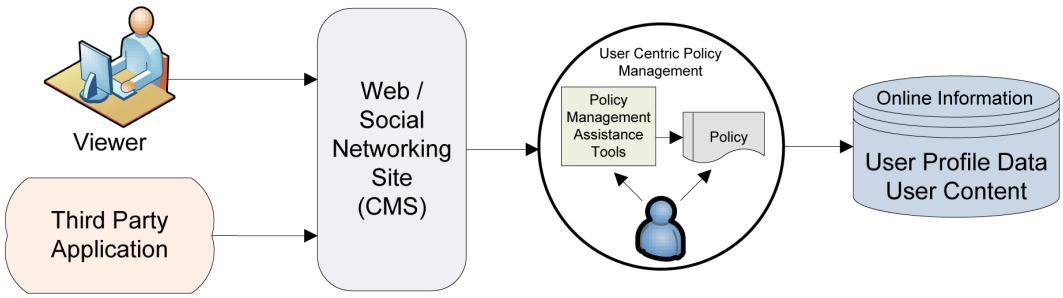


Figure 3: User centric policy management overview

Third party applications expand the capabilities and functionalities of web and social networking sites, of which content management systems (CMS) are used to build and maintain. Third party applications also consume online information. They are developed by a different entity other than the primary web or social networking site. These applications provide services to users and interact with users by combining and aggregating online information, to include user information. For example, an

application that provides birthday gift recommendations for friends needs to access the focus user's profile and friends' profiles to access such information as birth dates, interests and addresses in order to make appropriate birthday gift recommendations. Web and social networking sites provide sets of Application Programming Interfaces (API) that enable third party applications to interface and access user profile data and content. We further discuss these API's, which we call *Social Networks Connect Services*, in Chapter 5.

Access control mechanisms, including their policies, lie between the consumers of the information and the actual information, refer to Figure 3. User Centric Policy Management enhances these mechanism. It primarily consists of *Policy Management Assistance Tools* with *User Centric Characteristics* that enable the user to better manage their access control policies. We argue that these enhanced policy management mechanisms are more effective, efficient and satisfying directly resulting in improved privacy and security.

3.1.1 Policy Management Assistance Tools

We introduce Policy Management Assistance Tools which are semi-automated mechanisms controlled by the user for aiding in the management of their access control policies. These tools take input data from three primary sources: 1) user input, 2) user data, e.g., other user's profile data, social network structure data, etc. and 3) application data, e.g., historical access data, etc. Via these semi-automated mechanisms that leverage all available input data, the Policy Management Assistance Tools assist the user in building, recommending and maintaining their online policies. These

policies then govern access to their profile data (e.g., name, birthday, education, work history, etc.) and content (e.g., web links, notes, photos, etc.). – see Figure 4.

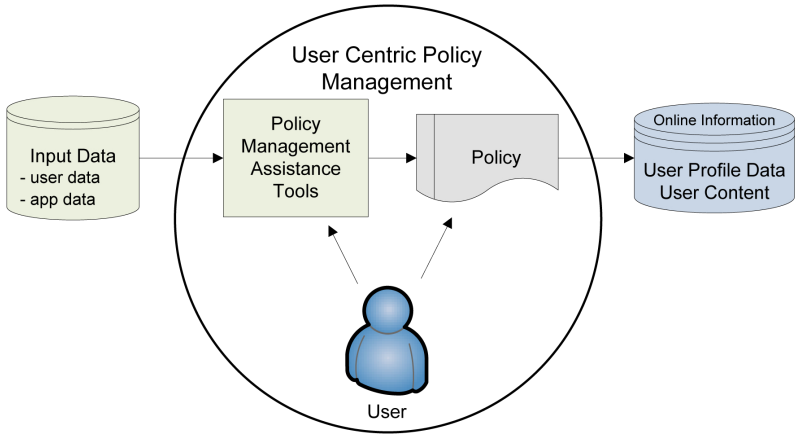


Figure 4: User centric policy management model

3.1.2 User Centric Characteristics

Our model takes a user centric view [30, 43, 29]. Whereas, the user has effective visibility and control in managing access to their online user profile data and content. User Centric Policy Management has the following characteristic: User in Control, Ease of Use, Policy Visibility and Readability, Flexibility and Accuracy – see Figure 5.

User in Control: The main premise of user centricity is that the user is in the center. In the case of User Centric Policy Management, the user is in control, or in the center, of managing access to their policies. The user is the decision maker in setting, updating and revoking access to their online information.

Ease of Use: But, putting the user in control of policy management is not enough. The user needs to be able to manage their policies in an easy, intuitive and effective way such that they have a consistent experience. Research has shown that com-

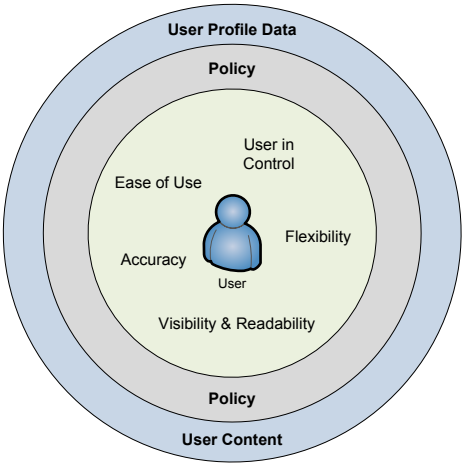


Figure 5: User centric policy management characteristics

plex and laborious policy management mechanisms can lead to ineffective policies. User Centric Policy Management is characterized by its ease of setting and managing policies that govern access to online information.

Policy Visibility and Readability: Not only does a policy management solution have to be easy to use, it must be decipherable. The core component of any access control mechanism is the policy which governs access. This policy not only must be available and visible to the user, but it also must be readable. Policies that are complex and difficult to understand are more likely to be misconfigured resulting in unintended consequences, e.g., data leakage, etc.

Flexibility: Policy management mechanisms must be flexible to accommodate the end user's needs and intentions. User Centric Policy Management creates a balance between coarse grained and fine grained access control. Traditionally, coarse grained access control provides few options to the end user. And, fine grained access control, although extremely flexible in that it provides lots of options and capabilities, is traditionally overwhelming and complex. User Centric Policy Management strikes a

balance between too little flexibility and an overly burdensome policy management mechanism.

Accuracy: Finally, policy management mechanisms that aid users in governing access to their online information must be accurate relative to the user's intentions. User Centric Policy Management is characterized by its accuracy.

3.2 Viewer Policy Management Assistance Tools

We believe more user centric online social network access control frameworks need to be placed in the hands of the user to aid them in managing and controlling viewer access to their privacy information and content. Traditionally, improvements are made in two areas: 1) changes to the underlying access control model and 2) enhancements to the user interface [55]. We introduce five Policy Management Assistance Tools that make improvements in both of these areas. See Figure 6.

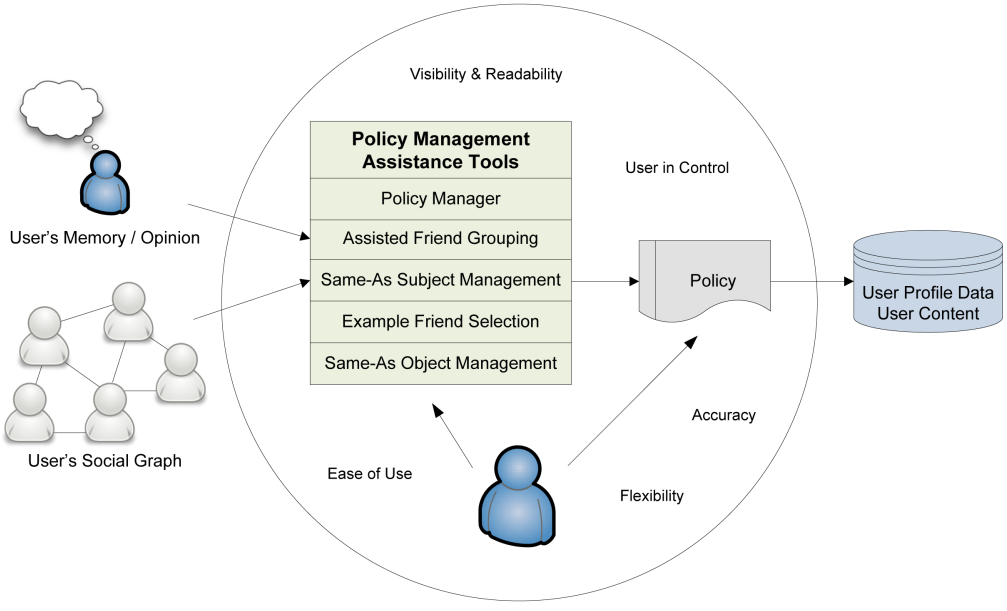


Figure 6: Viewer policy management assistance tools

Policy Manager: We propose Policy Manager (PolicyMgr) which assists users in managing access to objects posted to their profiles. Our approach leverages input from the user, metrics from their social graph, and proven supervised learning techniques to effectively identify trusted and non-trusted friends relative to a specific object, e.g., photo. The profile owner provides example policy settings as training sets to build classifiers that can precisely generate policies for other users in their friend's list – identifying them as trusted and non-trusted friends. Furthermore, we explore the fusion of policy decisions generated by their neighboring friends to enhance the accuracy of the supervised learning approach. We implemented our framework on the Last.FM social network. Policy Manager is further detailed in Section 4.1.

Assisted Friend Grouping: We introduce a user assisted friend grouping mechanism that enhances traditional group based policy management approaches. Assisted Friend Grouping leverages proven social graph clustering techniques to aid users in grouping their friends more efficiently for privacy policy management purposes. We found measurable agreement between clusters and user defined relationship groups. Our approach exhibited user centric characteristics and demonstrated promising results in assisting users in efficiently grouping and setting expressive policies for their friends. We implemented a prototype Facebook application of Assisted Friend Grouping and conducted an extensive user study. Assisted Friend Grouping is further detailed in Section 4.2.

Same-As Subject Management: We introduce a policy management approach for online social networks that leverages a user's memory and opinion of their friends to set policies for other similar friends, which we refer to as Same-As Subject Man-

agement. Using a visual policy editor that takes advantage of friend recognition and minimal task interruptions, Same-As Subject Management demonstrated improved performance and user centric characteristics over traditional group based policy management approaches. In addition, users of Same-As Subject Management authored more conservative policies. We implemented a prototype Facebook application of Same-As Subject Management and conducted an extensive user study. Same-As Subject Management is further detailed in Section 4.3.

Example Friend Selection: We introduce two techniques that leverage the user's social graph for aiding in the selection of an example friends used in developing policy templates for Same-As Subject Management. Both techniques reduced policy authoring time and were positively perceived by users. We implemented a prototype Facebook application of Example Friend Selection and conducted an extensive user study. Example Friend Selection is further detailed in Section 4.4.

Same-As Object Management: Most attention, in access control literature, has been placed on abstracting the complexity of managing large numbers of subjects. More limited research has focused on the object side of the equation, i.e., how to manage large sets of objects for the purposes of controlling access. The basic premise of access control systems is to protect and control access to sensitive resources / assets (i.e., objects). Just as subjects are dynamic, so are objects. Objects are created. Their properties change, e.g., sensitivity level, size, etc. They ultimately are retired or deleted. Managing the full life-cycle of large numbers of objects can be complex and difficult.

We introduce a policy management approach for online social networks that lever-

ages a user's memory and opinion of their objects to set policies for other similar objects, which we refer to as Same-As Object Management. Our visual policy editor is easy to use and aligns with the user's mental model for managing access to their online privacy information and content. It demonstrated improved user centric characteristics over traditional grouping based policy management interfaces. Same-As Object Management also demonstrated improved expressiveness and performance, in addition to more conservative policies, over more traditional grouping based access control models. We implemented a prototype Facebook application of Same-As Object Management and conducted an extensive user study. Same-As Object Management is further detailed in Section 4.5.

3.3 Application Policy Management Assistance Tools

We believe that web and social networking site administrators need additional tools and mechanisms to protect their information assets from attacks via third party applications. First, we analyze the general framework used by the major social networking sites for integrating third party applications, which we call Social Networks Connect Services. Next, we introduce a Policy Management Assistance Tools called iLayer which manages the accesses of third party applications. See Figure 7.

Social Networks Connect Services: Major social networking sites have rolled out new services such as Facebook Platform, Google Friend Connect, and MySpaceID. We call these new services Social Networks Connect Services. Social Networks Connect Services allow third party sites to develop social applications and extend their services into the social web without having to build their own social network. This extension

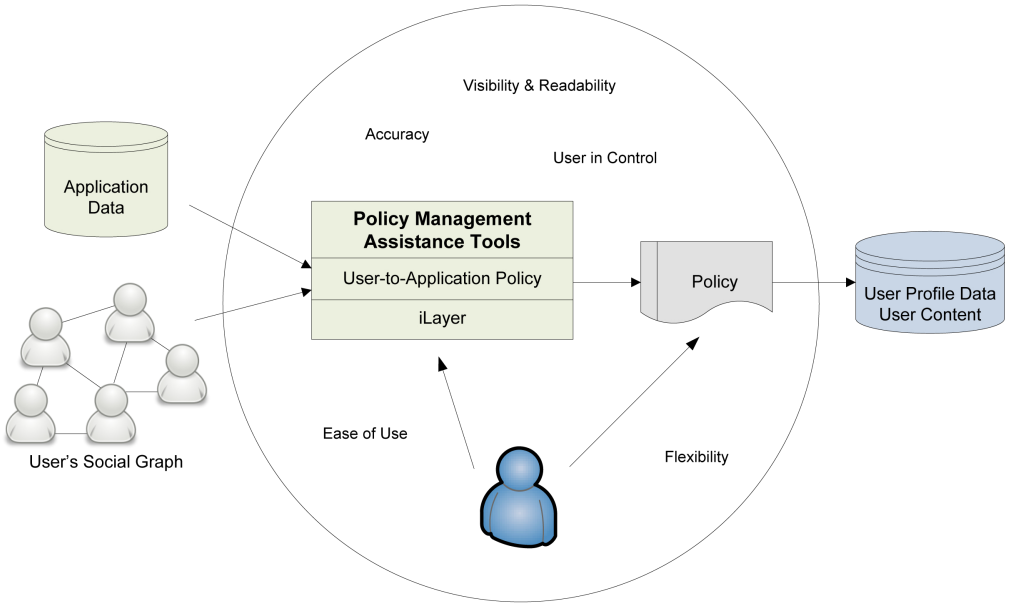


Figure 7: Application policy management assistance tools

allows third party sites to access and enrich user data in the social web. We describe the general framework of Social Networks Connect Services and compare the Facebook Platform, Google Friend Connect, and MySpaceID services relative to this general framework. We also present open research security challenges associated with the expanded use of Social Networks Connect Services. Social Networks Connect Services is further detailed in Section 5.1.

iLayer: We propose an application access control framework that is based on a least privilege security model [57]. This framework gives visibility into the accesses that third party applications request, in addition to aiding and guiding administrators in making policy decisions. The security policies are set at installation time and are enforced at run time. The framework adds another layer of protection to the web site and is an improvement of how content management systems implement access

control for third party applications. For example, some content management systems regulate access to third party applications via file permissions, which is cumbersome and difficult to administer. Administrators would need to analyze the application, know the intricacies of the CMS platform, and be familiar with the underlying operating system to effectively set the appropriate file permissions. This is a difficult and tedious undertaking. With our proposed approach (iLayer), the administrator need only set the policy at installation time and the framework enforces that policy at run time.

The iLayer Application Access Control Framework: 1) is based on a least privilege model, 2) protects web and social networking sites from third party applications and 3) provides administrative users with third party application policy setting functionality, including recommendations for policy settings. We implemented a prototype of our framework and demonstrated its feasibility. iLayer is further detailed in Section 5.2.

CHAPTER 4: VIEWER POLICY MANAGEMENT

4.1 Policy Manager

4.1.1 Preliminaries

Social Networks: Users and relationships between users are the core components of social networks. Each user maintains a user profile and is connected to a set of friends. We assume friendship is mutual, where if u_i is a friend of u_j this implies that u_j is also a friend of u_i. Each user $u_i \in V$ maintains a profile P_i which is composed of N profile attributes, $A_i = \{a_1^i, \ldots, a_N^i\}$. For example, a Facebook user profile includes attributes such as birthday, location, gender, religion, etc. Users are also able to post objects such as photos, videos and notes to their profiles to share with other users.

A social network can be modeled as an undirected graph $G(V, E)$ where the set of vertices V is the set of users and the set of edges E is the set of friendship relationships between users. The edge $(u_i, u_j) \in E$ implies that users u_i and u_j are friends. Using this model for social networks, we leverage the nodal network structural properties to provide additional user attributes. These attributes include several small world network metrics such as degree, betweenness, closeness, etc. [47, 10]. Meneely et al. [45] also use the nodal network structural properties to provide attributes for predicting code development failure rates. They leverage similar metrics (i.e., degree, closeness and betweenness) from software developer collaboration networks in their

model. Social network analysis and, more specifically, the centrality measures of degree and betweenness have also been used in the study of street gangs [61].

For a user u_i, we are able to compute M network metrics $B_i = \{b_1^i, \ldots, b_M^i\}$. Each metric provides different indicators about users in a given social network [71, 48, 47, 32].

For example:

- Degree Centrality: The number of direct friendship connections a user has.
- Betweenness Centrality: The extent to which a node lies between other nodes in the network; a node with high betweenness has great influence over what flows, and does not flow, in the network.
- Closeness Centrality: The distance between friends.
- Common Friends: The number of common friends between two users.
- Hyper-link Induced Topic Search (HITS): The measure of importance of a user based on network structure and friend relationships.
- Eigenvector Centrality: The importance of a friend relative to the friend list. PageRank is a variation of Eigenvector Centrality.

Each user u_i in a social network maintains a collection of user profile attributes and a set of user friendships of which social network metrics are computed.

`Definition 1` *(User Profile) The user profile denoted by* $P_i = \{A_i, B_i\}$ *for user* u_i *is characterized by a set of attributes* $A_i = \{a_1^i, \ldots, a_N^i\}$ *and a set of network metrics* $B_i = \{b_1^i, \ldots, b_M^i\}$. *The attribute* a_l^i *is a* (an_l^i, av_l^i) *pair of strings where* an_l^i *is the attribute name and* av_l^i *is the attribute value in a domain* D, *which also includes the null value referred to by* $\perp$. *The network metrics* B_i *follow a similar definition and their domain is* $\mathcal{R}^+$.

Policies in Social Networks: A user u_i posting an object O on their profile is allowed to setup an access control policy to specify which friends are allowed (denied) access to the posted object. The access control policy is managed and stored by the hosting social network site. Current social networks allow users to categorize friends into groups based on relationship, location, institution, family, work, etc. For example, Orkut has the following default *friend groups* defined: best friend, family, school and work. Orkut also allows its users to create new *friend groups*. Facebook has a similar capability which they call *friend lists*. Furthermore, users are able to specify policies in terms of theses groups. Users are also able to specify exception lists indicating explicitly which friends should and should not be given access to specific objects. For example, the Family photo album in Orkut can be restricted to just the Family friend group while the Graduation photo album is viewable by everyone. We define an access control policy as:

`Definition 2` *The access control policy of an object O is defined using two access control lists, namely the allow list ACL^+ and the exception list ACL^-, which are sets of the allowed and the denied users or groups respectively. Access control follows the closed world assumption, where if access is not explicitly specified, it is assumed to be not accessible. For an object O given ACL^+ and ACL^-, a user u is given access to O iff $u \in ACL^+$ and $u \notin ACL^-$ or in compact form, $u \in (ACL^+ \setminus ACL^-)$.*

4.1.2 Supervised Learning Framework

We investigated the access control mechanisms that govern access of viewers to a focus user's profile data and content. We propose Policy Manager (PolicyMgr) – a

Policy Management Assistance Tool. PolicyMgr assists users in managing access to objects posted to their profiles. Our approach leverages input from the user, profile data, metrics from their social graph and proven supervised learning techniques to effectively identify trusted and non-trusted friends relative to a specific object, e.g., photo. The profile owner provides example policy settings as training sets to build classifiers that can generate policies for other users in their friend's list – identifying them as trusted and non-trusted friends. Furthermore, we explore the fusion of policy decisions generated by their neighboring friends to enhance the accuracy of the supervised learning approach.

In most social networks, users are allowed to specify simple policies that are based on the principle of allow and deny, where the user has to decide who to allow and deny access or who to trust or not. Instead of asking the focus user to decide for each of her friends who to give access or not, our proposed framework only requires the focus user to choose the access rights (or label) of α carefully selected users from her friend's list. (Note: We refer to the profile owner as the *focus user*.) Our proposed framework uses supervised learning mechanisms to decide on the access control policy settings for the remaining users. The details of our framework are further discussed below.

In machine learning literature, a classification learning model is a function f that takes as an input a set of attributes and returns a label or classification, e.g., a function that can take the user's age, gender, credit rating and job status and generate a recommendation to either grant or deny a loan. A supervised learning mechanism uses training data Θ to learn the function f, which we refer to as f_{Θ}.

Taking a simple approach to address the policy composition problem would require each focus user to manually decide on trust and access control settings for each of his friends. This is a tedious task given that users have on average hundreds of friends [17, 65]. Instead, the approach we adopt is an enhanced user centric approach, where the user is required to only provide a small subset of their friends' permission mappings. These example permission mappings are used as a training set Θ for the supervised learning algorithm. Basically, we attempt to learn the mapping function $f_\Theta : \mathcal{X} \rightarrow \mathcal{Y}$, where:

1. $\mathcal{X}$ is a set of user profile attributes and network metrics $\{A_j, B_j\}$ describing user u_j.
2. $\mathcal{Y}$ is a set of labels $\{y_0, \ldots, y_m\}$, in our case it is $\{trusted, non-trusted\}$.
3. Θ is the training set, which is a set of labeled friends' profiles provided by the user.

Our goal is to learn the function f_Θ based on the provided training set Θ. Once f_Θ is learned, we can automatically decide if a user with a given profile is allowed or denied access. This supervised learning mechanism requires an example data set to train and guide the generation of the mapping function f_Θ. Given a friend u_j with a profile $P_j = \{A_j, B_j\}$, the classifier f_{Θ_i} for user u_i assigns the label y_l to user u_j provided that this label maximizes the classifier's confidence or the probability measure $P(u_j \rightarrow y_l | \Theta_i)$ based on the training set Θ_i.

There are five steps involved in the learning based policy management process – See Figure 8. In step 1, for each focus user's friends the profile attributes A_j are collected and the network attributes B_j are computed based on the generated social

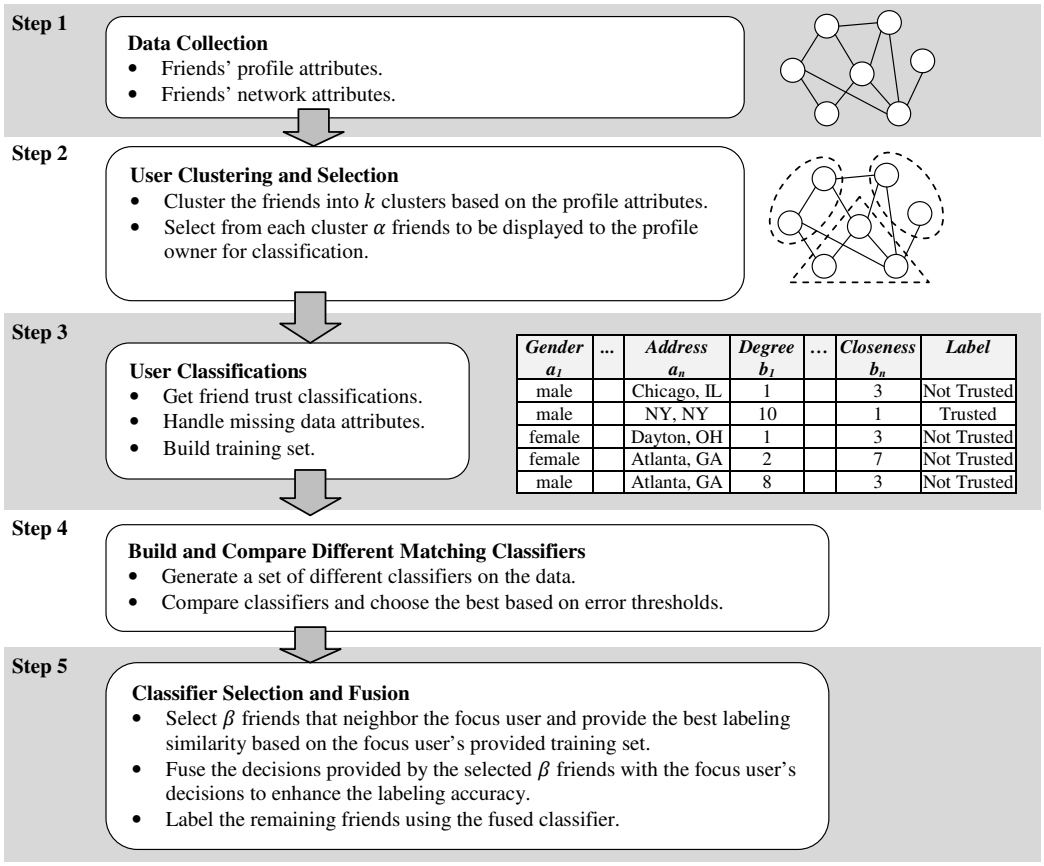


Figure 8: Learning based policy management process

graph information. In step 2, the collected attributes $\{A_j, B_j\}$ are used to cluster the focus user's friends into K non-overlapping clusters. The clustering is performed to ensure that the focus user labels representative members of each of the computed clusters. We use the k-means clustering algorithm [40]. From each of the clusters, we randomly select α friends. In step 3, The focus user is then asked to classify each of the α friends, that is to indicate which of the friends are trusted to access a specific object and which are not trusted. This implies the classification labels are selected from the set $\mathcal{Y} = \{y_0, y_1\}$, where y_0 and y_1 are trusted and non-trusted classes respectively. The labeled α friends are added to the training set Θ. In step 4, the training set Θ can be used directly to train a classifier. However, there are several

classifier algorithms and it is crucial to select the classifier that is most suited for this specific user instance. So, the mechanism we adopt is to train and tune several classifiers and then compare their performance based on standard cross validation methods such as n-fold cross validation [72]. Given m classifiers $\{f^1_{\Theta_i}, \ldots, f^m_{\Theta_i}\}$, the classifier with the highest accuracy is selected, which is denoted as $f^*_{\Theta_i}$.

In step 5, the knowledge accumulated by other users in the social network can be utilized to further enhance the classifier accuracy. It is important in this step to seek classification advice from other friends who classify users similar to the focus user. This is referred to as the *selection process* where β other user classifiers are selected based on their accuracy in labeling the focus user's training set Θ_i. In the *fusion process*, the decisions of the selected β classifiers are combined with the decisions of the focus user's classifier to classify the remaining focus user's friends. The details of this approach are discussed in the following section.

4.1.3 Classifier Selection and Fusion

The inherent advantage of social networks is the ease of sharing of news, photos, videos and several other data objects among users. We extend this sharing to include the accommodation of user experiences by leveraging their trained classifiers, where user u_j is able to share their mapping function f_{Θ_j} with other users. Assume a user u_i would like to leverage the experience of other users in the social network to improve their mapping function f_{Θ_i}. We use f_{Θ_k} to refer to the best classifier $f^*_{\Theta_k}$ for user u_k.

Given a user u_i and a set of users $S = \{u_1, \ldots, u_n\}$, the set S can be chosen from the neighboring trusted friends or from other experienced users in the social network.

Each user u_k in the set S is willing to share their mapping function f_{Θ_k} to improve the mapping function of user u_i. This translates into two sub-steps: (1) The selection of β users from the set S that are best fit to help user u_i in computing an improved mapping function, (2) The fusion of the different f_{Θ_k} functions provided by the β users with the focus user's function f_{Θ_i}.

`Definition 3` *(Selection) Given a user u_i, a set of user trained classifier functions $f_S = \{f_{\Theta_1}, \ldots, f_{\Theta_n}\}$, the training set Θ_i for user u_i and a classifier fitness function $\Phi : f_{\Theta_k} \times \Theta_i \rightarrow \Re$, select the best β classifiers based on the fitness function.*

The selection process is based on the fitness function as defined in Definition 3. The fitness function is a mechanism to rank the classifiers in f_S based on their similarity to the decisions taken by the classifier of user u_i. This mechanism attempts to locate the β classifiers that match the focus user's perspective. The fitness function tests each classifier f_{Θ_k} by labeling the tuples in the training set Θ_i and computing the vector $[TP, TN, FP, FN]^T$, which represents the true positive, true negative, false positive and false negative respectively. The fitness of f_{Θ_k} is based on the classifier accuracy [53, 7], computed as $\frac{TP+TN}{TP+TN+FP+TN}$. The β classifiers with the highest fitness are selected and are denoted by the set $S_\beta = \{f_{\Theta_1}, \ldots, f_{\Theta_\beta}\}$.

Given the β classifiers, the next step involves fusing the decisions of these classifiers and the decisions generated by the focus user's classifier (f_{Θ_i}) to improve the classification result. We adopt the following classifier fusion algorithms [31]: *group voting*, *group confidence product* and *most confident*.

After the β classifiers with the highest fitness are selected, an appropriate fusion

algorithm (of the three listed above) is chosen to fuse the results of the f_{Θ_i} and the f_{Θ_k} functions producing a predicted label, i.e., trusted or non-trusted. This final classifier function is designated as f_{Θ}^{β}. Note that when the focus user adds new friends, the generation of the their access rights can be automated based on the available function f_{Θ}^{β}. Furthermore, as the focus user adds similar objects to their profile, the new objects can also adopt the same classifier function.

4.1.4 Experimental Results

Last.FM [37] is one of the web's most popular recommendation and web radio services providing a social networking platform where friendships have an impact on what people listen to. Using Last.FM's public developer API, we implemented a crawler that was able to collect user profile attributes and friendship relationships. The crawler was written in Java 1.6; the crawler loads the focus user's profile information, their friends list and stores all the loaded data into a MySQL database. We collected about 1.6 million user profiles and about 13 million friendship links.

A series of experiments were conducted using a randomly selected subset of the Last.FM data set, approximately 200 focus users. For each focus user, the following profile attributes were obtained: Age, Gender and Home Country. We also collected Shouts, which are posts made by users on their friends' profiles. In addition, each focus user's social graph was built and a series of network metrics were computed on their respective social graph, which included degree centrality, betweenness centrality, closeness centrality, common friends, HITS and Eigenvector centrality.

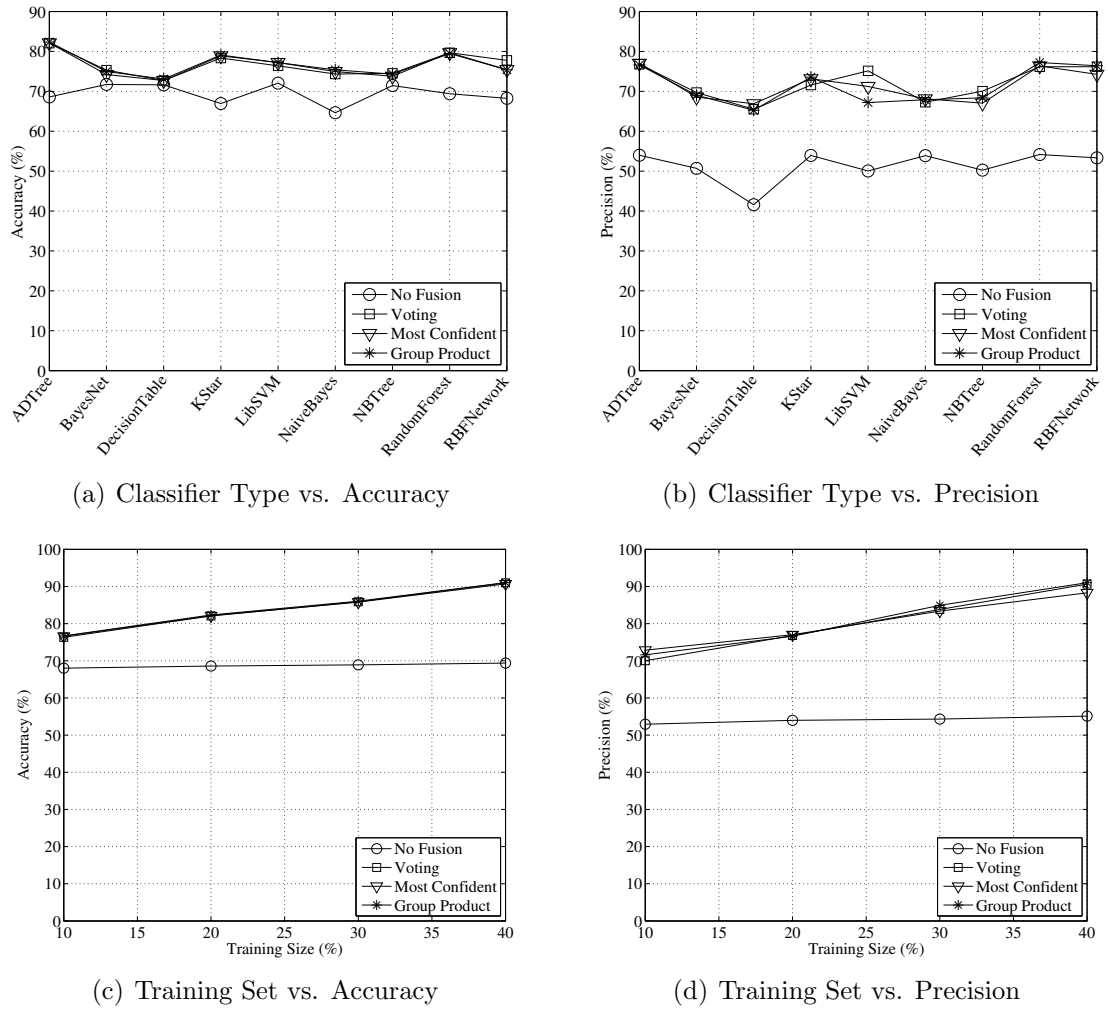
The network metrics for different users were computed using the open source Java

Universal Network Framework (JUNG) [1]. Following the collection of user data, a training set Θ was compiled out of a subset of the focus user's friend set. The overall friend set was clustered into K clusters leveraging the k-means clustering algorithm. (Note: We fixed K to two; further experiments are necessary for other values of K.) The training set Θ was comprised of a percentage α of friends randomly selected from each cluster. Each user in the training set Θ was labeled as trusted or non-trusted. The trusted and non-trusted label for each friend in the training set would normally be applied by the focus user, instead the number of Shouts generated was used as an indication of trust ($Shouts > 5$). Further user studies are planned to capture actual trust labels from the user.

The training set Θ includes, for each friend, a trusted and non-trusted label. The remaining friend set, called the test set, and the training set are inputs to a variety of different classifiers. In our experiments, nine different classifiers were trained which included the following: Naive Bayes, BayesNet, Radial Basis Function Network, K Star, AD Tree, Support Vector Machine, Naive-Bayes Tree, Random Forest and Decision Table. The classifiers were generated using the open source Java WEKA 3.6 library [2]. The classifiers were tested by labeling each friend in the test set as either trusted or non-trusted based on the users' profile attributes and network metrics. The true positive, true negative, false positive and false negatives for each classifier were recorded. The classifier with the highest accuracy is selected, which is denoted as $f^*_{\Theta_i}$.

Following the section of $f^*_{\Theta_i}$, additional classification advice is sought from the focus user's friends. For each focus user, β friends (between 10-40) were selected from the

focus user's friend set who classify users similarly as the focus user. These β additional classifiers are fused with the focus users classifiers to label the remaining friends as either trusted or non-trusted. A future enhancement will include an additional step giving the focus user an opportunity to review, and possibly re-label, each user and thus improve the overall classification results.



(a) Classifier Type vs. Accuracy

(b) Classifier Type vs. Precision

(c) Training Set vs. Accuracy

(d) Training Set vs. Precision

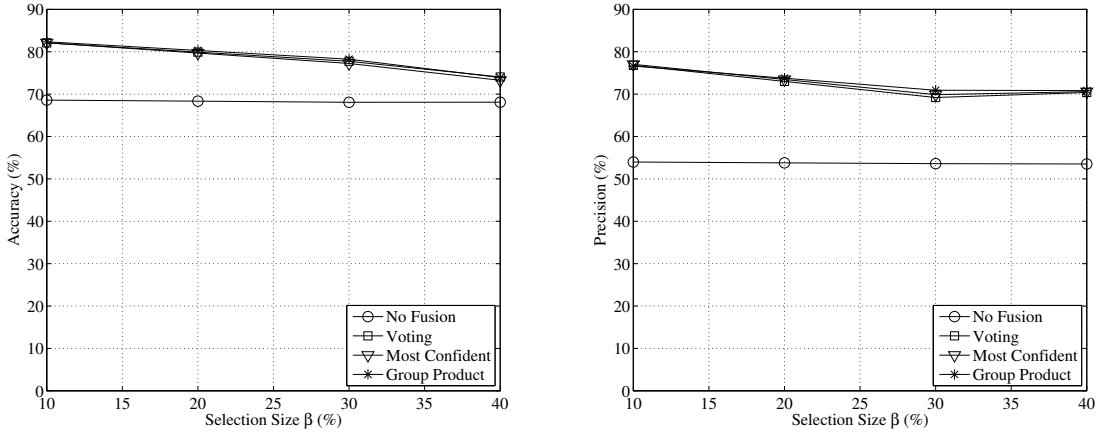
Figure 9: PolicyMgr experimental results

Figures 9(a) and 9(b) show the accuracy and precision results generated for the different classifiers using a training set (α) equal to 20% and 10 selected friends (β).

Accuracy (or correctness) was previously defined in Section 4.1.3 and precision (or reproducibility) is defined as $\frac{TP}{TP+FP}$. From these results, we are able to show that without any fusion, the classifier is capable of providing up to 70% accuracy and 55% precision. Using the different fusion mechanisms, the classifier accuracy improved to 83% and the precision increased to 78%. Based on these results, it is evident that our fusion based approach improves the classification result with the voting based approach leading the other fusion mechanisms. Furthermore, the AD Tree classifier provides the highest accuracy and precision results. The classifier results could be further improved if the user profile attributes collected were complete, as several user profile attributes were not available. Also, as previously mentioned, a future enhancement will include the presentation of the recommended label to the focus user for validation which would thus improve the overall accuracy. Our approach is an improvement over the current state of the norm of users completely ignoring their policy altogether or going through the hassle of labeling all users by hand.

Figures 9(c) and 9(d) depict the accuracy and precision of the fused classifiers (group voting, group confidence product and most confident) and the best classifier of the focus user (no fusion) holding all parameters constant (AD Tree classifier, $\beta = 10$) except for the size of the training set (α). The training set was varied from 10%-40%. The fused classifiers, performed consistently better than the no fusion case with the accuracy and precision improvements proportionally increasing with higher training set sizes. The fused classifiers, for the most part, performed similarly. There is an improvement going from 10% to 20%, i.e., if a user labels 20%, vice 10%, of his friends as trusted or non-trusted, there is improvement in the accuracy of the classifier

labeling the test set. Therefore, it is sufficient to ask the user to label between 10% to 20% of their friends in order for the policy manager to effectively label the remaining users as trusted or non-trusted.



(a) Selected β vs. Accuracy

(b) Selected β vs. Precision

Figure 10: PolicyMgr experimental results varying the selected β

To investigate the size of selected fusion classifiers (β), we conducted experiments holding all parameters constant (AD Tree classifier, $\alpha = 20\%$) while varying β. Figures 10(a) and 10(b) depict the accuracy and precision of the fused classifiers and the best classifier of the focus user (no fusion) for the different β values (10-40). Note that as we increase β, the accuracy and precision drop. This is because as the size of β increases, the fusion result is diluted with more users who have classifiers with low confidence. Note that even though the accuracy and precision drop as β increases, the fusion based classifier still consistently performs better than the no fusion classifier. If most user mapping functions f_{Θ_k} have low accuracy rates, a threshold for β can be introduced.

4.2 Assisted Friend Grouping

4.2.1 Preliminaries

Access control systems regulate the actions that subjects can take on objects. Subjects (e.g., friends) are the actors that invoke an action or access mode (e.g., read) on a user's object (e.g., privacy information like date of birth or content like a picture). An authorization is a tuple $< s, o, a >$, where s is a subject, o is an object and a is an access mode. Subjects and objects are dynamic. Friends come and go. User content is added, updated and deleted on a regular basis. Users must maintain up to date and accurate authorization lists. This can be a daunting task. If there are m subjects, n objects and p access modes, then there are potentially $m \times n \times p$ authorizations. See Figure 11. For example, if a user has 130 friends, 90 pieces of content and one access mode (read), the number of potential authorizations he must maintain is 11,700. Security administration, including granting / revoking authorizations (permissions), is very challenging.

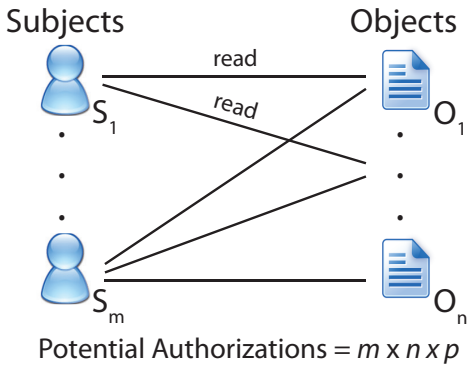


Figure 11: Subjects / objects

Many current social networking platforms offer a simple policy management ap-

proach. Security aware users are able to specify policies for their profile objects. For example, my work colleague is restricted from seeing my photos. But, my trusted best friend from school may access all my information. Facebook provides an optional mechanism that allows users to create custom *lists* to organize friends and set privacy restrictions. Similarly, Google+ allows users to create *Circles* of friends, such as family, acquaintances, etc., where the user can apply policies based on these *Circles*. Facebook also has *smart lists* which automatically group friends who live near by or attend the same school. However, managing access for hundreds of *friends* is still a very difficult and burdensome task [38]. In addition, security unaware users typically follow an open and permissive default policy. As a result, the potential for unwanted information leakage is great [4]. We believe that current capabilities to manage access to user profile information on today's social networking platforms are inadequate.

One approach that has been taken to alleviate the burden of managing large numbers of authorizations is the implementation of role based access control (RBAC) [18, 58, 59]. Role based access control introduces a role which is a container that has functional meaning, e.g., a specific job within an enterprise. Object permissions are assigned to roles. Roles are then populated with subjects who are granted the object permissions associated with the role(s) in which they belong. This level of abstraction alleviates the burden of managing large numbers of subject to object permissions assignments. See Figure 12. For the purposes of discussion, we will use the term *group* as to be synonymous with the term *role*, with the understanding that traditionally *roles* have subject to object permissions assignments and *groups* traditionally only

have subject assignments.

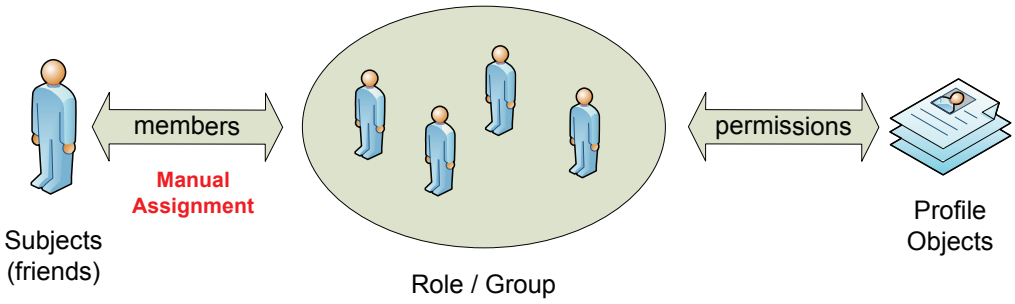


Figure 12: Role based access control

RBAC focuses on abstracting the complexity of managing large numbers of subjects for the purposes of security administration. Most attention, in access control literature, has been placed on the subject side of the equation. More limited research has focused on the object side of the equation, i.e., how to manage large sets of objects for the purposes of controlling access. The basic premise of access control systems is to protect and control access to sensitive resources / assets (i.e., objects). Just as subjects are dynamic, so are objects. Objects are created. Their properties change, e.g., sensitivity level, size, etc. They ultimately are retired or deleted. Managing the full life-cycle of large numbers of objects can be complex and difficult.

Moyer and Abamad extend RBAC by introducing Generalized Role Based Access Control [46]. GRBAC proposes environment and object roles. Environment roles describe environmental conditions on which access control decisions can be made, e.g., temporal, system load, etc. Object roles are similar to subject roles in that they provide a level of abstraction for objects. Objects can be grouped based on some common property, e.g., sensitivity level, creation date, size, etc. Thus, by grouping subjects and objects, the number of authorizations can be greatly reduced. If there

are j subject groups ($j \leq m$), k object groups ($k \leq n$) and p access modes, then there are potentially $j \times k \times p \leq m \times n \times p$ authorizations. See Figure 13. For example, if a user has 10 friend groups, 20 object groups and one access mode (read), the number of authorizations he must maintain is 200. Security administration starts to become more manageable with subject / object grouping access control models.

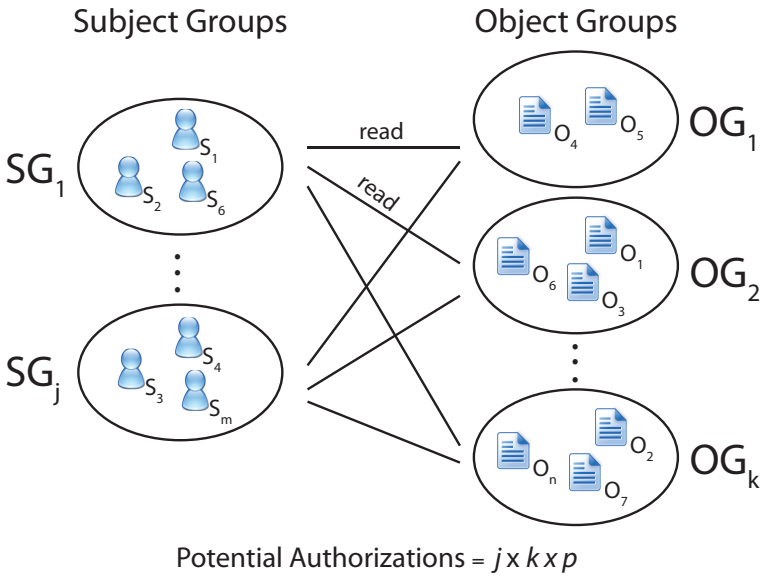


Figure 13: Subject grouping / object grouping

4.2.2 Framework

Traditional RBAC can be leveraged within social networks. Often, people's relationships drive privacy decisions. People like to specify groups for their friend relationships, in which they then can set privacy policies [25, 51]. We refer to this approach as group based policy management. However, populating relationship groups can be very time consuming and burdensome to the user [26]. We introduce a group based policy management model that assists users in placing their subjects (or *friends*) into relationship groups. Our approach leverages proven clustering techniques to aid the

user in grouping their friends more efficiently. In addition, we provide a mechanism to set friend-level exceptions within group policies. Our model is referred to as Assisted Friend Grouping – a *Policy Management Assistance Tool*.

Group based policy management allows users to populate groups based on relationship and assign object permissions to the groups, refer to Figure 12. Assisted Friend Grouping extends this model in two areas: 1) provides the user with assistance in grouping their friends, and 2) provides the user the ability to set friend-level exceptions within the group policy. See Figure 14.

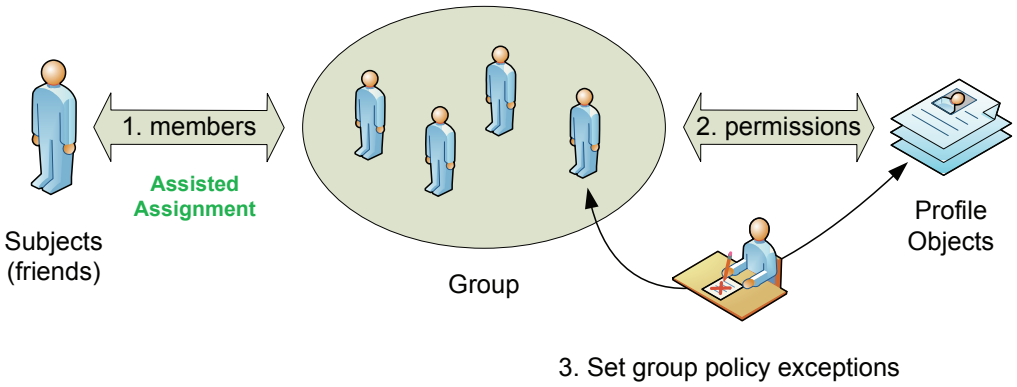


Figure 14: Assisted friend grouping model

For the purposes of our prototype Facebook application, we predefined 10 relationship groups: Family, Close Friends, Graduate School, Under Graduate School, High School, Work, Acquaintances, Friends of Friend, Community, and Other. These groups were carefully selected, in part, from the work of Jones et al. [26]. They postulate that users group their friends, for controlling privacy, based on six criteria: Social Circles, Tie Strength, Temporal Episodes, Geographical Locations, Functional Roles and Organizational Boundaries. Our friend relationship groups were selected to reflect these criteria.

Within our prototype, each friend is presented to the user in the center of a friend grouping page, refer to Figure 15. The user is asked to select, for each friend, the group that best represents their relationship. They can either "drag" the friend to the appropriate relationship group on the page. Or, the user can click the representative relationship group name. To assist users in populating their relationship groups, we leverage the Clasuet Newman Moore (CNM) network clustering algorithm [11]. This clustering algorithm analyzes and detects community structure in networks by optimizing their modularity [49]. Modularity is a metric that describes the quality of a specific proposed division of a network into communities. Our prototype clusters the user's social network graph creating CNM clusters (or groups) of friends. During friend grouping, we present the friends to the user in CNM group order as recommendations. For example, Bob has 50 friends and clustering his social network graph using CNM produces five clusters. We present to Bob, as recommendations for grouping, all the friends of one CNM group before presenting the friends of each subsequent CNM group. The premise is that CNM groups roughly align with user defined friend populated relationship groups.

By presenting friends in the order they potentially will be grouped, the friend grouping time can be vastly reduced. The user's mental model is focused on roughly one relationship at a time, e.g., work colleagues. The user can quickly ascertain that the stream of friends being presented are all work colleagues and can be placed in the *Work* group. This approach reduces the number of "mental task switches" the user must perform between multiple relationship groups. After all the friends are grouped, the user sets the group policy by setting permissions that allow or deny access to the

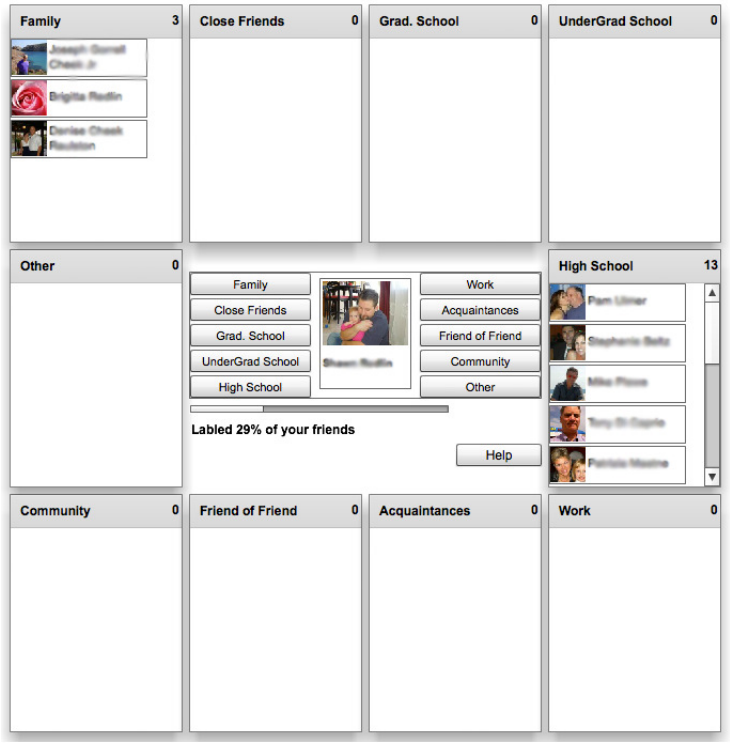


Figure 15: Assisted friend grouping user interface

user's profile objects, e.g., email address, photos, etc. Finally, we provide the user the ability to set friend-level exceptions for each group policy. For example, a group policy may deny access to the user's email address except for group member *Alice*. Most social networking platforms also provide a policy exception setting capability.

4.2.3 User Study

In designing our user study [1], we set out to answer the following research questions:

Q1. Do proven clustering techniques align with user defined relationship groups?

Q2. Can proven clustering techniques assist users in grouping their friends more efficiently?

Q3. What are users' perceptions of assisted friend grouping techniques?

[1] Approved IRB Protocol #11-08-01

4.2.3.1 Design

In order to answer these research questions, we built three tasks and one survey into a prototype Facebook application. The three tasks and the survey were designed to evaluate traditional group based policy management and our Assisted Friend Grouping Model. See Table 2.

Table 2: Assisted friend grouping user study tasks

Assisted Friend Grouping ***(Facebook Application: Assisted Friend Grouping)***	
Task 1	Group friends
Task 2	Set group policy
Task 3	Review and possibly set friend-level exceptions to group policy
Survey 1	Complete a brief survey for Tasks 1-3

In the first task (Task 1), the user is instructed to place 50 of their randomly selected friends into the ten predefined groups. We divided the user participants into two groups, namely *Not Assisted* and *Assisted*. For the Not Assisted population, the 50 friends were presented to the user for grouping in random order. For the Assisted population, the 50 friends were presented to the user for grouping in CNM group order, as described in Section 4.2.2. Friends were presented to the user for grouping based on clustering the user's social graph using the CNM algorithm. We measured the grouping time for both populations. After the user placed their friends into groups, they were asked to select access permissions for each group (Task 2). Allow/Deny permissions were selected for each profile object and/or profile object category. Finally in Task 3, the user was asked to review and possibly select friend-

level exceptions to the group policy that was set in Task 2.

Table 3: Sampling of survey questions

Ease of Use	
Question 1	It was simple to use this system.
Question 2	Overall, I am satisfied with the ease of completing the tasks.
Question 3	It was easy to learn to use this system.
Readability	
Question 4	I easily understood who had access to what in my profile.
Question 5	The information was effective in helping me complete the tasks and scenarios.
Question 6	I could understand what my friends could access in my profile.
Flexibility	
Question 7	The system had enough flexibility in allowing me to set what my friends could access in my profile.
Question 8	This system has all the functions and capabilities I expect it to have.
Question 9	I could easily set what my friends could access in my profile.

Upon completion of Tasks 1, 2 and 3, the user was asked to complete a survey. The user responded to questions designed to capture their perceptions of group based policy management, both the Not Assisted and Assisted friend grouping approaches. Table 3 provides a sampling of the questions, which were presented to the user in a different order than they actually appear in the table. The question responses are on a Likert-scale of 1 (Strongly Disagree) to 7 (Strongly Agree). Each question is designed to capture the user's perceptions in the following areas:

Ease of Use: The user needs to be able to manage their policies in an easy, intuitive and effective way such that they have a consistent experience. Complex and laborious policy management mechanisms can lead to ineffective policies.

Readability: Not only does a policy management solution have to be easy to use, it must be decipherable. The core component of any access control mechanism is the

policy which governs the access. The policy not only must be available and visible to the user, but it also must be readable. Policies that are complex and difficult to understand are more likely to be misconfigured resulting in unintended consequences, e.g., data leakage.

Flexibility: Policy management mechanisms must be flexible to accommodate the user's needs and intentions. Effective policy management must create a balance between coarse grained and fine grained access control. Traditionally, coarse grained access control provides few options to the end user. On the other hand, fine grained access control, although extremely flexible in that it provides lots of options and capabilities, is traditionally overwhelming and complex. A balance between too little flexibility and an overly burdensome policy management mechanism is needed.

4.2.3.2 Participants

We recruited our user study participants from Amazon Mechanical Turk. Amazon Mechanical Turk is a crowd sourcing marketplace that pairs *Requesters* of work and *Workers*. Requesters formulate work into Human Intelligent Tasks (HIT) which are individual tasks that workers complete. We set up our prototype Facebook application as a HIT, which included the three tasks and the survey. To better control the quality of the recruited participants, we mandated that each worker have a minimum of 100 friends and a 95% HIT approval rating, or better. The HIT took approximately 10-15 minutes to complete, for which each worker was paid a fee of $1.00.

Our user study consisted of 145 participants. See Table 4. The male / female ratio was approximately 6:4. Most of our user participants were young, fairly well

educated and active Facebook users. Approximately 54% were between the ages of 18 to 25. Almost 69% had between two and four years of college. 74% used Facebook daily. In addition, as part of the demographics portion of our survey, we collected Westin privacy sentiment information with definitions of *Unconcerned*, *Pragmatist* and *Fundamentalist* provided by [35]:

Table 4: Assisted friend grouping user study participants

	n=145	% of n
Age		
18 to 25	79	54.5%
26 to 39	59	40.7%
40 and above	7	4.8%
Gender		
male	85	58.6%
female	60	41.4%
Education Level		
$\leq$ high school	11	7.6%
2 years of college	34	23.4%
4 years of college	66	45.5%
$>$ 4 years of college	34	23.4%
Facebook Experience		
$<$ 1 year	12	8.3%
1 to 2 years	32	22.1%
2 to 3 years	44	30.3%
3 to 4 years	36	24.8%
$>$ 4 years	21	14.5%
Facebook Use		
Daily	108	74.5%
2 to 3 times per week	28	19.3%
Weekly	9	6.2%
Westin Data		
Unconcerned Users	9	6.2%
Pragmatists	106	73.1%
Fundametalists	30	20.7%

Unconcerned Users: *This group does not know what the "privacy fuss" is all about,*

supports the benefits of most organizational programs over warnings about privacy abuse, has little problem with supplying their personal information to government authorities or businesses, and sees no need for creating another government bureaucracy (a Federal Big Brother) to protect someone's privacy.

Pragmatists: *This group weighs the value to them and society of various business or government programs calling for personal information, examines the relevance and social propriety of the information sought, wants to know the potential risks to privacy or security of their information, looks to see whether fair information practices are being widely enough observed, and then decides whether they will agree or disagree with specific information activities - with their trust in the particular industry or company involved being a critical decisional factor.*

Fundamentalists: *This group sees privacy as an especially high value, rejects the claims of many organizations to need or be entitled to get personal information for their business or governmental programs, thinks more individuals should simply refuse to give out information they are asked for, and favors enactment of strong federal and state laws to secure privacy rights and control organizational discretion.*

4.2.3.3 Prototype Architecture

We implemented a prototype Facebook application called Assisted Friend Grouping. The application is hosted on our server. The back-end is based on PHP and MySQL. The client-side was implemented using Adobe Flex as a flash application. Upon installing the application, REST like Facebook API's and Facebook Query Language are used to retrieve the user's profile and social connections. The collected data

is transmitted over secure HTTPS based API's to our server and stored in a MySQL database. The application built the participant's social graph, which is clustered using the CNM implementation provided by the Flare Toolkit Library. The application implements several additional functionalities including: user grouping, group policy specification and survey tools.

4.2.3.4 Results

In evaluating our Assisted Friend Grouping Model, we set out to show that CNM will aid in grouping users' friends more efficiently for group based policy management approaches. Our hypothesis is that CNM clusters roughly align with user defined friend relationship groups. In the example illustrated in Figure 16, CNM partitions the user's social graph into distinct clusters, as depicted by the large circles. The user also categorizes their friends into user defined relationship groups, i.e., Family, Graduate School, etc. Figure 16 illustrates that there is overlap and agreement between the CNM clusters and the user defined relationship groups. We leverage this alignment by presenting friends to the user for grouping based on cluster/relationship order. By presenting friends in this manner, the user's mental model is focused on one relationship at a time. This approach results in fewer "mental task switches" between multiple relationship groups and thus improved friend grouping times.

We used the Adjusted Rand Index to measure the agreement between CNM clusters and user defined relationship groups [22]. The Adjusted Rand Index compares the predicated labels (CNM clusters) with the actual labels (user defined relationship groups) and produces an index between 0 and 1, where 0 indicates no overlap and

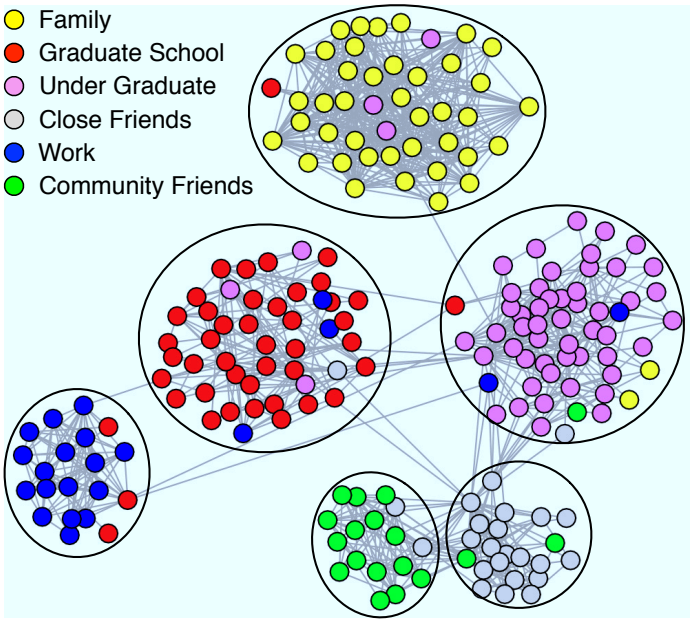
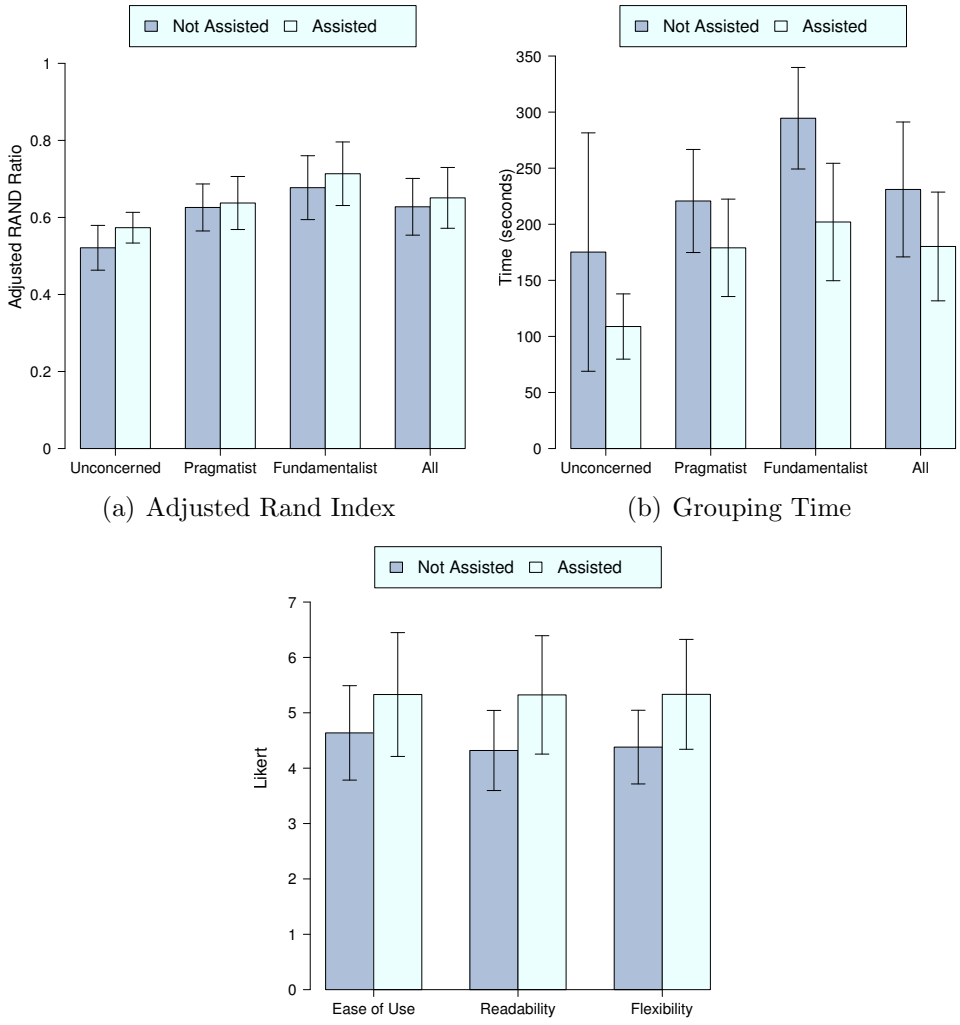


Figure 16: Example cluster/group alignment

1 is complete agreement or overlap. The Adjusted Rand Index, in general form, can be described as $\frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$. We clustered users' social graphs who were Not Assisted in grouping their friends, i.e., we presented their friend set for grouping in random order. We compared the clusters generated by CNM and the populated groups defined by the user. We found, that on average, users populated 6.3 relationship groups. Overall, our results showed an average Adjusted Rand Index of 0.627. This demonstrates that there is overlap and a level of alignment between CNM clusters and user defined relationship groups. In looking just at Fundamentalist Users, we saw a higher level of alignment (Adjusted Rand Index = 0.677).

We also wanted to determine if presenting friends in CNM group order would influence the user in how they grouped their friends. We compared the Assisted friend grouping population with those that were Not Assisted. Using a Welch Two-Sample T-Test, we found no statistical significance between the two populations ($p = 0.118$).

Refer to the Adjusted Rand Index section of Table 5 and Figure 17(a), where error bars show one standard deviation above and below the mean. Our Assisted Friend Grouping Model does not bias the user, i.e., the user would produce the same groups and populate those groups with the same friends either using our Assisted Friend Grouping approach or not.



(a) Adjusted Rand Index

(b) Grouping Time

(c) User Perceptions

Figure 17: Assisted friend grouping user study results

Next, we set out to measure the time it took a user to populate their relationship groups. We measured the time it took a user to group 50 of their friends presented

Table 5: Assisted friend grouping user study results

Measure	**Not Assisted** (μ, σ)	**Assisted** (μ, σ)	**p-value**
Adjusted Rand Index			
Unconcerned	(0.521, 0.05)	(0.573, 0.03)	0.218
Pragmatist	(0.625, 0.06)	(0.637, 0.06)	0.431
Fundamentalist	(0.677, 0.08)	(0.713, 0.08)	0.320
All	(0.627, 0.07)	(0.650, 0.07)	0.118
Grouping Time (seconds)			
Unconcerned	(175.2, 106.2)	(108.8, 29.0)	0.306
Pragmatist	(220.7, 45.9)	(178.9, 43.4)	< 0.001
Fundamentalist	(294.5, 45.2)	(202.0, 52.3)	< 0.001
All	(231.0, 60.1)	(180.2, 48.4)	< 0.001
User Perceptions (7 point Likert-scale)			
Ease of Use	(4.63, 0.85)	(5.33, 1.11)	< 0.001
Readability	(4.31, 0.72)	(5.32, 1.06)	< 0.001
Flexibility	(4.38, 0.66)	(5.33, 0.99)	< 0.001

in random order (Not Assisted). We compared that with the time it took a user to group 50 of their friends presented in CNM group order (Assisted), as described in Section 4.2.3.1. For Unconcerned Users, there was no statistical significance between Not Assisted and Assisted ($p = 0.306$). However, we did see statistical significance between the other categories of users: Pragmatists, Fundamentalists and the population as a whole – all p-values were less than 0.001. Overall, using CNM, we saw a 22% reduction in time that it took a user to group 50 of their friends, 180.2 seconds (Assisted) versus 231 seconds (Not Assisted). Refer to the Grouping Time section of Table 5 and Figure 17(b). One factor for this reduction in time is that the user's mental model is focused on one relationship group at a time, which enables the user to quickly group most family members, for example, before grouping the next set of friends. Fewer "mental task switches" between relationship groups are required

thus reducing the overall friend grouping time. It is also interesting to note, although not entirely surprising, that Fundamentalists took longer, on average, to group their friends than Pragmatists and Unconcerned Users. One possible reason that Fundamentalists took more time may be because they apply more scrutiny as they group their friends.

We also measured users' perceptions of the Not Assisted and Assisted friend grouping approaches, as described in Section 4.2.3.1. A T-Test was used to compared the Not Assisted and Assisted populations. We found statistical significance in all user perception areas: Ease of Use, Readability and Flexibility – all p-values were less than 0.001. Users found friend grouping easier to use when their friends were presented in CNM order. For Ease of Use, Not Assisted averaged 4.63 and Assisted averaged 5.33 on a 7 point Likert-scale. Readability and Flexibility also had similar results. Refer to the User Perceptions section of Table 5 and Figure 17(c). Overall, users had more positive perceptions of grouping their friends leveraging CNM than not having the assistance of CNM.

4.2.3.5 Discussion

Complex and laborious policy management mechanisms can lead to ineffective policies and compromises of information. Group based policy management is an improvement which provides a level of abstraction to the user (i.e., group) that allows them to manage permissions of large friend sets easier. However, this approach has some limitations, one being the ability to set fine grained access control policies. Introducing the capability to set friend-level exceptions to group policies overcomes this

limitation. By doing so, users have the ability to set more expressive access control policies. Another shortcoming of group based policy management approaches is the burden associated with populating relationship groups for large friends sets. Our Assisted friend grouping model alleviates this burden by reducing the amount of time it takes to populate friend groups. User perceptions of our approach are encouraging. Providing tools in the hands of the user, which assist them in managing access to their profile objects, translates into more effective privacy management.

4.3 Same-As Subject Management

4.3.1 Framework

A shortcoming of the group based policy management approach is that the user's attention (mental model) is focused in multiple areas. For example, a user must first focus on the friend's relationship in order to group them appropriately. Next, the user must change focus to the group in order to set the group-level policy. Finally, the user must switch focus back to the friend in order to set any friend-level exceptions for each group policy. We introduce an approach that overcomes this weakness. Our model leverages users' memory and opinion of their friends to set policies for other similar friends. Studies have shown that users perform more efficiently using recognition based approaches that have minimal task interruptions [12, 23]. Using our visual policy editor, a user selects a representative friend (Same-As Example Friend), assigns appropriate object permissions to this friend and then associates other similar friends to the same policy. Our model is called Same-As Subject Management – a *Policy Management Assistance Tool*. Figure 18 illustrates our model; the Same-As Example Friend is depicted in front of the user's other similar friends who have been assigned the same set of object permissions.

First, the user selects a friend (Same-As Example Friend) that is representative of a subset of their friend set. The notion is that we all have subsets of friends that have similar levels of trust. The user selects one easy to remember friend from each subset as its respective representative.

Second, using our visual policy editor, the user assigns the appropriate object level

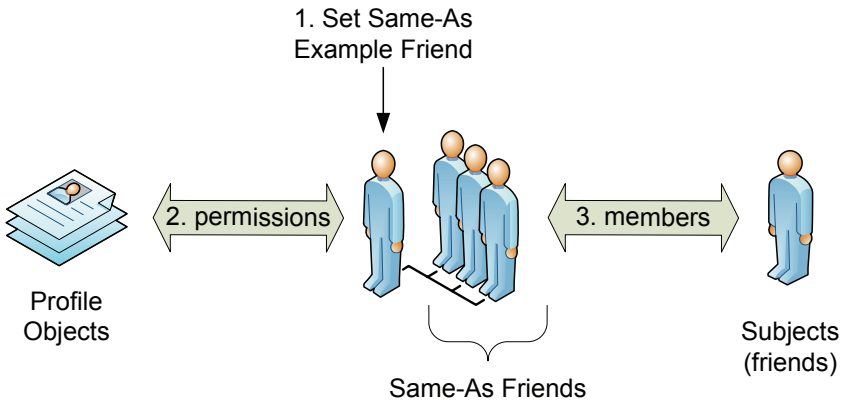


Figure 18: Same-As subject management model

permissions for each object within their profile to this Same-As Example Friend. For the purposes of our prototype Facebook application, we presented three profile object categories: *Albums*, *About Me* and *Education and Work*. Within each profile object category, objects of the same family are presented. For example, *About Me* includes Birthday, Status, Current City, email, etc., as indicated by our visual policy editor in Figure 19. The user can allow or deny access to any object or object category by simply clicking on the object or object category. For example, if the user doesn't want the Same-As Example Friend to have access to a specific photo album, they merely click on that album and the object permission is set to deny. The selected photo album will be grayed out. Or, for example, if the user doesn't want to allow access to any of their education and work information, they click on Deny for the object category *Education and Work* and the entire object category will be grayed out, thus effectively setting the permissions to deny for each profile object within that category. Any permutation of permissions is allowed.

Third, after the permissions are set for the Same-As Example Friend, other like

Figure 19: Same-As subject management user interface

or similar friends (Same-As Friends) are assigned to the policy. The visual policy editor presents to the user their friend set, where the user can associate a friend to an already defined Same-As Example Friend. Or, the user can designate a friend as a new Same-As Example Friend, thereby setting a new policy which would be assigned to other similar friends. This process repeats itself for the user's entire friend set.

4.3.2 User Study

In designing our user study, we set out to answer the following research questions:

Q4. Will a policy management approach based on leveraging a user's memory and perception of their friends outperform traditional group based policy management approaches?

Q5. Do different policy management approaches impact the conservativeness of a

user's policy?

Q6. Will users' perceptions of a policy management approach based on leveraging a user's memory and perception of their friends be higher than traditional group based policy management approaches?

4.3.2.1 Design

In order to answer these research questions, we built one task and one survey into a prototype Facebook application. The task and survey were designed to evaluate our Same-As Subject Management Model. See Table 6.

Table 6: Same-As subject management user study tasks

Same-As Subject Management ***(Facebook Application: Same-As Subject)***	
Task 4	Set permissions for friends using another friend's permissions as the model/example
Survey 2	Complete a brief survey for Task 4

The second prototype Facebook application (of which was of a similar architecture as described in Section 4.2.3.3) includes the fourth task and second survey. This task was designed to evaluate our Same-As Subject Management Model, as described in Section 4.3.1. The user was instructed, for a subset of their friends (50 randomly chosen ones), to select a Same-As Example Friend. After the user seleced their Same-As Example Friend, they then set appropriate profile object permissions for this example friend and assigned the policy to appropriate like or similar friends. This step was repeated as necessary, i.e., for as many unique policies the user would like to assign for their friend set. We measured the total time to complete Task 4. After

completing Task 4, the user completed a second survey similar to the survey presented in Section 4.2.3.1.

4.3.2.2 Participants

Similarly, as described in Section 4.2.3.2, our user study participants were recruited from Amazon Mechanical Turk. We set up our prototype Facebook application as a HIT, which included both the task and the survey, as described in Section 4.3.2.1.

Table 7: Same-As subject management user study participants

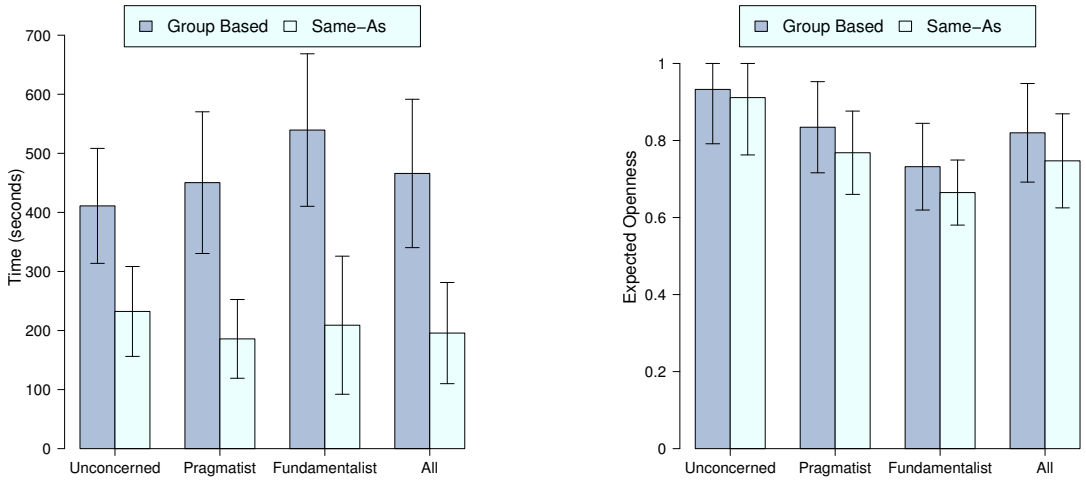
	n=153	% of n
Age		
18 to 25	92	60.1%
26 to 39	53	34.6%
40 and above	8	5.2%
Gender		
male	99	64.7%
female	54	35.3%
Education Level		
$\leq$ high school	10	6.5%
2 years of college	58	37.9%
4 years of college	52	34.0%
$>$ 4 years of college	33	21.6%
Facebook Experience		
$<$ 1 year	7	4.6%
1 to 2 years	33	21.6%
2 to 3 years	36	23.5%
3 to 4 years	46	30.1%
$>$ 4 years	31	20.3%
Facebook Use		
Daily	118	77.1%
2 to 3 times per week	31	20.3%
Weekly	4	2.6%
Westin Data		
Unconcerned Users	10	6.5%
Pragmatists	98	64.1%
Fundametalists	45	29.4%

Our user study consisted of 153 participants. See Table 7. The male / female ratio was approximately 6:4. Similarly as our previous user study, most of our user participants were young, fairly well educated and active Facebook users. Approximately 60% were between the ages of 18 to 25. Almost 72% had between two and four years of college. 77% used Facebook daily. In addition, as part of the demographics portion of our survey, we collected Westin privacy sentiment information with definitions of *Unconcerned*, *Pragmatist* and *Fundamentalist* provided by [35].

4.3.2.3 Results

We compared the policy authoring times between Group Based Policy Management (hereafter referred to as Group Based) and Same-As Subject Management (hereafter referred to Same-As Subject). Our results are summarized in the Policy Authoring Time section of Table 8 and illustrated in Figure 20(a). In analyzing these results, we found that there is statistical significance across all user categories, i.e., Unconcerned Users ($p = 0.001$), Pragmatists ($p < 0.001$) and Fundamentalists ($p < 0.001$). Overall, Same-As Subject outperformed Group Based in policy authoring time. Across the board, we observed more than a two-fold decrease in the amount of time it took a user to author their policy. One factor attributing to this reduction is the steps involved in authoring a policy. Group Based approaches have three distinct steps: 1) group friends, 2) set group policy and 3) assign friend-level exceptions to the group policy. Using this approach, the user first focuses on the friend's relationship in order to group them appropriately. Next, the user switches their attention to the group in order to set the group policy. Finally, the user switches their attention back to the

friend in order to set any friend-level exceptions to the group policy. Whereas, using our Same-As Subject approach and visual policy editor, the user simply leverages their memory and opinion of a friend to set policies for other similar friends. As a result, users can author policies in less time and thus ease the burden associated with managing their online privacy settings.



(a) Policy Authoring Time

(b) Policy Openness

Figure 20: Same-As subject management user study results

Table 8: Same-As subject management user study results

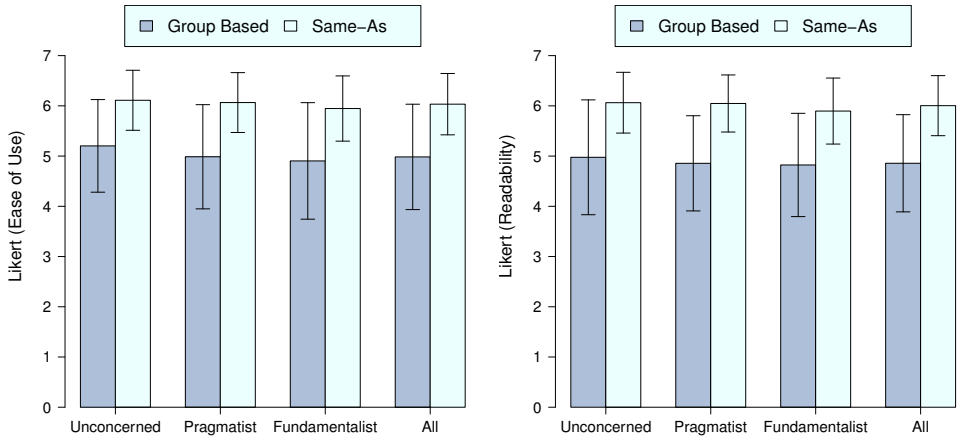
Measure	**Group Based** (μ, σ)	**Same-As Subject** (μ, σ)	**p-value**
Policy Authoring Time (seconds)			
Unconcerned	(411.0, 97.3)	(232.3, 76.1)	0.001
Pragmatist	(450.3, 119.9)	(185.8, 66.6)	< 0.001
Fundamentalist	(539.4, 129.1)	(208.9, 116.8)	< 0.001
All	(465.9, 125.6)	(195.6, 85.6)	< 0.001
Policy Openness (see Definition 4)			
Unconcerned	(0.932, 0.141)	(0.911, 0.148)	0.769
Pragmatist	(0.834, 0.118)	(0.768, 0.108)	< 0.001
Fundamentalist	(0.732, 0.112)	(0.664, 0.084)	0.018
All	(0.819, 0.127)	(0.747, 0.122)	< 0.001

Not only are users able to set their policies more rapidly using Same-As Subject, they are also setting more conservative policies, policies that are less permissive. We examined the *openness* of each user's policy, where Policy Openness is defined as:

`Definition 4` *(Policy Openness) The probability of a user permitting a friend access to a specific profile object.* $O(u,o) = \frac{|Allow(f,o)|}{|F_u|}$*, where* $Allow(f,o) \subseteq F_u$ *is the set of friends of user* u *who are allowed access to profile object* o *and* F_u *is the friend set of* u*.*

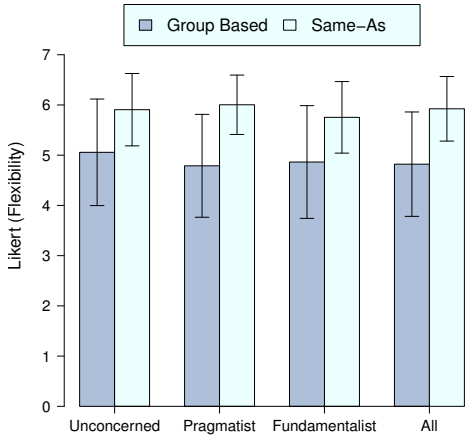
We measured Policy Openness relative to a user's profile object (i.e., email address) and found, for Unconcerned Users, no statistical significance between Group Based and Same-As Subject ($p = 0.769$). Unconcerned Users have "little problem with supplying their personal information" to others in either approach. However, we do see statistical significance between Group Based and Same-As Subject for Pragmatists ($p < 0.001$), Fundamentalists ($p = 0.018$) and for the population as a whole ($p < 0.001$). Our findings are summarized in the Policy Openness section of Table 8 and Figure 20(b). Using Group Based, users associate the policy with a group. Whereas, using Same-As Subject, users associate the policy with a friend and in doing so have the friend in the forefront of their mind. This allows users to be more selective and careful in assigning permissions. Users are thinking of people, not groups. In addition, as would be expected, our results show that Fundamentalists write more conservative policies than Pragmatists and Unconcerned Users.

Overall, users found Same-As Subject easier to use than Group Based, 6.03 versus 4.98 on a 7 point Likert-scale, where 7 is Strongly Agree. We found statistical sig-



(a) Ease of Use

(b) Readability



(c) Flexibility

Figure 21: Same-As subject management user study results – perceptions

nificance in our comparison ($p < 0.001$). Refer to Figure 21(a) and the Ease of Use section of Table 9. Using Same-As Subject over Group Based, we observed statistical significance and improved Ease of Use ratings for all user categories: Unconcerned Users, Pragmatists and Fundamentalists. We attribute the improved ratings to reasons similar to what was discussed with regard to the reduction in policy authoring time: reduced number of steps for authoring policies, our visual policy editor and consistent focus with limited memory interruption. It is interesting to note that Unconcerned Users averaged Ease of Use ratings higher than Pragmatists and Fundamentalists. Unconcerned Users don't necessarily care much about privacy and appreciate mechanisms that are easier. Fundamentalists find privacy to be "hard" regardless of approach and Pragmatists fall somewhere in the middle.

Table 9: Same-As subject management user study results – perceptions

Measure	**Group Based** (μ, σ)	**Same-As Subject** (μ, σ)	**p-value**
Ease of Use (7 point Likert-scale)			
Unconcerned	(5.20, 0.92)	(6.11, 0.59)	0.046
Pragmatist	(4.98, 1.03)	(6.06, 0.59)	< 0.001
Fundamentalist	(4.90, 1.15)	(5.94, 0.64)	< 0.001
All	(4.98, 1.04)	(6.03, 0.61)	< 0.001
Readability (7 point Likert-scale)			
Unconcerned	(4.97, 1.14)	(6.06, 0.60)	0.049
Pragmatist	(4.85, 0.94)	(6.04, 0.56)	< 0.001
Fundamentalist	(4.82, 1.02)	(5.89, 0.65)	< 0.001
All	(4.85, 0.96)	(6.00, 0.59)	< 0.001
Flexibility (7 point Likert-scale)			
Unconcerned	(5.05, 1.06)	(5.90, 0.72)	0.095
Pragmatist	(4.78, 1.02)	(6.00, 0.59)	< 0.001
Fundamentalist	(4.86, 1.12)	(5.75, 0.71)	0.002
All	(4.82, 1.03)	(5.92, 0.64)	< 0.001

Users found Same-As Subject to be substantially more readable than Group Based. There is statistical significance across all user categories. Refer to Figure 21(b) and the Readability section of Table 9. We attribute these high ratings to the simplicity of the Same-As Subject approach. Users could easily understand who had access to what profile object. Users found the organization of the information on the screen to be decipherable and ease to read. Using Same-As Subject and leveraging our visual policy editor, a user need only to recall their opinions of their friends in order to set access control policies. This was accomplished all on one screen. Whereas, the Group Based approach was more complex with multiple steps and screens.

In evaluating Flexibility, on average, users gave higher ratings to Same-As Subject over Group Based, 5.92 versus 4.82. We found statistical significance for Pragmatists, Fundamentalists and the population as a whole. However, we didn't find significance between the two approaches for Unconcerned Users. Refer to Figure 21(c) and the Flexibility section of Table 9. In access control terms, both Group Based and Same-As Subject have similar expressive power. That is, users can compose policies of the same granularity with either Group Based or Same-As Subject. Group Based allows finer grained policies with the inclusion of friend-level exceptions to group policies. Same-As Subject inherently has this capability and was perceived to be more flexible.

4.3.2.4 Discussion

Same-As Subject Management further improves upon group based policy management. It provides a similar level of expressive power for setting fine grained policies. But, doing it in a way that is easier for the user to manage and intuitively easier

to comprehend. Using our visual policy editor, users can compose readable policies that are not complex and difficult to understand. In addition, users can compose these policies in less than half the time it takes traditional group based policy management approaches. Policy management becomes less of a laborious and tedious task and results in more properly configured and maintained policies, which leads to improved privacy. In addition, users are authoring more conservative policies, which ultimately provide better levels of protection. Same-As Subject Management keeps users more informed, improves the adoption and accuracy of access control policies and, ultimately, improves user security.

4.4 Example Friend Selection

4.4.1 Framework

Same-As Subject Management is an improvement over Group Based Policy Management. To further improve Same-As Subject Management, we propose two approaches for recommending Same-As Example Friends to the user, which we call Example Friend Selection – a *Policy Management Assistance Tool*. Both approaches, called *CNM Order* and *Sample CNM*, leverage the Clasuet Newman Moore (CNM) network clustering algorithm for clustering the user's social network graph.

In CNM Order, we present the user's friends in CNM cluster order, i.e., all the friends in Cluster #1 are presented to the user followed by all the friends in Cluster#2, etc. The first friend presented for each cluster is the friend with the highest degree in that cluster. The premise is the highly connected friends are potentially more well known and thus easier to remember making them good candidates for Same-As Example Friends. For example, Figure 22 illustrates a user's social network graph that has three CNM clusters of friends. Friend A has the highest degree in Cluster #1 and, therefore, Friend A is presented to the user first as a recommendation for a Same-As Example Friend. After Friend A is presented to the user, the remaining friends of Cluster #1 are presented for association with an already defined Same-As Example Friend or for assignment as a new Same-As Example Friend. After all of Cluster #1 friends are presented, Cluster #2 friends are presented in a similar fashion, i.e., Friend L has the highest degree in Cluster #2 and thus is presented to the user as a possible candidate for a Same-As Example Friend followed by the remainder of

the friends in Cluster #2. This same process is repeated for all clusters.

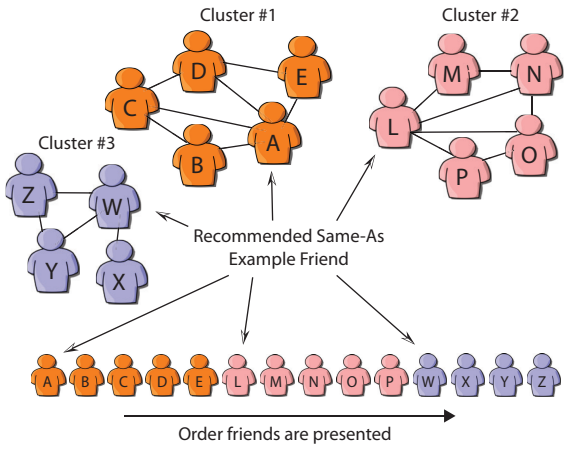


Figure 22: CNM order model

The premise is by presenting the friends in CNM cluster order, the user can set the policy for the Same-As Example Friend and then associate all other similar friends with this Same-As Example Friend. The user's mental model is focused on one Same-As Example Friend at a time. After the policy is set for the Same-As Example Friend, the user can quickly ascertain that the stream of friends that follow may potentially be associated with this Same-As Example Friend. By presenting friends is this manner, policy authoring time can potentially be reduced.

In our second approach for assisting users in selecting their Same-As Example Friend, called Sample CNM Order, we present all of the friends with the highest degree within their cluster first. These friends are highly connected and are potentially more well known and thus easier to remember making them good candidates for Same-As Example Friends. Using the example social network graph depicted in Figure 23, Sample CNM Order will present Friends A, L and W first followed by the remainder of the friends from Cluster #1, followed by the reminder of the friends from Cluster

#2 and then the remainder of the friends from Cluster #3. In Sample CNM Order, users enable their policies globally followed by policy assignment for each of their friends. The premise of this approach is that the user will set all their policies for all their Same-As Example Friends first and then quickly associate the stream of friends that follow with their respective Same-As Example Friend, thus potentially reducing policing authoring time.

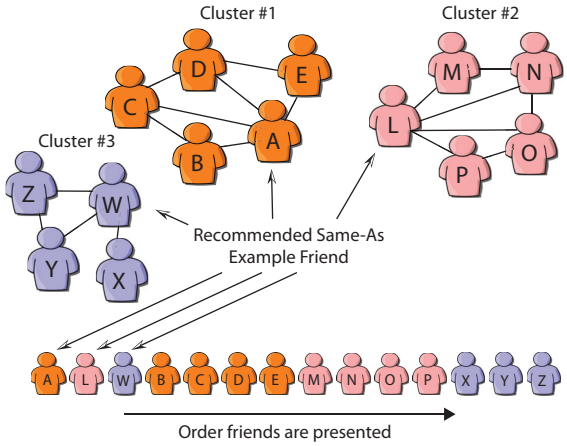


Figure 23: Sample CNM order model

4.4.2 User Study

In designing our user study, we set out to answer the following research question:

Q7. Can different friend selection techniques effectively aid users in picking example friends that are used in developing policy templates?

4.4.2.1 Design

We leveraged the same user study design and prototype Facebook application as described in Section 4.3.2.1, with one addition. The visual policy editor was modified to use three approaches for assisting users in selecting their Same-As Example Friend:

Random, CNM Order and Sample CNM Order. Random presents friends to the user in random order.

We also leveraged the same user participants as described in Section 4.3.2.2. They were divided into three groups, namely Random, CNM Order and Sample CNM Order. For the Random population, the 50 friends were presented to the user in random order. For the CNM Order and Sample CNM Order populations, the 50 friends were presented to the user in CNM Order and Sample CNM Order respectively, as described in Section 4.4.1.

4.4.2.2 Results

We evaluated the three approaches used by Same-As Subject Management for assisting users in selecting their Same-As Example Friend. Using analysis of variance (ANOVA), we measured the effects of the three approaches. Our results are summarized in Table 10.

Table 10: Random vs. CNM order vs. sample CNM order

Measure	**Random (control)** μ	**CNM Order** μ	**Sample CNM** μ	$F - Statistic$ $p - value$
Policy Authoring Time (seconds)				
	250.0	192.3	149.2	F(2,150)=21.65 $p < 0.001$
User Perceptions (7 point Likert-scale)				
Ease of Use	5.67	6.04	6.35	F(2,150)=18.68 $p < 0.001$
Readability	5.66	5.97	6.34	F(2,150)=19.67 $p < 0.001$
Flexibility	5.58	5.90	6.26	F(2,150)=16.52 $p < 0.001$

In evaluating authoring time, we observed a 23% reduction in the time it took a user to author a policy leveraging CNM Order (192.3 seconds) versus Random (250 seconds). Figure 24 displays the policy authoring time results in the form of box plots, where the top and bottom of the boxes are the first and third quartiles respectively and the band near the middle of the box is the median. We see statistical significance amongst the three groups ($p < 0.001$) with the F-Statistic (21.65) greater than 3.06 for a probability of 95%. We also ran a pairwise comparison leveraging the Bonferroni correction where we observed statistical significance across all pairings.

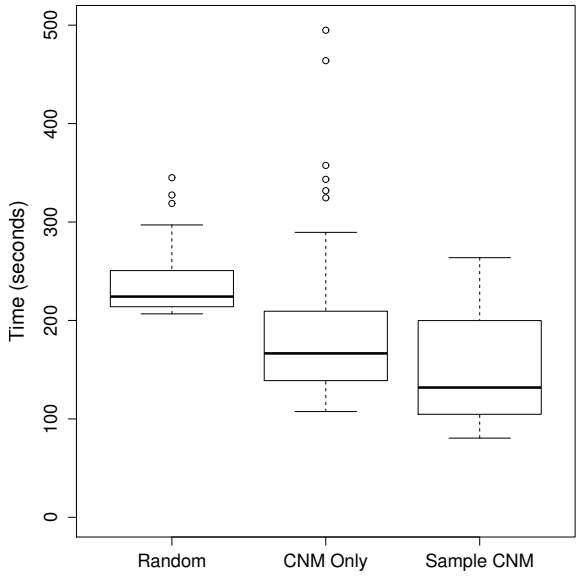


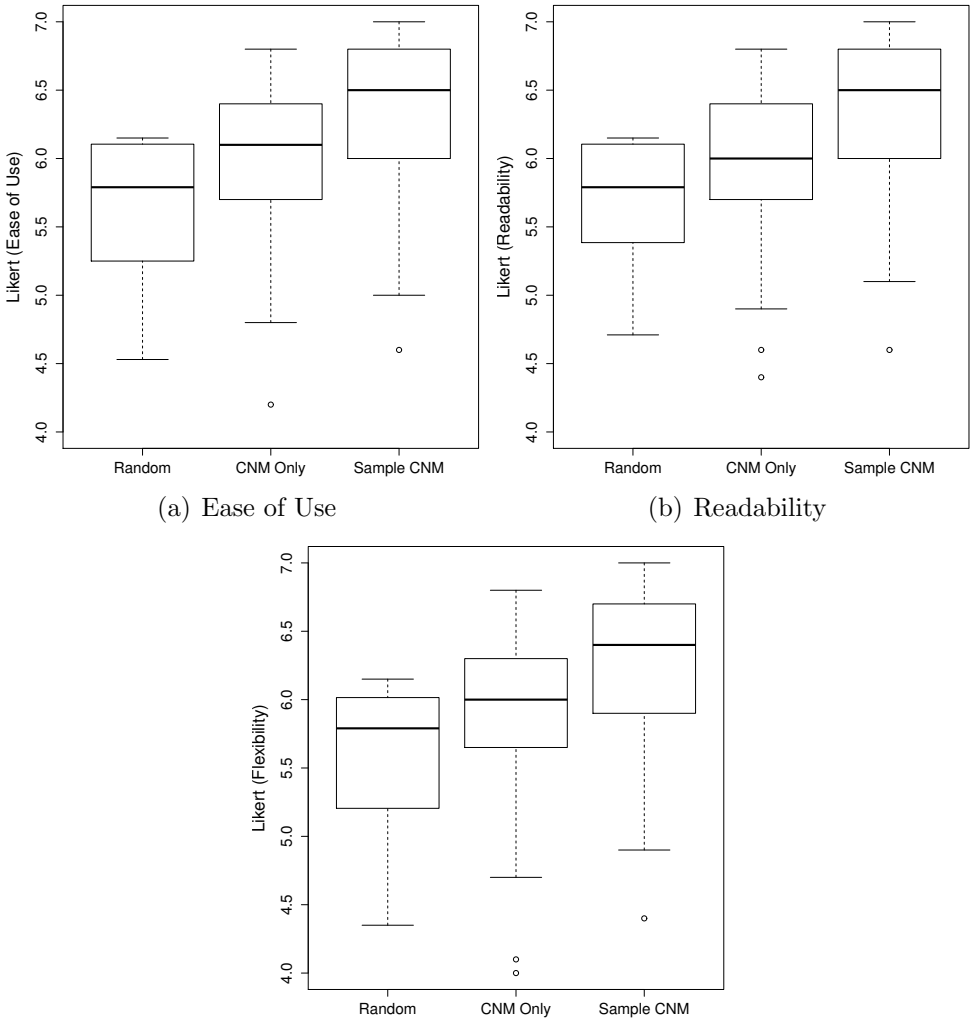
Figure 24: Policy authoring time

CNM Order allows users to author policies faster because we recommend highly connected friends as Same-As Example Friends. The most highly connected friend of a cluster is presented first and is more likely to be selected as a Same-As Example Friend. This highly connected friend is potentially more well known and thus easier to remember making them good candidates for Same-As Example Friends. After the

policy is set, the stream of friends presented next are of the same cluster and potentially the same relationship group and policy template. The user's mental model is focused on one Same-As Example Friend where they can quickly associate, if appropriate, the stream of friends that follow with this Same-As Example Friend. This process repeats itself for each of the user's clusters.

Sample CNM Order (149.2 seconds) outperformed CNM Order with a 22% reduction in policy authoring time. In addition, Sample CNM Order outperformed Random with a 40% reduction in policy authoring time. With Sample CNM Order, all the user's clusters' most highly connected friends are presented first for policy authoring and then the remaining members of each cluster are presented in cluster order for association with the appropriate Same-As Example Friend. With this example friend selection technique, the user sets all their policy templates (Same-As Example Friends) first and then associates appropriate friends with each policy template. Users were able to author policies much faster leveraging this technique over Random and CNM Order.

In measuring user perceptions of the three approaches for selecting the Same-As Example Friend, we observed that Sample CNM Order was more positively perceived than Random and CNM Order. Sample CNM Order was found to be easier to use (6.35 on a 7 point Likert-scale), more readable (6.34) and more flexible (6.26). See Figure 25. We found statistical significance ($p < 0.001$) across the three areas measured (Ease of Use, Readability, Flexibility) with all F-Statistics greater than 3.06 for a probably of 95%. We also ran a pairwise comparison leveraging the Bonferroni correction where we observed statistical significance across all pairings. Sample CNM



(a) Ease of Use

(b) Readability

(c) Flexibility

Figure 25: Random vs. CNM order vs. sample CNM order

Order, where the user authors all the policies for their Same-As Example Friends first, outperformed both Random and CNM Order for both policy authoring time and user perceptions.

4.4.2.3 Discussion

In evaluating different friend selection techniques for aiding users in picking example friends (Same-As Example Friends), we found that both techniques introduced (CNM Order and Sample CNM Order) outperformed the Random approach. Each new technique reduced the time it took to author a policy. In addition, users' perceptions were higher over the Random approach. Presenting friends in CNM cluster order using either technique (CNM Order or Sample CNM Order) potentially cuts down on the amount of "searching" a user must do to find the "right" Same-As Example Friend. We present the friends in an order that is potentially meaningful to the user. As such, we would expect to have faster policy authoring times and improved user perceptions.

In evaluating Sample CNM Order versus CNM Order, we see the former outperforming the latter in both policy authoring time and user perceptions. Sample CNM Order allows a user to build their global policy set upfront. After which, they can quickly assign appropriate friends to each policy. In leveraging this approach, users were able to author policies more quickly than the other two approaches. In addition, users had higher perceptions of Sample CNM Order.

4.5 Same-As Object Management

4.5.1 Framework

Same-As Object Management leverages the same basic principles as Same-As Subject Management but in reverse. With Same-As Object Management (a *Policy Management Assistance Tool*), the user first groups their subjects. After which, the user is asked to select a representative object (Same-As Example Object), set subject group permissions for this Same-As Example Object and assign other similar objects the same set of subject group permissions. This process is repeated for each of the user's representative objects (Same-As Example Objects). Figure 26 illustrates our model; the Same-As Example Object is depicted in front of the user's other similar objects who have been assigned the same set of subject group permissions.

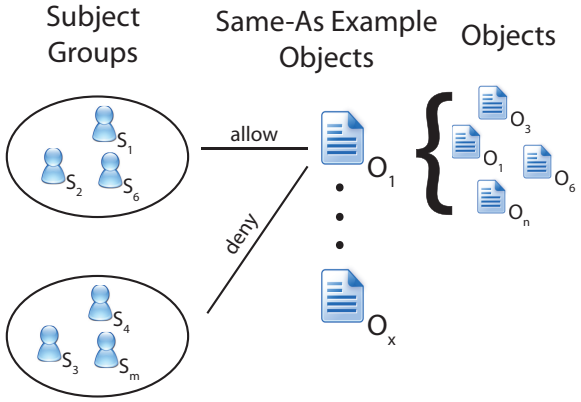


Figure 26: Same-As object management model

For the purposes of our prototype Facebook application, the user is asked to group approximately 30 of their randomly selected friends into ten predefined friend groups: *Family, Close Friends, Graduate School, Under Graduate School, High School, Work, Acquaintances, Friends of Friend, Community, and Other*. These groups were care-

fully selected, in part, from the work of Jones et al. [26]. They postulate that users group their friends, for controlling access to privacy information and content, based on six criteria: Social Circles, Tie Strength, Temporal Episodes, Geographical Locations, Functional Roles and Organizational Boundaries. Our friend groups were selected to reflect these criteria.

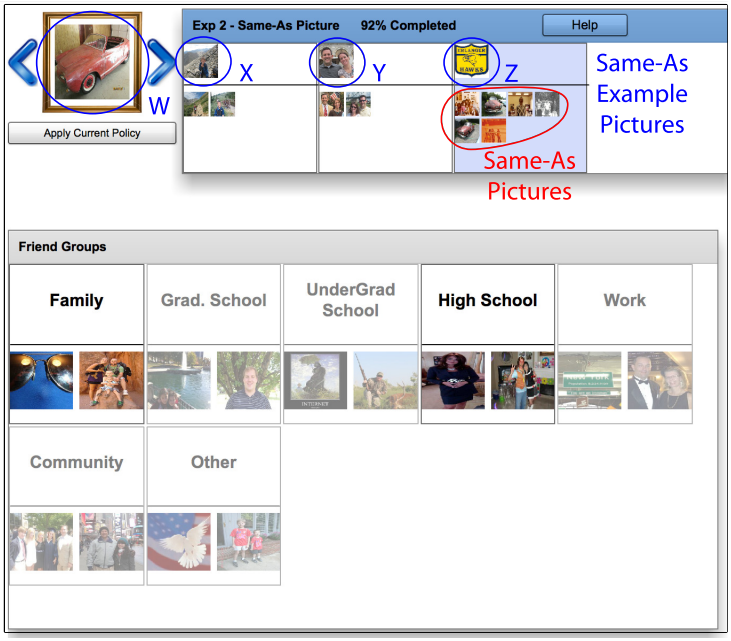


Figure 27: Same-As object management user interface

Using our visual policy editor, the user selects a picture (Same-As Example Picture) that is representative of a subset of their picture set. See larger blue circle labeled "W" in Figure 27. This Same-As Example Picture represents some subjective meaning of sensitivity to the user, of which other pictures of a similar sensitivity level can be associated. For example, Bob may select a picture that depicts him drinking alcohol as the Same-As Example Picture that he uses to associate other similar pictures of him drinking alcohol. Bob considers pictures of him drinking alcohol to be of a certain sensitivity level and wants to limit who can view these pictures. The Same-As

Example Picture should be easy to remember and is the representative for other like or similar pictures.

Next, the user assigns the appropriate friend group permissions for this Same-As Example Picture. The user can allow or deny access to any friend group by simply clicking on the group to toggle between allow and deny permissions. If the user doesn't want a specific friend group to have access to the Same-As Example Picture, they merely click on that friend group and the group will be grayed out. This indicates that access is not allowed. For example, Bob may allow his *Family* group to view pictures of him drinking alcohol, but deny viewing rights for his *Work* group. The default permissions are set to deny access.

After the permissions are set for the Same-As Example Picture, other like or similar pictures (Same-As Pictures) are assigned to the policy. The visual policy editor presents to the user their picture set, where the user can associate a picture to an already defined Same-As Example Picture. Or, the user can designate a picture as a new Same-As Example Picture, thereby setting a new policy, which would be assigned to other similar pictures. This process repeats itself for the user's entire picture set. For example in Figure 27, our visual policy editor depicts Same-As Example Picture "Z" as having access to the *Family* and *High School* friend groups. (The remaining friend groups are grayed out and, therefore, access is denied). All the Same-As Pictures circled in red inherit these same permissions.

4.5.2 User Study

In designing our user study, we set out to answer the following research questions:

Q8. Does Same-As Object Management allow for more expressive policies?

Q9. Does Same-As Object Management outperform other policy management approaches?

Q10. Does Same-As Object Management lead to more conservative policies?

Q11. What are user's perceptions of Same-As Object Management compared to other policy management approaches?

4.5.2.1 Design

In order to answer these research questions, we designed a within subjects user study consisting of three experiments. To avoid ordering bias, the experiments were presented in random order to the study participants. These three experiments were implemented as part of our prototype Facebook application.

Experiment 1: The first experiment was designed to evaluate traditional subject / object grouping access control models (refer back to Figure 13). This experiment has three tasks, as indicated in Table 11. First, the user was instructed to group approximately 30 of their randomly selected friends into ten predefined friend groups. After which, the user was asked to group approximately 15 of their randomly selected pictures in up to four predefined and four user defined (optional) sensitivity groups. Finally, the user was asked to select allow / deny access permissions specifying which subject groups have access rights to each object group. We measured how many unique policy templates the user created, how long the user took to author all their policies (to include grouping activities) and the conservativeness of their policy set.

Upon completion of the three tasks for Experiment 1, the user completed a brief survey designed to capture their perceptions of subject / object grouping access control models.

Table 11: Same-As object management user study experiments

Experiment 1 – Subject / Object Grouping	
Task 1	Group subjects
Task 2	Group objects
Task 3	Set permissions
Survey 1	Complete a brief survey for Tasks 1-3
Experiment 2 – Same-As Subject Management w/ Object Grouping	
Task 4	Group objects
Task 5	Set permissions for subjects using another subject's permissions as the model / example
Survey 2	Complete a brief survey for Tasks 4-5
Experiment 3 – Same-As Object Management	
Task 6	Group subjects
Task 7	Set permissions for objects using another object's permissions as the model / example
Survey 3	Complete a brief survey for Tasks 6-7

Experiment 2: The second experiment was designed to evaluate Same-As Subject Management with Object Grouping. Same-As Subject Management, as described in Section 4.3.1, is extended to allow for object grouping. Most online social networks provide some means for grouping subjects, e.g., Facebook *Friend Lists* and Google+ *Circles*. However, most do not provide a means for grouping objects. With Same-As Subject Management with Object Grouping, objects can be grouped by their specific properties, e.g., size, creation date, type, event, location, sensitivity level, etc. Once the objects are grouped, the user is asked to select a representative subject (Same-As Example Subject), set object group permissions for this Same-As Example Subject

and assign other similar subjects the same set of object group permissions. This process is repeated for each of the user's representative subjects (Same-As Example Subjects). Figure 28 illustrates our model; the Same-As Example Subject is depicted in front of the user's other similar subjects who have been assigned the same set of object group permissions.

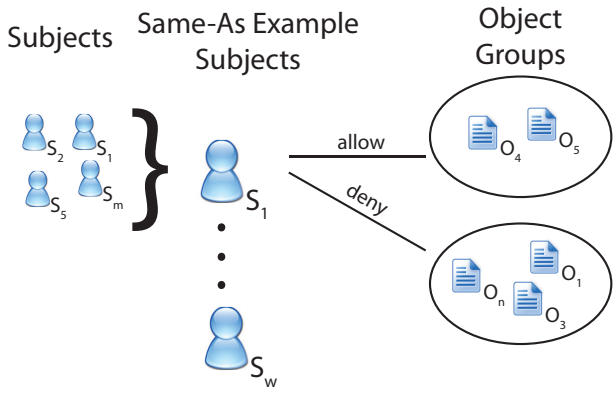


Figure 28: Same-As subject management model w/ object grouping

For the purposes of our prototype Facebook application, the user is asked to group approximately 15 of their randomly selected pictures into four predefined and up to four more user defined object sensitivity groups. The predefined groups are labeled *Public* (Viewable by everyone), *Private* (Viewable by most, but not all), *Sensitive* (Viewable by select friends) and *Highly Sensitive* (Viewable by a very select small set of friends). These labels were designed in part from the classification schemes used by many governmental organizations (e.g., *Unclassified*, *Confidential*, *Secret* and *Top Secret*) and those used in the commercial sector, e.g., as described in RFC 3114. Using our visual policy editor, the user selects a friend (Same-As Example Friend) that is representative of a subset of their friend set. See larger red circle labeled "A" in Figure 29. The notion is that we all have subsets of friends that have similar

levels of trust. The user selects one easy to remember friend from each subset as its respective representative.

Next, the user assigns the appropriate sensitivity group permissions for this Same-As Example Friend. The user can allow or deny access to any sensitivity group by simply clicking on the group to toggle between allow and deny permissions. For example, if the user doesn't want the Same-As Example Friend to have access to a specific sensitivity group, they merely click on that sensitivity group and the group will be grayed out. This indicates that access is not allowed. The default permissions are set to deny access.

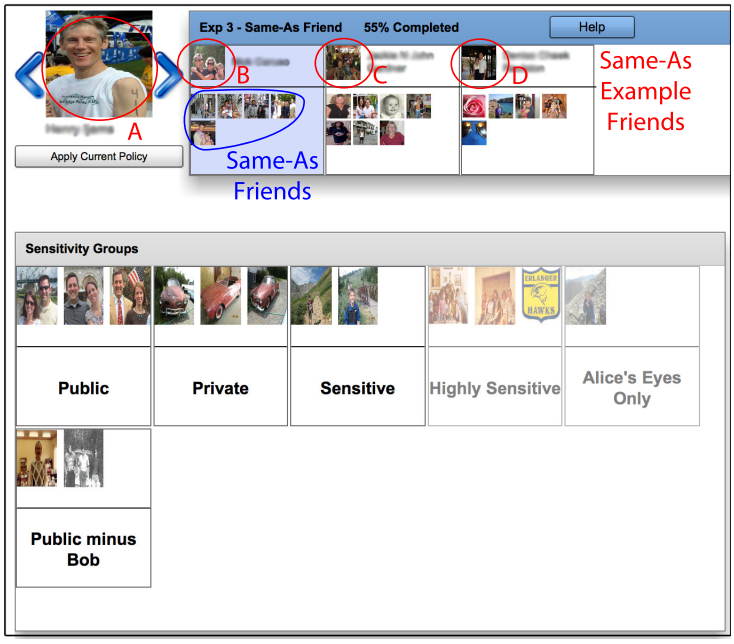


Figure 29: Same-As subject management user interface w/ obj grouping

After the permissions are set for the Same-As Example Friend, other like or similar friends (Same-As Friends) are assigned to the policy. The visual policy editor presents to the user their friend set, where the user can associate a friend to an already defined Same-As Example Friend. Or, the user can designate a friend as a new Same-As

Example Friend, thereby setting a new policy, which other similar friends would be assigned. This process repeats itself for the user's entire friend set. For example in Figure 29, our visual policy editor depicts Same-As Example Friend "B" as having access to the following sensitivity groups: *Public*, *Private*, *Sensitive* and *Public minus Bob*. (*Highly Sensitivity* and *Alice's Eyes Only* are grayed out and, therefore, access is denied). All the Same-As Friends circled in blue inherit these same permissions.

We measured how many unique policy templates the user created, how long the user took to author all their policies (to include grouping activities) and the conservativeness of their policy set. After completing Tasks 4 and 5, the user completed a second survey identical to the survey presented in Experiment 1.

Experiment 3: The third experiment was designed to evaluate Same-As Object Management, as described in Section 4.5.1. This experiment also has two primary tasks. See Table 11. In the first task (Task 6), the user was instructed to group approximately 30 of their randomly selected friends in up to ten predefined friend groups. In the second task (Task 7), the user was instructed, for a subset of their pictures (approximately 15 randomly chosen ones), to select a Same-As Example Picture, set appropriate allow / deny access permissions for this example picture and assign the policy to appropriate like or similar pictures. This step was repeated as necessary, i.e., for as many unique policies the user would like to assign for their picture set. We measured how many unique policy templates the user created, how long the user took to author all their policies (to include grouping activities) and the conservativeness of their policy set. After completing Tasks 6 and 7, the user completed a third survey identical to the survey presented in Experiment 1.

Similarly, as described in Section 4.2.3.2, our user study participants were recruited from Amazon Mechanical Turk. We set up our prototype Facebook application as a HIT, which included the three experiments as described in Section 4.5.2.1. Of the 99 participants, 61 were male and 38 were female. Most of our user participants were young, fairly well educated and active Facebook users. 70% were between the ages of 18 to 25. 80% had between two and four years of college. Almost 87% used Facebook daily. In addition, as part of the demographics portion of our survey, we collected Westin privacy sentiment information. 9% of our participants were classified as Unconcerned Users. 64% were Pragmatists and 27% were classified as Fundamentalists.

4.5.2.2 Results

We evaluated the three access control models: Subject / Object Grouping – hereafter referred to as Group Based (Experiment 1), Same-As Subject Management with Object Grouping – hereafter referred to as Same-As Subject w/ Obj Gping (Experiment 2) and Same-As Object Management – hereafter referred to as Same-As Object (Experiment 3). Using analysis of variance (ANOVA), we measured the effects of the three approaches. For the Unconcerned user population, we observed no statistical significance for the three experiments across all measurements. The remaining results are summarized in the tables and figures that follow.

Number of Policy Templates: We measured how many unique policy templates a user created as part of each experiment. A policy template is a collection of authorizations created by the user with semantic meaning established by the user. For Group Based, unique policy templates equates to the number of sensitivity groups the user

leverages. For Same-As Subject w/ Obj Gping, unique policy templates equates to the number of Same-As Example Friends the user creates and similarly for Same-As Object, where the number of unique policy templates equates to number of Same-As Example Pictures the user creates.

For Fundamentalists, there is no statistical significance ($p = 0.06$) with regard to the number of policy templates the user generates. See Number of Policy Templates section of Table 12. We do see statistical significance for Pragmatists ($p < 0.01$) and the population as a whole ($p < 0.01$). The *F-Statistics* are greater than 3.04 (Pragmatist) and 3.02 (All) for a probability of 95%. We also ran a pairwise comparison leveraging the Bonferroni correction and observed no statistical significance between Same-As Subject w/ Obj Gping and Same-As Object. However, we do see statistical significance between Group Based and Same-As Subject w/ Obj Gping and Group Based and Same-As Object. Table 13 summarizes the results of the pairwise comparison. Same-As Subject w/ Obj Gping and Same-As Object create approximately four policy templates versus approximately three for Group Based. Figure 30(a) displays the number of policy templates by experiment in the form of a box plot, where the top and bottom of the box is the first and third quartiles respectively and the band near the middle of the box is the median.

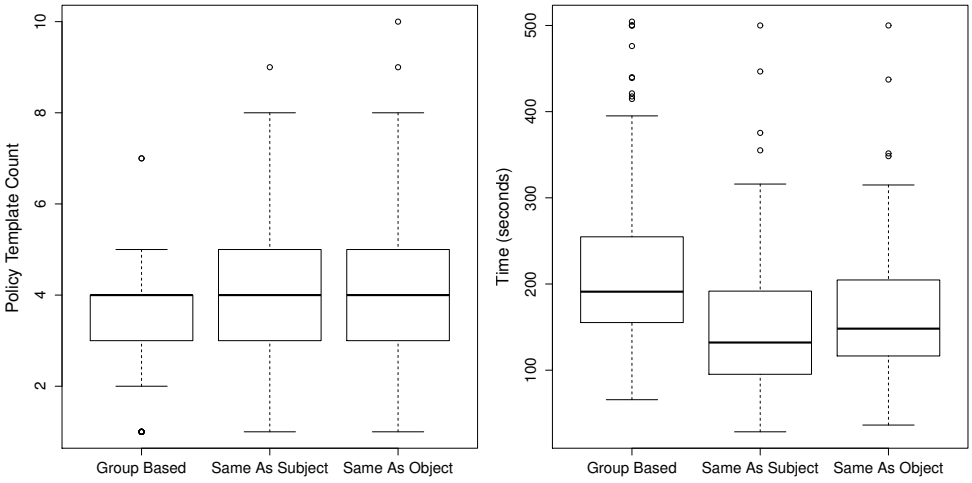
Policy Authoring Time: Next, we set out to measure how long it took a user to author all their policies for each experiment. For Group Based, policy authoring time included grouping of friends, grouping of pictures and setting of permissions. For Same-As Subject w/ Obj Gping, policy authoring time included grouping of pictures and setting of permissions for friends using the Same-As Example Friend as the policy

Table 12: Same-As object management user study results

Measure	**GB** μ	**SaS w/ Obj Gping** μ	**SaO** μ	$F - Statistic$ $p - value$
Number of Policy Templates				
Unconcerned	3.44	3.77	3.22	F(2,24)=0.34 $p = 0.71$
Pragmatist	3.44	4.69	4.47	F(2,186)=8.65 $p < 0.01$
Fundamentalist	3.11	3.55	3.92	F(2,78)=2.89 $p = 0.06$
All	3.35	4.30	4.21	F(2,294)=9.71 $p < 0.01$
Policy Authoring Time (seconds)				
Unconcerned	213.0	144.0	160.7	F(2,24)=2.25 $p = 0.12$
Pragmatist	223.3	157.8	177.8	F(2,186)=8.52 $p < 0.01$
Fundamentalist	227.7	148.2	152.7	F(2,78)=5.47 $p < 0.01$
All	223.6	153.9	169.4	F(2,294)=15.84 $p < 0.01$
Policy Openness (see Definition 4)				
Unconcerned	67.7	69.9	69.6	F(2,24)=0.01 $p = 0.98$
Pragmatist	67.8	56.8	53.2	F(2,186)=5.18 $p < 0.01$
Fundamentalist	60.2	55.4	51.5	F(2,78)=1.24 $p = 0.29$
All	65.7	57.6	54.2	F(2,294)=5.13 $p < 0.01$

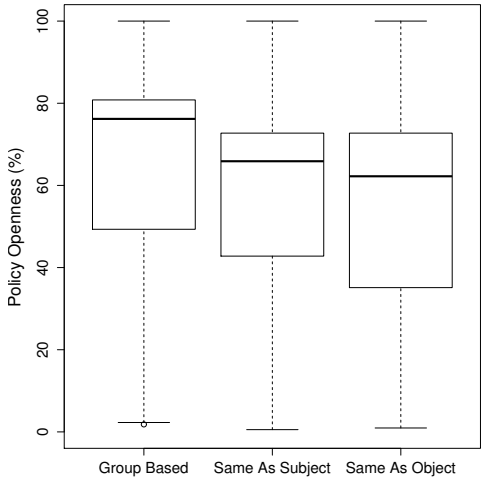
Table 13: Same-As object management pairwise comparison

Measurement	**GB vs. SaS** $p-value$	**GB vs. SaO** $p-value$	**SaS vs. SaO** $p-value$
Policy Templates	< 0.01	< 0.01	1
Authoring Time	< 0.01	< 0.01	0.70
Openness	0.08	< 0.01	1
Ease of Use	0.07	< 0.01	0.53
Readability	< 0.01	< 0.01	< 0.01
Flexibility	< 0.01	< 0.01	0.61



(a) Number of Policy Templates

(b) Policy Authoring Time



(c) Policy Openness

Figure 30: Same-As object management user study results

template. For Same-As Object, policy authoring time included grouping of friends and setting of permissions for pictures using the Same-As Example Picture as the policy template.

For Pragmatists, Fundamentalists and the population as a whole, we see statistical significance as it pertains to policy authoring time – all *p-values* are less than 0.01 and *F-Statistics* are greater than 3.04 (Pragmatist), 3.11 (Fundamentalist) and 3.02 (All) for a probability of 95%. Refer to the Policy Authoring Time section of Table 12 and Figure 30(b). We also ran a pairwise comparison leveraging the Bonferroni correction and observed no statistical significance between Same-As Subject w/ Obj Gping and Same-As Object. However, we do see statistical significance between Group Based and Same-As Subject w/ Obj Gping and Group Based and Same-As Object. See Table 13. Pragmatists, Fundamentalists and the population as a whole, took less time authoring their policies with the Same-As approaches over the Group Based approach. Overall, we see a 31% reduction in policy authoring time when using Same-As Subject w/ Obj Gping (153.9 seconds) versus Group Based (223.6 seconds). We also see a 24% reduction in policy authoring time when using Same-As Object (169.4 second) versus Group Based.

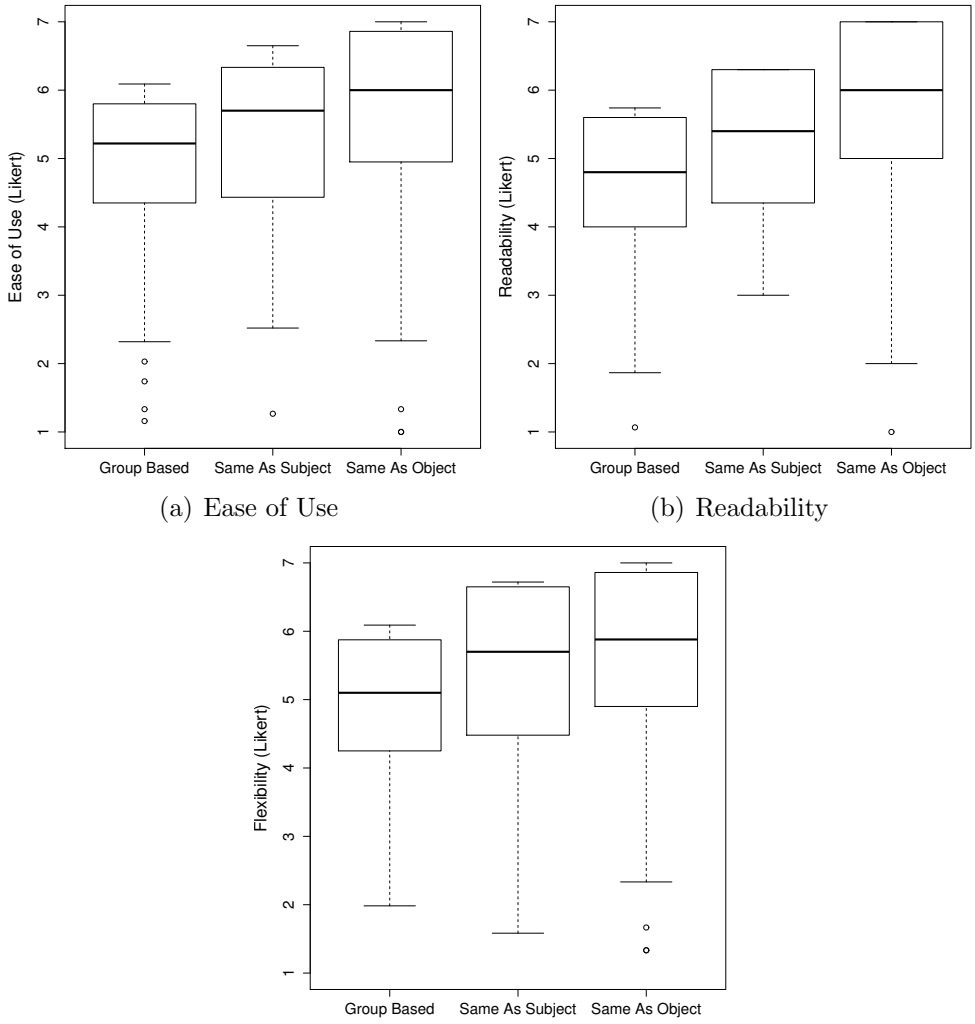
Policy Openness: We examined the *openness* of each user's policy or conversely, the conservativeness of a user's policy. This measurement gives an indicator of the restrictiveness (or not) of a user's policy, where a measurement of 100% indicates a totally permissive policy and a measurement of 0% indicates that the policy provides no access. See Definition 4 for the definition of Policy Openness.

For Fundamentalists, there is no statistical significance as it pertains to Policy

Openness ($p = 0.29$). See the Policy Openness section of Table 12 and Figure 30(c). There is statistical significance for Pragmatists and the population as a whole; both *p-values* are less than 0.01. In running a pairwise comparison, the only observed statistical significance is between Group Based and Same-As Object. See Table 13. Same-As Object policies were more conservative than Group Based policies, 54.2% versus 65.7%.

Ease of Use: The user completed a brief survey designed to capture their perceptions after each experiment. The question responses are on a Likert-scale of 1 (Strongly Disagree) to 7 (Strongly Agree). Each question is designed to capture the user's perceptions in the following areas: Ease of Use, Readability and Flexibility. For Ease of Use, the user needs to be able to manage their access control policies in an easy, intuitive and effective way such that they have a consistent experience. We see statistical significance across all the user segments (minus Unconcerned) for Ease of Use. See the Ease of Use section of Table 14 and Figure 31(a). However, in doing a pairwise comparison, we only found statistical significance between Group Based and Same-As Object. Refer back to Table 13. Overall, users found Same-As Object easier to use than Group Based – 5.59 versus 4.92 on a seven point Likert-scale.

Readability: Not only does a policy management solution have to be easy to use, it must be decipherable. For Readability, we see statistical significance across all the user segments (minus Unconcerned). See the Readability section of Table 14 and Figure 31(b). We also see significance across all experiment pairings. See Table 13. Overall, users found Same-As Object (5.71) more readable than Same-As Subject w/ Obj Gping (5.20) than Group Based (4.64).



(a) Ease of Use

(b) Readability

(c) Flexibility

Figure 31: Same-As object management user study results – perceptions

Table 14: Same-As object management user study results – perceptions

Measure	**GB** μ	**SaS w/ Obj Gping** μ	**SaO** μ	$F - Statistic$ $p - value$
Ease of Use (7 point Likert-scale)				
Unconcerned	5.02	5.38	5.37	F(2,24)=0.10 $p = 0.90$
Pragmatist	4.95	5.51	5.54	F(2,186)=4.13 $p = 0.01$
Fundamentalist	4.82	4.92	5.77	F(2,78)=6.15 $p < 0.01$
All	4.92	5.34	5.59	F(2,294)=6.59 $p < 0.01$
Readability (7 point Likert-scale)				
Unconcerned	4.62	5.36	5.59	F(2,24)=1.05 $p = 0.36$
Pragmatist	4.53	5.19	5.62	F(2,186)=14.54 $p < 0.01$
Fundamentalist	4.89	5.16	5.95	F(2,78)=7.21 $p < 0.01$
All	4.64	5.20	5.71	F(2,294)=21.45 $p < 0.01$
Flexibility (7 point Likert-scale)				
Unconcerned	5.22	5.34	5.92	F(2,24)=0.52 $p = 0.60$
Pragmatist	4.92	5.44	5.52	F(2,186)=4.43 $p = 0.01$
Fundamentalist	4.71	5.44	5.85	F(2,78)=7.64 $p < 0.01$
All	4.89	5.43	5.65	F(2,294)=10.17 $p < 0.01$

Flexibility: Policy management mechanisms must be flexible to accommodate the user's needs and intentions. For Flexibility, we see statistical significance across all the user segments (minus Unconcerned). See the Flexibility section of Table 14 and Figure 31(c). We also ran a pairwise comparison and observed no statistical significance between Same-As Subject w/ Obj Gping and Same-As Object. However, we do see

statistical significance between Group Based and Same-As Subject w/ Obj Gping and Group Based and Same-As Object. Refer back to Table 13. Overall, users found Same-As Subject w/ Obj Gping (5.43) and Same-As Object (5.65) more flexible than the Group Based (4.89).

4.5.2.3 Discussion

Same-As Subject Management with Object Grouping improves upon Subject / Object Grouping. We found that Group Based is more limiting in how a user may express their policies. On average, users of Group Based only leverage three policy templates versus four for Same-As Subject w/ Obj Gping. Fewer policy templates reflect that the user is authoring policies that aren't as expressive as they would like them to be. With Group Based, a user is being forced into expressing their policy in a way that may not align with their mental model – how the user views a policy versus how the access control model allows that policy to be expressed. We also see this reflected in Figure 30(a) where Group Based has a smaller distribution of policy templates versus Same-As Subject w/ Obj Gping.

Group Based provides fewer options for expressing one's policy. Same-As Subject w/ Obj Gping provides a means that aligns with how the user views their policies. Users are thinking about subjects (their friends) and the trust levels of those subjects when they are thinking about setting access permissions. With Same-As Subject w/ Obj Gping, users can create any number of different permutations for expressing their policy, all aligning with their intentions. This is reflected in our Flexibility measurement. On average, users gave higher ratings for flexibility to Same-As Subject

w/ Obj Gping over Group Based. In access control terms, Same-As Subject w/ Obj Gping has more expressive power for representing policies than does Group Based.

The expressiveness of Same-As Subject w/ Obj Gping is also very readable compared to Group Based. Using our visual policy editor, users are able to see the summarized expressiveness of their policy in a format this is easy to understand. With Group Based, a user must change between the different grouping and policy views to get a comprehensive understanding of their policy. Our Same-As Subject w/ Obj Gping visual policy editor presents the policy in a single view providing a global perspective to the user which is decipherable and aligns with their mental model. This allows the user to construct policies that align with their intent.

Not only are users authoring more expressive policies, they are taking less time to do so. We found that users are taking approximately 31% less time in authoring a policy using Same-As Subject w/ Obj Gping versus Group Based. One factor attributing to this performance improvement is that with Group Based, a user must complete three disjoint tasks, i.e., group subjects, group objects and set permissions. With Same-As Subject w/ Obj Gping, a user first groups their objects. Then within one task, a user authors their policy by setting permissions for their object groups using the Same-As Example Subject as the policy template. With Same-As Subject w/ Obj Gping, the user is conducting fewer mental task switches. Conversely with Group Based, a user must focus on their relationship with their subjects and how they should be organized / grouped. Next, the user must think about their objects and how they are similar from a sensitivity perspective. Finally, the user must think about access permissions when they are authoring their policies. With Same-As Subject w/ Obj Gping, after

grouping their objects, the user relies on their memory and opinion of their subjects to set policies for other similar subjects. As a result, users can author policies much faster.

As with Same-As Subject Management with Object Grouping, Same-As Object Management also out performs Subject / Object Grouping for many of same reasons discussed above. Same-As Object is similar to Same-As Subject w/ Obj Gping in that they both allow the user to author expressive policies which align with the user's mental model. With Same-As Object, the user generates a similar number of policy templates (with similar distribution) as Same-As Subject w/ Obj Gping. Same-As Object is also equally as flexible as Same-As Subject w/ Obj Gping in providing a similar level of expressive power for representing policies over Group Based.

With regards to readability, users found Same-As Object not only more readable than Group Based but also more readable than Same-As Subject w/ Obj Gping. We attribute this to the introduction of our new paradigm in how objects are managed with Same-As Object. In both Group Based and Same-As Subject w/ Obj Gping, objects are grouped in the traditional fashion, i.e., by some common property – in our study, sensitivity level. Using Same-As Object, the user associates objects with other like objects that possess a common user assigned subjective meaning of sensitivity. The user assigns the importance of the object by associating it with an easy to remember representative object, thus creating a level of abstraction based on the user's intent. In doing so, the policy is more readable and understandable to the user. As a result, users are authoring more conservative policies. We see Same-As Object policies to be more conservative (less open or permissive) than those policies

authored using Group Based. Users are thinking about their objects in terms in which they assign, which aligns with their mental model and intentions. Therefore, they are more likely to create more expressive policies which are more *least-privilege* like, i.e., conservative.

All these factors attribute to the positive Ease of Use ratings received by the study participants. Users found Same-As Object to be easier to use than Group Based. Users found the new paradigm and visual policy editor simple to use and easy to convey their access control intentions. As a result, users were able to author policies in less time with Same-As Object over Group Based, where we measured a 24% reduction in policy authoring time.

CHAPTER 5: APPLICATION POLICY MANAGEMENT

5.1 Social Networks Connect Services

Online social networking sites allow users to build social connections with family, friends, co-workers, etc., hereafter referred to as *friends* or a *social graph*. These online social networking sites also allow their users to build profiles for storing and sharing various types of content with their friends, including photos, videos, messages, etc. Updating user profiles with interesting content is a form of self-expression that increases the interaction between friends in social networking sites. In order to encourage this interaction and to provide more rich content, social networking sites expose their networks to web services in the form of online API's. These API's allow third party developers to interface with the social networking site, access information and media posted on user profiles, and build social applications that aggregate, process, and create content based on users' and friends' interests.

Social networking sites are able to provide various application services that mash up user profile data with third party data. Third party sites can rapidly distribute their services via social networking sites in order to keep in touch with their users while they are on these social networking sites. Moreover, users can now enjoy various applications with content from numerous third party sites, see Figure 32. Clients (social networking site users) access social networking sites where they maintain their profiles. These profiles are retrieved by third party sites. The user provided content

is enriched by the third party site and returned to the social network for consumption by the user and, possibly, other friends. For example, Facebook users can share music with friends, create play lists and get concert alerts on their profile page by installing the third party music application iLike (`www.iLike.com`). This was the first step in the evolution of social networks from a walled garden to a more open environment aggregating content from a variety of sources.

Figure 32: Social network application

Major social networking sites have since launched new services such as Facebook Platform, Google Friend Connect, and MySpaceID. We call these new services *Social Networks Connect Services*. Social Networks Connect Services allow third party sites to develop social applications and extend their services without having to either host or build their own social network. This extension allows third party sites to leverage the features of the social networking site. For example, third party sites can leverage the authentication services provided by a social networking site. Users, therefore, are not required to create yet another username and password to access the third party site. They can just leverage their social network credentials and established profile to include friends list. In another example, a client (social networking site user) can access a third party site which leverages social network user profile content, see Figure 33. The third party site retrieves users' profiles from the social networking site

returning to the clients an enhanced experience and enriched content. Third party sites are able to increase membership by providing more interesting content from a variety of sources in a seamless manner. Social Networks Connect Services further break down the garden walls of social networking sites.

Figure 33: Social networks connect services

5.1.1 Framework

In order to better understand the Social Networks Connect Service, we begin by describing user data in the social web. User data is composed of three types of information: identity data, social graph data, and content data. Identity data describes who I am in social the web including my identifier, my profile information, and my privacy policy. Social graph data represents who I know in the social web including my friendship connections with descriptions such as family, coworker, colleague, etc. Content data represents what I have in the social web including my messages, photos, videos and all other data objects created through various social web activities.

In order for social networking sites to be able to share user social web data with other third party sites, a secure and reliable Social Networks Connect Services framework is required. This framework is composed of a collection of API's that allow third party sites to interface with the social networking site. These API's are grouped into four categories:

- *Identity Authentication:* Used to prove the identity of the user. Users can authenticate using their existing accounts from various identity providers, to include the social networking site.

- *Authorization:* Used to govern access to user data in the social web based on pre-defined authorization access rights. The Authorization API allows third party sites to create new content and extract existing content from the user's social web data.

- *Streams:* Allows third party sites to publish to users' activity streams and vice versa.

- *Application:* Allows third party sites to develop rich social features in the form of applications and thus creating an extension of the social web.

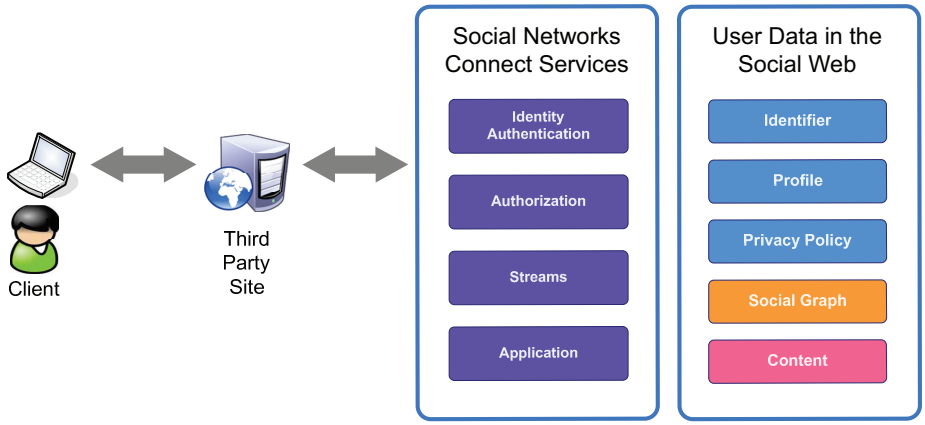


Figure 34: Social networks connect services framework

These four API's make up the Social Networks Connect Services, see Figure 34. The implementation of the services can vary widely using different protocols and

technologies. In the next sections, we explore the Social Networks Connect Services in Facebook, Google, and MySpace called Facebook Platform, Google Friend Connect, and MySpaceID respectively.

5.1.2 Facebook Platform

Facebook Platform became generally available in December of 2008 under the Facebook Connect brand. Since then, thousands of third party sites have implemented it. Facebook Platform allows third party sites to integrate with Facebook and send information both ways to create more engaging and rich social experiences on the web. Facebook Platform allows users to bring their identity, profile, privacy policy, social graph, and content from Facebook to third party sites, see Figure 35.

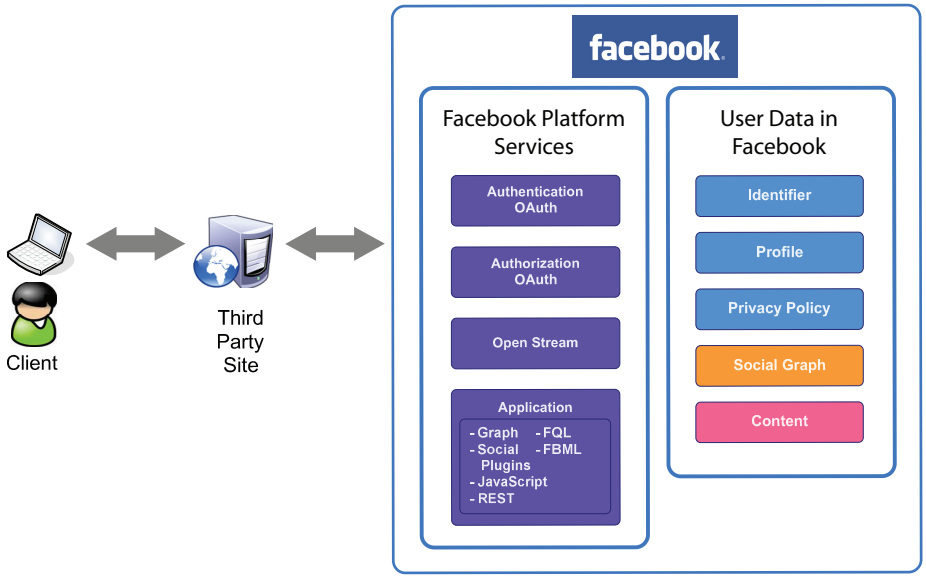


Figure 35: Facebook platform services

Authentication is, by far, the most used component of Facebook Platform. This API allows third party sites to leverage Facebook as an identity provider. For example, Digg.com, a social news web site where users can share content, doesn't require

its new members to register and create a profile, see Figure 36, Step 1. New members of Digg.com can just leverage their Facebook credentials to authenticate and use their existing Facebook profile – Step 2. Once authenticated, users can extend their social graph to the third party site and invite their friends (or link to them, if they are already members) to join them on Digg.com – Step 3.

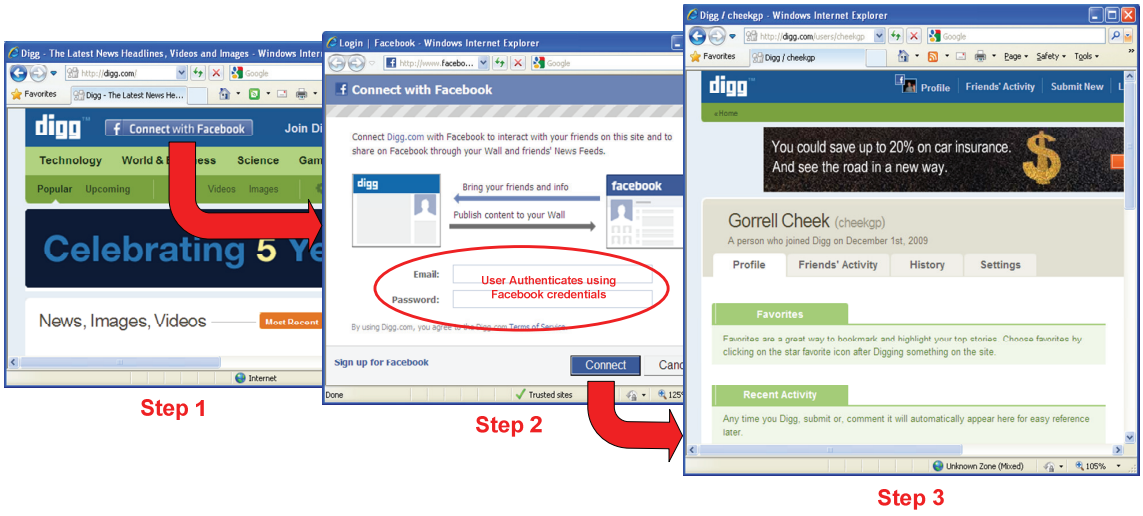


Figure 36: Digg.com authentication via Facebook platform

Facebook Platform leverages OAuth 2.0 for Authentication and Authorization [21]. OAuth 2.0 is a simplified and improved version of the standard that allows third party sites to obtain authorization tokens from Facebook. First, a user of the third party site authenticates using Facebook as an identity provider. Next, Facebook issues a token that can be used by the third party site to access the user's basic profile information including name, profile picture, gender, and friend's list. Extended permissions can be requested by the third party site, depending on the specific application requirements, e.g., permission to access the user's activity stream like their Wall. (OAuth is further discussed in Section 5.1.3.) Facebook user's privacy policy settings are also applied

through the third party site using Facebook's dynamic privacy controls. The third party site is able to provide the same privacy controls as on Facebook. For example, if Alice does not allow Bob to access her content on Facebook, then Bob would not be allowed to access Alice's content on the third party site.

The Open Stream API allows third party sites to read and write to users' activity streams, e.g., read and write short messages on new events and activities for others in the social web to consume. This API supports multiple stream publishing methods, in addition to the Atom Feed standard. Third party sites can read content from a user's activity stream in addition to publishing to their activity streams. For example, Facebook user activity on third party sites can be shared with their friends on Facebook through their News Feeds.

Facebook allows developers to create social applications that interface with third party sites. They provide a series of API's and tools to assist developers in creating these third party social applications (`developers.facebook.com/docs`). The primary API is Facebook's Graph API. It allows third party applications to read and write content objects (e.g., photos, friends, etc.), and the connections between them, in Facebook's social graph. The API is simple in the sense that it allows access to content objects via URLs. For example, with proper authorization, `graph.facebook.com/FBUserID/friends` will provide access to *FBUserID's* friends. Facebook also provides support for the Representational State Transfer (REST) API. However, future enhancements will mainly focus on the Graph API.

Facebook's Social Plugins also enable traditional third party sites to easily transform into the social web with only minimal HTML. The *Like* plugin allows third

party site users to share pages / content back to their Facebook profile. The *Recommendations* plugin allows third party site users to suggest or recommend content on the site. The *Activity Feed* plugin shows Facebook users what their friends *Like*. Facebook also offers a variety of other Social Plugins including: *Comments*, *Facepile*, *Login with Faces*, etc.

Facebook has a JavaScript Software Development Kit (SDK) consisting of classes and methods that can be used to integrate third party sites using Facebook Platform. Facebook also provides a series of tools that developers can use for building social applications. Facebook Markup Language (FBML) is a proprietary variant of HTML and Facebook Query Language (FQL) provides a quick and easy mechanism to query Facebook user data without using API methods.

5.1.3 Google Friend Connect

Google Friend Connect was released approximately the same time as Facebook Platform. And, like Facebook Platform, Google Friend Connect makes it easy to share profile, social graph, and content data with third party sites (`www.google.com/friendconnect`). Google Friend Connect takes a standards based decentralized approach to facilitating the integration of social and non-social web sites. It uses open standards like OpenID, OAuth, and OpenSocial (`openid.net`, `oauth.net`, `www.opensocial.org`), see Figure 37.

Authentication services are provided using OpenID. OpenID takes a decentralized approach for user authentication by using existing identity providers like Google, Yahoo, and OpenID. Users can authenticate to third party sites using credentials

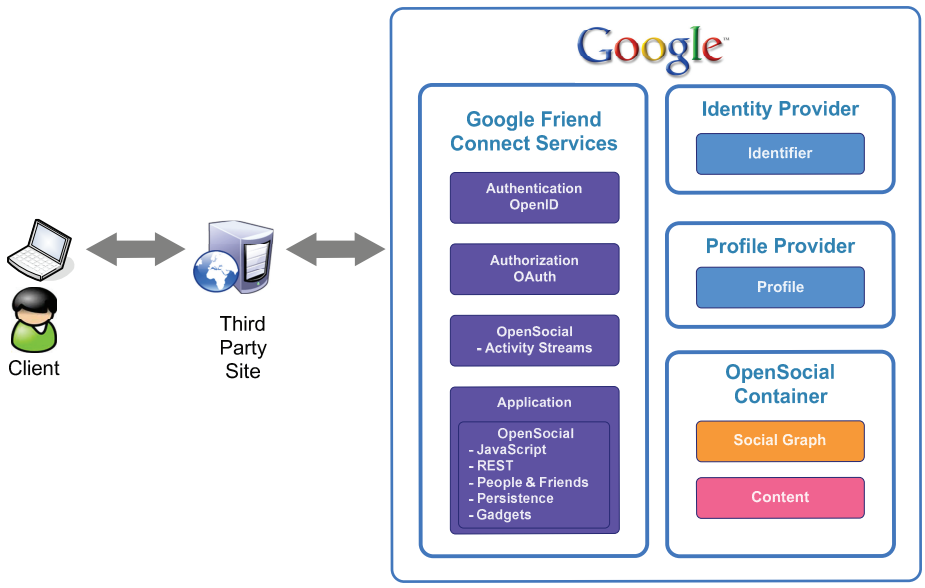


Figure 37: Google friend connect services

that are issued by a supported OpenID identity provider. The user, therefore, doesn't have to go through the third party site new member registration process. After initial authentication to the the third party site, the user then can select their existing profile from a profile provider site like Plaxo. After which, they can import their social graph from a social networking site such as Orkut and share their activities in the third party site with a selected social graph. For example, if a user posts a message on a board in a third party site, the message is only visible to friends in the selected social graph.

Google Friend Connect provides authorization services via OAuth, which is an open standard that provides secure API authorization. OAuth provides granular authorization control of user content in the social web based on privacy policies. OAuth allows users to share their content hosted on a third party site without having to provide a username and password to the requesting site. Authorization tokens are issued to a specific site for a specific content object for a specified time. For example,

assume Site A houses photos on behalf of a user and Site B requires access to these photos. Site B requests access from the user who then authorizes the access request by authenticating to Site A and authorizing Site B's access request. Site A will then issue an authorization token for the photos to site B. Site B never gains access to the user's Site A credentials.

Google Friend Connect uses OpenSocial to provide support for activity streams. This API is used to read and write to a user's activity stream on Google Friend Connect sites. This enables users to see what their friends are up to and to share what they are doing. For example, a blog site leveraging Google Friend Connect can publish activities to a variety of OpenSocial sites. See Figure 38 for a screen shot of a blog site user profile page that leverages Google Friend Connect – notice options to *Connect with friends from social networks...* and an option to *Publish my activities...*

The OpenSocial API's are used by Google Friend Connect to build social applications. OpenSocial was developed primarily by Google with the support of a number of other social networking sites including MySpace, Hi5, etc. The primary objective of OpenSocial is to create a platform that allows social applications to integrate with a number of different social networking sites. Thus, developers of third party sites can create a social application that can integrate with a variety of OpenSocial containers like MySpace, Hi5, LinkedIn, Orkut, etc. The advantage of OpenSocial is that social applications can potentially reach more users if they integrate with multiple container sites. And, theoretically, users will have access to more social applications.

OpenSocial's client side API leverages standard web development technologies like JavaScript, HTML, AJAX, etc. Server side development is accomplished via the

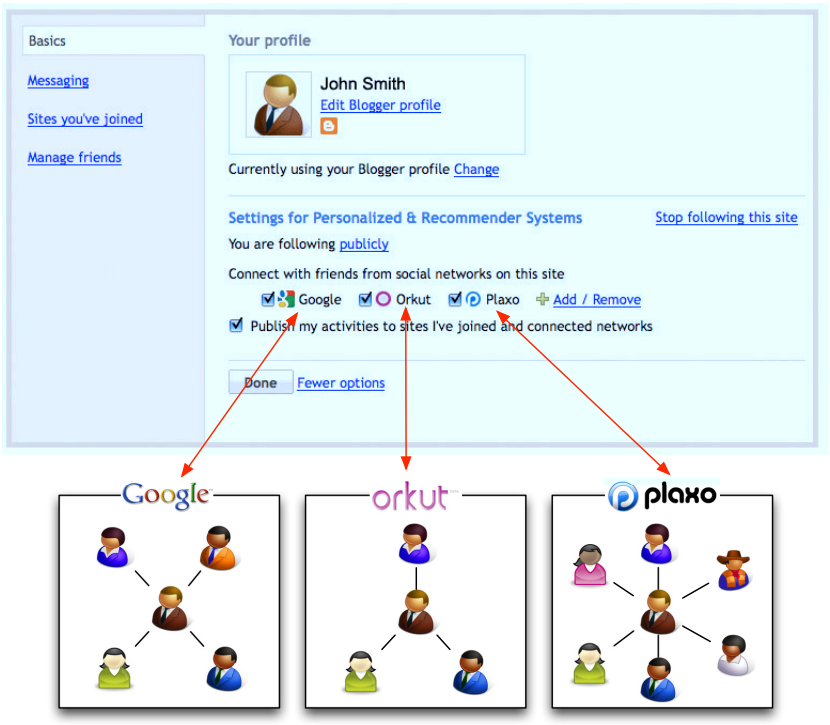


Figure 38: Google friend connect blog site profile page

RESTful data API's. These API's expose three primary sets of data: 1) People and Friends – who I am and who I know (Social Graph), 2) My Activities – discussed above, and 3) Persistence – providing the ability to read and write data with your friends (Content).

A simpler mechanism used by many sites that adopt Google Friend Connect is to insert gadgets into their third party sites' pages. A gadget is a client side HTML / JavaScript applet. A third party site can quickly add social features by just integrating a few snippets of code into their site. Different social gadgets, such as a rating gadget – giving users the ability to rate a movie, easily transform the third party site from the web to the social web. The limitation of this approach is that gadgets cannot access social graphs or content. The JavaScript or RESTful data API's must

be used to overcome this limitation.

5.1.4 MySpaceID

MySpace has aligned with Google's open standards approach to social web development relying on standards such as OpenID, OAuth, and OpenSocial to share profile, social graph, and content data with third party sites (`developer.myspace.com/myspaceid`), see Figure 39. MySpaceID allows users to login to third party sites using their MySpace credentials. In addition, it enables MySpace users to share their profiles and social graphs with third party sites. MySpaceID also provides support for activity streams. OpenSocial and MySpace's own RESTful API's are the primary resources for creating third party social applications. Since, MySpace is an OpenSocial container, MySpace user data is easily accessible to any third party site using the OpenSocial standard.

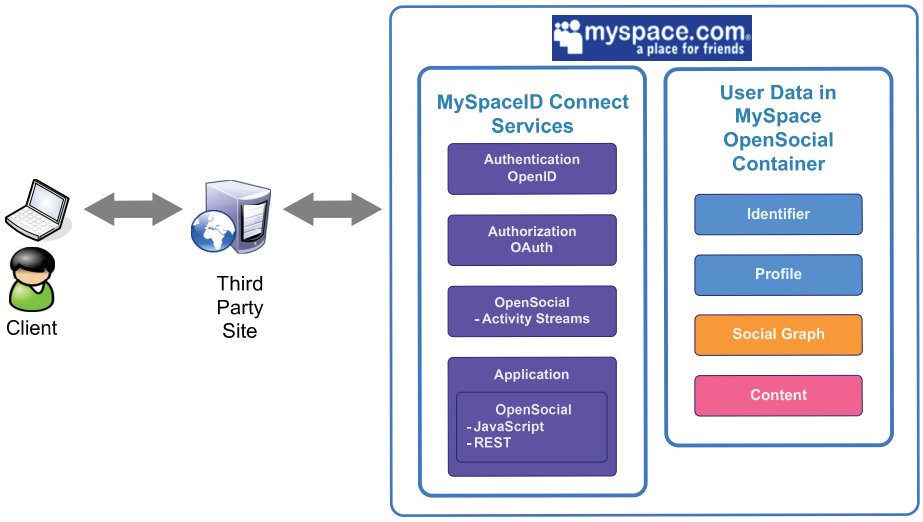


Figure 39: MySpaceID connect services

5.1.5 Comparison

Facebook Platform, Google Friend Connect, and MySpaceID enable third party sites to integrate with the social web without having to build their own social network. Using these connection services, social networks users are able to leverage their existing identity, profile, social graph and content on these third party sites. Moreover, these connection services provide social web integration technologies to third parties in the form of suites of API's and tools.

One difference between Facebook / MySpace's approach and Google Friend Connect, is that Google Friend Connect uses decentralized identity, profile, and social graph providers. Facebook and MySpace leverage their social network platforms for user data in the social web. By using decentralized identity, profile, and social graph providers, users can leverage specialty providers selecting the best of all possible worlds. Users can customize their presence on the social web. The down side to a decentralized approach is that users must maintain their social data in multiple locations increasing administration costs. Using a single provider, like Facebook or MySpace, cuts down on this administrative overhead. But, a single provider limits the user's capabilities and choices to just that of the one provider.

Google and MySpace take an open standards approach while Facebook has traditionally taken a proprietary approach; but, most recently, Facebook has adopted the use of OAuth. Social applications that take an open standards approach, theoretically, can reach more users because there are many standards based service providers giving access to more users. However, these users are spread across a variety of service

providers. Having millions of users within one service provider, as Facebook does, provides significant mass in the overall social web. Third party sites want access to Facebook's membership because developing one integration gives access to a extremely large user base. See Table 15 for an overview of the different Social Networks Connect Services.

Table 15: Social networks connect services comparison

	Facebook Platform	**Google Friend Connect**	**MySpaceID**
Auth. & Auth.	- single service provider w/ 100's of millions users - open standard: OAuth	- multiple service providers w/ 100's of millions of users - open standards: OpenID & OAuth	- single service provider w/ millions of users - open standards: OpenID & OAuth
Streams	- full support - proprietary	- full support - open standard: OpenSocial	- full support - open standard: OpenSocial
App.	- full suite of tools & API's - proprietary	- full suite of tools & API's - open standard: OpenSocial	- full suite of tools & API's - open standard: OpenSocial

5.1.6 Open research security challenges

Because of Social Networks Connect Services, the social web is growing exponentially. With this growth, comes challenges to include security and privacy challenges. In this section, we introduce some of these challenges. We briefly describe the challenges, discuss them, high-light select related work, and offer ideas that may lead to solutions. Our goal is to introduce open research security challenges that have arisen from the introduction of Social Networks Connect Services.

5.1.6.1 Identity Mapping

Effective user identity mapping between social networking and third party sites is one of challenges in social web. Users have multiple identities scattered across various social networking and third party sites. For new users of third party sites, Social Networks Connect Services eliminates the need for a cumbersome registration process. Social Networks Connect Services also supports the scenario where a user has an existing account on a third party site and would like to link it to an existing account on a social networking site, thus providing a seamless social web experience. For example, Facebook provides a method that creates an association between an existing user account on a third party site and the user's Facebook account. To link these two accounts (assuming the third party site and Facebook have the user's email addresses), the two email hash values are compared. If the hash values match, the two accounts are linked and are considered the same owner's accounts. This approach is also used to find user's friends between social networking and third party sites. If a user uses the same email address between a social networking and a third party site, this email hash based mapping approach is sufficient to accurately create a link. However, if a user uses a different email address on the social networking site than the third party site, the email hash based mapping approach will not work. Moreover, if the user's friends use different email addresses, linking friends' accounts is almost impossible.

The mapping of friends' accounts is especially important in protecting user's privacy on the social web. Currently, existing identity mapping methods are not perfect

and have many shortcomings. For example, as mentioned above, email hash based mapping is only applicable when users' friends are using the same email address on the social networking sites as on third party sites. One ideal solution is that all users use a global unique identity offered by an OpenID service provider. However, implementation and acceptance would be quite difficult.

Researchers have investigated attribute based identity mapping. But, Wang et al. [70] revealed that incomplete records with many missing data elements would significantly increase the error rate of the record comparison algorithm. This is a common limitation of many identity matching techniques that only use personal attributes. Xu et al. [74] showed that combining social features with personal features could improve the performance of criminal identity matching. However, this approach may not be reliable in today's social web since the quality of user attributes in profiles is low due to deception, errors, or missing attributes.

Mapping multiple identities issued by different identity providers is another issue in Social Networks Connect Service. Google Friend Connect provides authentication services using multiple identity providers. If a user logins into a third party site using one identity provider and then later logins into the same third party site using a different identity provider (assuming the user has identities with both providers), the third party site will treat this user as two separate identities and the user will have two profiles, two activity streams, two sets of friends, etc. Also, given the fact that most users easily forget their usernames and passwords, adding an additional dimension (select your identity provider) will further burden the user.

Identity mapping techniques that overcome the limitations of existing approaches

are necessary to effectively manage user's and user's friends identities in the social web. An effective friend mapping technique will enhance the user's privacy protection since it will provide correct friends' accounts linking and therefore allow users to effectively control their friends' accesses in the social web.

5.1.6.2 User Data Portability

Most social web users have multiple profiles and social graphs on different sites and most of them are not interlinked. Users may need to maintain multiple profiles and social graphs based on the context of the social networking site. For example, a user may have an account on Facebook to keep in touch with family and friends – with one profile and set of friends. The same user may also have an account on LinkedIn for professional reasons – with a totally different profile and set of colleagues. Maintaining multiple profiles and social graphs across different social networking sites can be a burdensome chore. One possible solution could allow for users to maintain a global profile and social graph in one place. The user could then create sub-profiles tailored around a specific community (e.g., family / friends, professional colleagues, etc.) setting appropriate privileges on these sub-profiles depending on the intended audience. There are challenges with this approach: identification of single profile repository, duplicate profile attribute resolution, multiple profile attribute name space mapping, and mapping of friends / colleagues across different social graphs. Sutterer et al. [64] propose a user profile ontology that describes situation dependent sub-profiles. This ontology is designed to be leveraged in context aware environments. It structures a user profile into situation dependent sub-profiles and delivers the appropriate

sub-profile based on the specific environment or context. Their ontology is primarily geared for the personalization of services on mobile communications platforms. But, a similar approach may be applicable to the social web. For example, a user could maintain a global profile with multiple sub-profiles that are made available dependent on the context of the site being visited.

5.1.6.3 Common Enhanced Privacy Policy Framework

Social Networks Connect Services have rapidly expanded the social web into a very large online community. Opening the user's social web data to third party sites with an immature privacy policy framework creates significant privacy issues and concerns. Social networking and third party sites have their own access control mechanisms with privacy policies that are composed using different and incompatible policy languages. Therefore, it is difficult to apply a consistent privacy policy between social networking sites and third party sites.

Current implementations of Social Networks Connect Services have very primitive privacy policy capabilities, i.e., limited ability to set and enforce a policy on user data in the social web. Facebook Platform extends users' privacy policies to social applications hosted on third party sites, but only if the social application developer fully implements these capabilities. Google Friend Connect and MySpaceID have an even more immature and basic approach to managing the privacy of user profiles and content. Maintaining user privacy policies using current implementations of Social Networks Connect Service is not trivial.

Privacy policies are expressed as access control policies which can be composed

in policy languages using policy management tools. Several researchers have investigated access control systems for sharing content in the social web. Lockr [67] is an access control scheme based on social relationships which regulates the sharing of personal content in social web. It separates content delivery and sharing from managing social networking information and lets users manage a single social graph with access control lists (ACLs). A person who has a social relationship described in an ACL can access the shared content under the Lockr system. Mannan et al. [44] propose an access control model based on an existing "circle of trust " in Instant Messaging (IM) networks. Sun et al. [63] propose a system for content sharing in the social web where users leverage their email accounts as an OpenID identity and content owners use their email based contact lists to specify access policies in Open-Policy providers. These proposed models have their limitations. For example, they enforce access control based on a single social graph from an email contact list, IM, or a social networking site and are not applicable across multiple social graphs.

Designing a generic privacy policy framework for social networking and third party sites is a challenge. OAuth has propelled authentication and authorization in the social web forward. We've see how the community developed an open authorization protocol that is being widely deployed in the social web. An open privacy policy framework would also need to be designed and developed by the community in a similar fashion. However, without collaboration and cooperation of the community, a consistent privacy policy framework in social web would be difficult to attain. However, even with open collaboration, there are challenges in building an open privacy policy framework: detailed access control methods which include fine-grained con-

trols, compatible policy languages, and a consistent user experience.

5.1.6.4 Cascaded Authorization

Many internet sites provide mashup services, i.e., a service that combines data or functionality from two or more third party sites to provide a new enhanced aggregated service. Third party sites that implement Social Networks Connect Services are required to obtain the user's consent prior to being authorized to retrieve the user's social web data. Third party sites frequently rely on other third party sites to provide services, which require the sharing of user data from one third party site to another. Using the current authentication and authorization approaches would require the user to consent multiple times – one for each third party site accessing their data. This approach is very cumbersome and time consuming. To enhance the user's experience and privacy in sharing data with mashup based services, a cascaded authorization mechanism is necessary that would obtain a user's consent once for a specific content object and that authorization would cascade to all third party sites that use that content object as part of the mashup service.

The current state of the art is the 4-legged authentication of OAuth 2.0 [21] which enables third party sites to generate and share authentication tokens to delegate access rights to other third party sites. The challenges raised by this approach are that it requires the user to trust third party sites into making authentication decisions on behalf of the them. *OAuth for Recursive Delegation* [69] is a proposed extension of OAuth that allows a client (or third party site) to delegate the authorization it received from the user to another client. However, this is an emerging standard with

no concrete implementations.

To ensure the security and privacy of the user, Social Networks Connect Services need to provide easy to use fine-grained access control techniques that enable the user to decide which attributes are to be shared with what third party sites. Furthermore, the user will need to be able to track and control all third party site accesses and delegated accesses.

5.1.6.5 Privacy in Social Plugins

Social Networks Connect Services enable users to share not only site URLs, events, and photos with other users, they also allow users to share their opinions. For example, a user watching a Youtube video is able to share that he likes this video with his friends on the social web. Furthermore, the user is able to share comments posted on Youtube with friends too. Facebook's Social Plugins (discussed in Section 5.1.2) are widely deployed. Several third party sites are adopting the thumbs-up *Like* button to enable users to easily post their opinions on these sites. The "liked" web object is referenced and shared using the object URL, which makes it easy for a user's friends to check the shared object. Even though, this enables the seamless sharing of user opinions, it does not provide for data integrity guarantees. For example, what happens if the content of the "liked" object changes after a user indicates that they liked such content? The lack of integrity guarantees when using Social Plugins is a user privacy concern. Adequate controls for Social Plugins guaranteeing integrity is an open research security challenge.

5.2 iLayer: Application Access Control Framework

5.2.1 Preliminaries

5.2.1.1 Content Management Systems

A Content Management System (CMS) is an online application that provides users the ability to easily create, design, publish and manage the content of a web site or social networking site. In addition, content management systems manage work flow allowing for collaboration. Users of content management systems are not required to have an in depth technical knowledge of web design or programming languages. Users traditionally just leverage a browser based interface to easily build and manage the content of a web site or social networking site. Content management systems support multiple users with varying roles, e.g., content contributors, content consumers, site administrators, etc.

There are both commercial and open source varieties of content management systems on the market today including Joomla, WordPress, MediaWiki, Plone and Drupal. They are deployed in various forms supporting a variety of different market segments including online publishing, online social networking, enterprise content management, and document management. Content management systems support many companies, government entities and academic institutions. For instance, Whitehouse.gov, Popular Science Magazine and the New York Observer web sites are all powered by Drupal [13]. Joomla supports several notable web sites including Harvard University and the United Nations Regional Information Center [28].

Third party applications expand the capabilities and functionalities of content man-

agement systems. There are thousands of applications available for most CMS platforms providing functionality like news publishing, location mapping, photo galleries, etc. Few formal application development security practices are in place, e.g., code reviews of third party applications, etc. There are some secure coding guidelines available [60, 3, 52, 27]. But, there is little enforcement of their use.

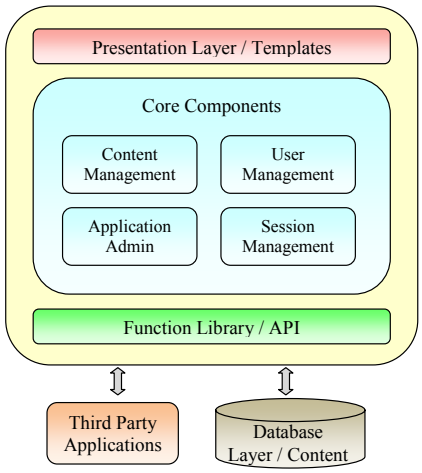


Figure 40: Content management system overview

Typically, a CMS consists of four components: Presentation Layer, set of Core Components, Function Library and Database Layer. See Figure 40. What follows is a brief description of each:

1. Presentation Layer: Displays to the visitor of the web site / social networking site the output (or content) of the CMS. Typically, templates populated with content are used to facilitate the creation of the web site / social networking site.
2. Core Components: Provides foundational CMS functionality including basic content management, user management, application administration, session management, etc.
3. Function Library / API: Library of functions that perform various tasks (e.g., database calls, etc.). In addition, an Application Programming Interface (API) is provided for interfacing with third party applications.

4. Database Layer: Stores all content such as users' profiles, blogs, files, etc. It also contains configuration and policy management information.

One of the primary benefits of most content management systems is that they can easily be expanded to include additional functionality. Content management systems publish their API's which allow third party developers to create applications that interface with the CMS. These third party applications provide additional capabilities. For example, a third party developed calendar application can provide user scheduling management. Or, a third party developed mapping application can provide location information for a restaurant web site. More often than not, these applications are free to the community to download, install and use.

5.2.1.2 Application Access Control Approaches

Content management systems provide user access control and management capabilities. However, access control functionality for third party applications is not as well developed. Third party applications can, and in some cases do, have full administrator level access to the web site's content and data, e.g., configuration information, user information, passwords, etc. CMS platforms, such as WordPress, Plone and Joomla, allow web site administrators to easily install third party applications. But, most CMS platforms advise the administrative user to secure their database and server configurations and change all the default passwords after installing new versions of software [73, 54, 28]. CMS platforms have traditionally addressed application access control as an after thought.

Third party applications developed for WordPress have full access to the content management system and the database, unless restricted through file permissions.

There is no mechanism where an administrative user can set permissions during the installation of a third party application. File permissions are not handled through a designated WordPress screen. Instead, the administrative user must set the file permissions via the command line of an operating system shell. The administrative user needs to find the right balance between restrictive and lax permissions [73]. Application and WordPress software functionality is affected if permissions are too strict. Permissions that are too lax present a security risk. The skills to set these permission levels is not universal among CMS administrators. Managing third party application access in WordPress is not trivial and can be easily misconfigured.

Joomla is another CMS platform that allows third party developed applications to interface with its content management system. The administrative user must also use file permissions to control the access given to these applications [28]. Other than file permissions prohibiting access, third party application access is not explicitly controlled. There is not a permissions granting process during installation either.

Thus, even though there are methods that allow web site administrators to control the access of third party developed applications, they are limited in capability and are not very easy to use. The administrative user would need to understand and translate the access requirements of the third party application into file permissions (e.g., in the case of Joomla) and, possibly, database and server configurations. This is not an easy undertaking and, more often than not, administrators take minimal or no action to secure their web sites from third party developed applications. The average web site administrator may not have sufficient skills or experience to know all of the risks associated with third party applications and how they may impact their

web site.

5.2.2 Framework

We investigated the access control mechanisms that govern access to online information from third party applications developed for content management systems. In order to improve the security of content management systems, we introduce iLayer – a *Policy Management Assistance Tool*. iLayer is a least privilege based model that protects content management systems from third party developed applications. iLayer makes policy recommendations to CMS administrative user for third party applications. These policies are reviewed and set by the CMS administrative user and enforced by the iLayer Framework.

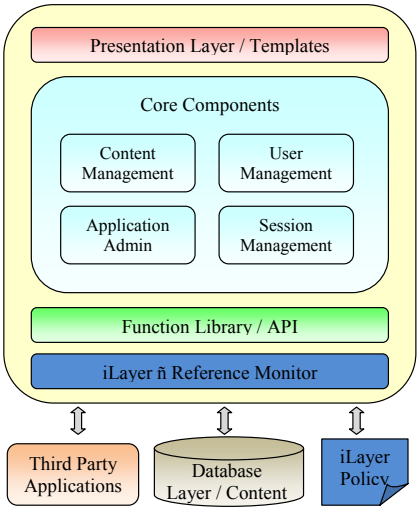


Figure 41: iLayer architecture

iLayer's primary goal is to set and enforce least privilege access for third party developed applications that integrate with content management systems. A subsequent design goal of iLayer is ease of use with minimal administration while enhancing the

security of the content management system. iLayer's Architecture has two primary components: Reference Monitor and a corresponding Policy. See Figure 41. The reference monitor controls all access from third party applications to the database. Basically, the reference monitor verifies that third party application database requests are allowed by the policy. There are three primary steps in establishing the architecture: iLayer Setup, Third Party Application Installation and Runtime Enforcement. See Figure 42.

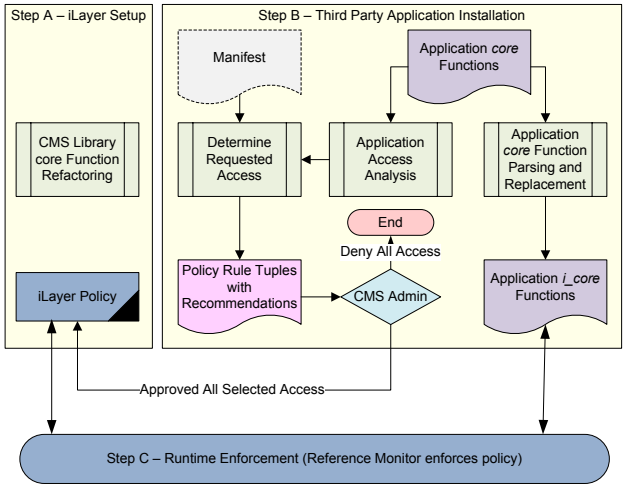


Figure 42: Establishing the iLayer architecture

5.2.2.1 iLayer Setup

The first step in installing the iLayer Architecture within a CMS platform is creating a table in the database to store and manage the iLayer Policy. A policy is made up of three components:

- *subject:* third party application that will be granted access
- *object:* database table being given access to

- *permission:* access privilege that is granted which could be either:
  1. *read* (select)
  2. *write* (delete, insert, update)
  3. *read & write*

After the policy table is set up, all core CMS platform functions that perform database calls are identified in the CMS Function Library, e.g., *core(arg)*, where *arg* contains the database table name and the requested action, i.e., select, delete, insert, or update. After which, the core functions are refactored [19], i.e., a corresponding *i_core(3PA_Params, arg)* function is added to the CMS Function Library for each core function that makes a database call, e.g., *core(arg)*. The *i_core(3PA_Params, arg)* function will have the same arguments as its corresponding core function with the addition of the calling third party application parameters, where:

```
3PA_Params = array(
      "name" => "Application Name is...",
      "id" => "Application ID is...",
      ...
)
```

The *i_core* function performs a policy check. If the policy is violated, access to the requested database table is denied and an error message is returned. If the policy is not violated, the *i_core* function calls its corresponding *core* function and operation proceeds normally. Figure 43 shows a code sample of a *core* function and its corresponding *i_core* function.

5.2.2.2 Third Party Application Installation

After the iLayer is setup within a CMS platform, third party applications can be installed. The first step in the installation process is the determination of the requested

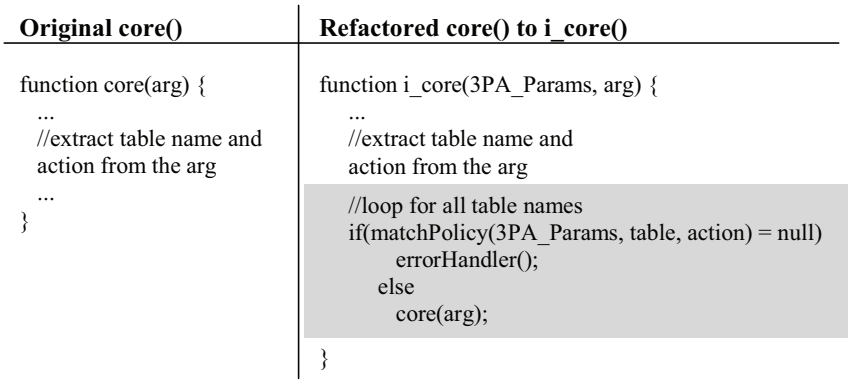


Figure 43: Refactoring core function

access by the third party application. Policy recommendations are then presented to the CMS administrative user for review and the policy is selected and approved by the CMS administrative user. After which, the third party application code is parsed and all instances of *core* functions are replaced with their corresponding *i_core* functions. Finally, the remaining installation steps for the third party application proceed normally.

1. Determination of Requested Access: Two approaches are leveraged in determining the requested access by the third party application: 1) Manifest provided by the third party developer and 2) Application access analysis. A manifest is a file provided by the third party application developer that outlines the required and optional application privileges; the application developer declares all of the application's database accesses. The manifest is stored in XML format and contains a set of *(subject, object, permissions, required_flag, comments)* policy rule tuples. Figure 44 displays a sample file.

The *required_flag* indicates whether the access is required for the proper execution of the application. If the flag is not set, the *comments* field can be used by the developer

```
<manifest>
        <policy_rule id="pr1">
                <subject>appName</subject>
                <object>birthday_table</object>
                <permission>select</permission>
                <required_flag>0</required_flag>
                <comments>Access is not required; but...</comments>
        </policy_rule>
        <policy_rule id="pr2">
        …
        </policy_rule>
</manifest>
```

Figure 44: Sample manifest file

to elaborate on the optional nature of the access, e.g., "Access to *birthday_table* is not mandatory; but if access is not provided, ages will not be displayed."

Regardless of whether a manifest is available, the application code is statically analyzed at installation time for all database calls. Similarly as for the manifest, the output of the application access analysis is a set of *(subject, object, permissions)* policy rule tuples that describe the third party application database accesses, e.g., *(application_name, database_table_name, read).* In addition to identifying all database calls, all called third party applications functions are identified and their respective iLayer Policies are retrieved from the policy repository.

2. Setting the Policy: The manifest and the output of the application access analysis (to include all called application iLayer Policies) are presented to the CMS administrative user as a series of policy rule tuples. In addition, CMS administrative users are presented with additional information to assist them in making their policy decisions. For each policy rule tuple, a thumbs up or thumbs down policy rule recommendation is presented, where a thumbs up recommends adoption of the policy rule tuple and a thumbs down does not. This recommendation is an indicator of the

community's usage of the policy rule tuple.

Application ID	Granted Accesses (object - permission)								
	sessions - read	*Sessions - write*	*user_roles - read*	*user_roles - write*	*node_revs - read*	*node_revs - write*	...	*files - read*	*files - write*
001	0	1	0	0	0	1		0	0
002	0	0	0	0	0	0		1	0
003	1	1	1	0	0	0		0	1
...									
412	0	1	0	0	0	1		0	1
413	0	0	0	0	0	0		?	?

Figure 45: Historically granted accesses of third party applications

Policy Rule Recommendation: The thumbs up/down policy rule recommendation is based on the maximum likelihood of the set of possible permission combinations for all requested objects based on historically granted accesses. Let A equal the set of all possible accesses that can be requested by third party applications, $A = \{a_1, a_2, \ldots, a_n\}$, where a_i is an object to permission pairing, e.g., *sessions-read*. Let λ equal the set of all previously granted third party application accesses, where $\lambda \subseteq A$. Figure 45 displays a sample of previously or historically granted accesses (Application IDs 001 to 412), where the decision variable:

$$x_i = \begin{cases} 1 & \text{if } a_i \textit{ is granted} \\ 0 & \text{if } \textit{otherwise} \end{cases}$$

When installing a third party application, the set of requested accesses (object-permission pairings) is denoted by R, where $R \subseteq A$. In our framework, the set R is obtained from the application access analysis and / or the manifest. For example, in Figure 45, Application ID 413 is being installed and $R = \{$*files-read, files-write*$\}$.

$\bar{R}$ denotes the set of accesses that were not requested. The accesses that were not requested ($\bar{R}$) are automatically not granted to the third party application.

The policy rule recommendation for the requested accesses is computed based on statistically examining the set λ. Without the loss of generality, let $R = \{a_1, a_2, \ldots, a_{r-1}\}$ and let $\bar{R} = \{a_r, a_{r+1}, \ldots, a_n\}$. The decision is not to grant $\bar{R}$ – therefore, using the decision variable, $\bar{R} = \{x_r = 0, x_{r+1} = 0, \ldots, x_n = 0\}$. For R, the values (or recommendations) for $x_1, x_2, \ldots, x_{r-1}$ are computed by:

$$X = \arg \max_{x_1,\ldots,x_{r-1}} P(x_r = 0, \ldots, x_n = 0 \mid x_1, \ldots, x_{r-1})$$

$X = \{x_1, \ldots, x_{r-1}\}$ is the set of recommended accesses that maximize the conditional probability of the set of accesses that were not requested (or granted) given the possible permission combinations for all requested objects taking into consideration the historically granted accesses λ. In other words, this mechanism chooses the recommendations X that are most probable based on historical accesses. For example, Application ID 413 requests two accesses {*files-read, files-write*}, see Figure 45. There are four possible recommendations, see Figure 46. The conditional probability $P(\bar{R}|X)$ is computed for each possible recommendation combination. In our example, allowing both requested accesses {*files-read, files-write*} are recommended because this combination has the maximum probability.

For each access x_i, the policy rule recommendation is presented to the CMS administrative user in the form of a thumbs up/down where a thumbs up recommends the adoption of the requested object-permission pairing x_i and conversely for a thumbs

files - read	*files - write*	X	$P(\overline{R}\,|X)$
deny	deny	$\{x_1 = 0, x_2 = 0\}$	0
deny	allow	$\{x_1 = 0, x_2 = 1\}$	0
allow	deny	$\{x_1 = 1, x_2 = 0\}$	.2
allow	allow	$\{x_1 = 1, x_2 = 1\}$	.5

Figure 46: Recommendation computation example

down. For example, if $x_i = 1$, a thumbs up is presented to the CMS administrative user signifying a recommendation for this access request and a thumbs down would be presented when $x_i = 0$.

The number of conditional probability computations equal 2^n, where n equal the number of requested accesses. Our research shows that the number of requested accesses n is relatively small and therefore manageable from a computational perspective. Figure 47 shows the distribution of the number of accesses (database table-permission) for 412 Drupal modules (third party applications). The average number of accesses is 2.45 and the median is 2.

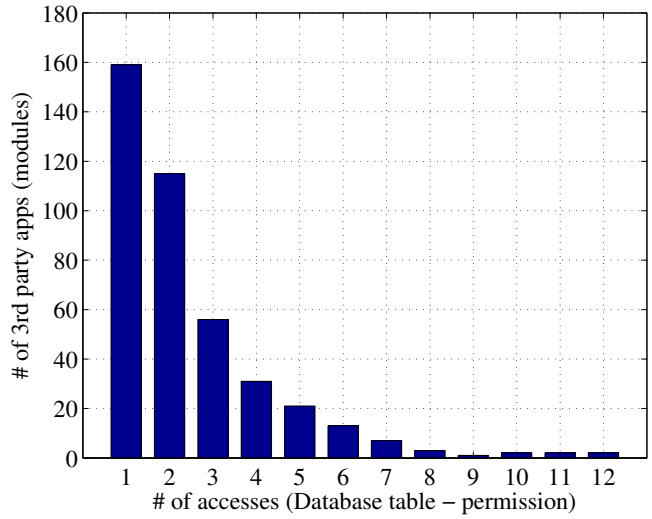


Figure 47: Accesses of Drupal modules

Policy Decision: All the policy rule tuples are presented to the CMS administrative

user who can then can review each and grant the appropriate access as necessary, see Figure 50 for an example screen shot. After reviewing and granting access for each policy rule tuple, the CMS administrative user has two options:

- Approve all selected access: All selected accesses (from both the manifest file and the application access analysis) are approved and the policy is written to the *iLayer_policy* table.
  - Note: Required accesses (as specified in the manifest file) cannot be disallowed. The only option, other than allowing the required accesses, is to "Deny all access".
  - Note: When only application access analysis is presented, some knowledge of the third party application is required; otherwise, selectively disallowing access could lead to unpredictable behavior at execution.
- Deny all access: All access presented is denied and installation of the application is terminated.

3. Function Parsing and Replacement: After the policy is composed and stored in the *iLayer_policy* table, the third party application code is parsed and all instances of *core* functions are replaced with their corresponding *i_core* functions that were added to the CMS Function Library during the iLayer Setup phase. The remaining native installation steps for the third party application are unchanged and the installation of the application proceeds normally.

5.2.2.3 Runtime Enforcement

Upon execution of the third party application, the iLayer Reference Monitor enforces the iLayer Policy that was set at installation time. The default action of the reference monitor is to deny all access, i.e., if there does not exist an explicit permit statement in the policy, the reference monitor denies all access attempts. During execution time, the *i_core* functions are called and thus invoke a policy check for every

database call. The *i_core* function takes the third party application name and the query arguments as its parameter. It will then extract the table name, action (select, delete, insert, or update) from the query arguments and try to establish a match with one of the policy statements in the *iLayer_policy* table. If a match is found, the corresponding *core* function is called and normal operations are allowed to proceed. Otherwise, an *errorHandler* is called to display an access denied error message to the user.

5.2.3 CMS Application Access Control Prototype

5.2.3.1 Drupal Overview

We prototyped our application access control framework on Drupal. Drupal is a popular open source CMS platform capable of providing a wide range of services, from personal web sites to the foundation of a social networking site. Like other CMS platforms, Drupal has a mature user-based access control framework. Also, like other CMS platforms, third party applications allow Drupal web site administrative users to add custom capabilities and features. And, similarly to other CMS platforms, Drupal has very limited third party application access control capabilities – mainly only in the form of file permissions.

Using the concepts of modules, nodes and hooks, Drupal provides an abstracted approach to handling web content and functionality [68]. Drupal is based on a modular framework. Its functionality is provided in the form of these modules or applications. Modules can be part of the base installation – built in core modules. Modules can also be developed by the Drupal community, called contributed modules and previ-

ously referred to as third party applications. See Figure 48. There are thousands of contributed modules available for download. These modules expand the functionality of Drupal. But, they also increase the attack surface and thus potentially increase the risk of compromise.

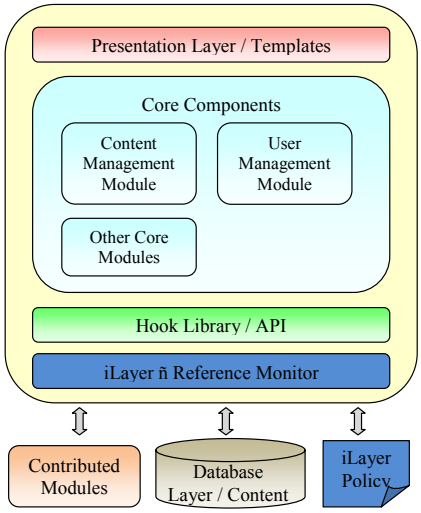


Figure 48: Drupal iLayer architecture

The primary way to access Drupal functionality is via hooks implemented in modules. In the most basic sense, hooks act as functions and are stored in the Hook Library. Drupal functionality is usually delivered in the form of content. A node in Drupal is a piece of content. Drupal supports many different node types, e.g., blog node type, page node type, story node type, etc. The characteristics of each node (or piece of content) is inherited from a node type. Therefore, managing content by node type impacts all nodes of that type. For instance, in order to add a user access rule for a blog entry node, the administrative user only needs to modify the blog node type and the change will propagate to all of the other blogs on the site.

5.2.3.2 Drupal iLayer Setup

The first step in implementing the iLayer Architecture in Drupal is setting up the policy table. We created a table called *iLayer_policy*. Policy statements are stored in the policy table and are made up of three components: the module name (subject), the database table name (object) and the access request (permission). Next, all the core Drupal hooks (functions) that perform database calls are identified and refactored. We chose to refactor the *db_query()* hook for our prototype because *db_query()* is the main Drupal hook that every module uses in order to execute a database call / query. We added the refactored *db_query()* hook, *idb_query()*, to the Drupal Hook Library. It has the same arguments as *db_query()* with the addition of the calling module name. The *idb_query()* hook performs a policy check. If the policy is not violated, *idb_query()* forwards the call to the original *db_query()* hook and operation proceeds normally. If the policy is violated, access to the requested table is denied and an error message is returned. See Figure 49 for a code sample of *db_query()* and its corresponding *idb_query()* hook.

5.2.3.3 Contributed Module Installation

After the iLayer Architecture has been setup on the Drupal platform, we then can install third party applications or contributed modules. One Drupal module we installed in our prototype was Flash Node [20]. Flash Node allows CMS administrative users to easily add flash content to their sites. As part of the module installation process, the access control policy is presented to the CMS administrative user. Since Flash Node doesn't come with a developer generated manifest file that declares the re-

Original db_query()

```
function db_query($query) {
  …
  //extract table name and
  action from the $query

  return _db_query($query);
}
```

Refactored db_query() to idb_query()

```
function idb_query($moduleID, $query) {
    …
    //extract table name and
    action from the $query
    //loop for all table names
    foreach($strArray as $key => $table_name) {
        $action = 0;    // 0 is read, 1 is write
        if($strArray[0] ==”select”) $action = 0; else $action = 1;
        …
        $result = db_query("SELECT count(*) as cnt
            FROM {iLayer_policy} WHERE
            moduleID = '%s' and
            tableName = '%s' and
            grantAccess = %d",
            $moduleID, $table_name, $action);
        $row = db_fetch_object($result);
        if($row->cnt  < 1)
            //no match found; display error; return null
    }
    //after all matches found, forward call to original _db_query
    return _db_query($query);
}
```

Figure 49: Refactoring db_query function

quired and optional accesses for the module, application access analysis is performed to determine the requisite database calls for Flash Node. The output of the static analysis is presented to the CMS administrative user for review and action, see Figure 50. The CMS administrative user is presented the module policy in the form of a subject (Flash Node), object (Table Name) and permissions (select, delete, insert, update) policy rule tuple. The CMS administrative user reviews the policy, to include the policy rule recommendation in the form of a thumbs up/thumbs down, where thumbs up means the policy rule is recommended and conversely for a thumbs down. After which, the CMS administrative user grants the appropriate access by checking the box for each selected policy rule tuple and then clicks on *Approve all selected access*. The policy is written to the *iLayer_policy* table. After which, all instances of *db_query()* in the Flash Node module are replaced by the refactored *idb_query()* hook. The remaining native Flash Node module installation steps proceed normally.

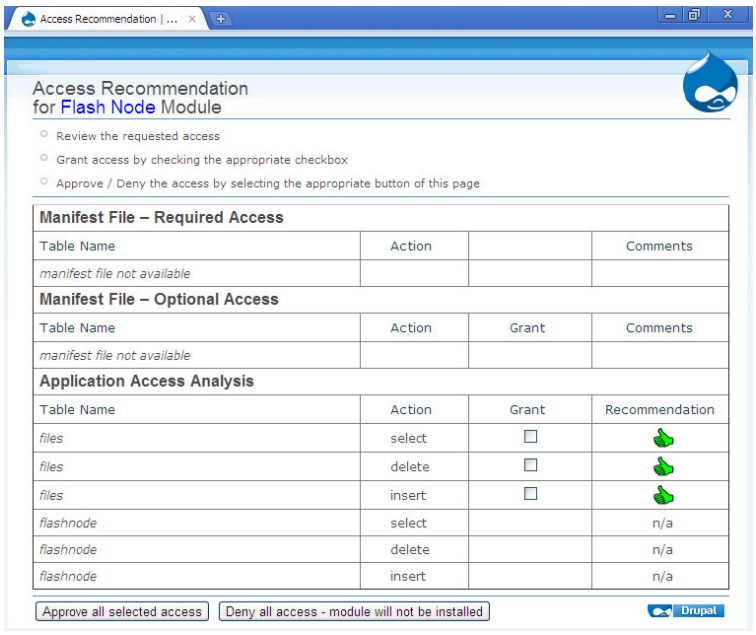


Figure 50: iLayer policy review page

5.2.3.4 Runtime Enforcement

Upon execution of the Flash Node module, database accesses are made by calling *idb_query()*. The *idb_query()* hook takes the module name and query arguments as its parameters. The called database table name and action (select, delete, insert, or update) are extracted from the query arguments. These two parameters coupled with the module name are used as criteria to find a policy statement match in the *iLayer_policy* table. If a matching policy statement is found, access to the table is granted and execution proceeds normally. If a policy statement is not found, an error message is displayed and access to the requested table is disallowed.

CHAPTER 6: CONCLUSION

Access control mechanisms, including their policies, lie between the consumers of the information (both viewers and third party applications) and the actual information. User Centric Policy Management enhances these mechanism. It primarily consists of *Policy Management Assistance Tools* with *User Centric Characteristics* that enable the user to better manage their access control policies. Our new access control paradigm provides a more usable access control framework that is effective, efficient and satisfying to the user. We proposed both Viewer and Application Policy Management Assistance tools.

Viewer Policy Management Tools:

Policy Manager assists users in managing access to their online information. We proposed a user centric approach, based on supervised learning mechanisms, to decide on trust and access control settings for each user's friends. We incorporated knowledge from others in the social network to enhance the supervised learning results.

Assisted Friend Grouping leverages proven clustering techniques, with measurable alignment with user intentions, to assist users in grouping their friends for policy management purposes. Our approach demonstrated reduced grouping times and improvements in user centric characteristics over traditional group based policy management approaches.

Same-As Subject Management leverages a user's memory and opinion of their

friends to set policies for other similar friends. Our visual policy editor uses friend recognition and minimal task interruption to obtain substantial reductions in policy authoring times and more conservative policies. In addition, Same-As Subject Management was positively perceived by users over traditional group based policy management approaches.

Example Friend Selection introduces two techniques for aiding users in selecting their example friends that are used in developing policy templates for Same-As Subject Management. Both techniques reduced policy authoring times and exhibited improved user centric characteristics.

Same-As Object Management leverages a user's memory and perception of their objects for setting permissions for other similar objects. Our access control framework's policy management mechanism is more flexible; users were able to author more expressive policies that aligned with their mental model and intentions. These policies are also more readable than ones created using more traditional grouping based access control frameworks. Because of the improved expressiveness and readability, policies are more conservative (less permissive) resulting in better security. Also, policy authoring time is reduced and users perceived Same-As Object Management to be easier to use.

Application Policy Management Tools:

We introduced a general framework describing *Social Networks Connect Services* and explored the Facebook Platform, Google Friend Connect, and MySpaceID implementations. We also proposed a number of open research security challenges, discussing them, presenting select related work, and offering some ideas that may

lead to solutions.

We presented *iLayer*: an Application Access Control Framework. Our framework is based on least privilege and provides web and social networking sites protection from third party applications. In addition, our framework provides administrators third party application policy setting functionality, including a policy setting recommendation capability that enhances knowledge in making policy decisions.

REFERENCES

[1] Java universal network graph framework. `jung.sourceforge.net`, 2009.

[2] Weka machine learning project. `www.cs.waikato.ac.nz/~ml/index.html`, 2009.

[3] Cert secure coding. `www.cert.org/secure-coding`, 2010.

[4] Google buzz criticized for disclosing gmail contacts. `www.pcworld.com/businesscenter/article/189081`, 2010.

[5] Acquisti, A., and Gross, R. Imagined communities: Awareness, information sharing and privacy on the facebook. In *Privacy Enhancing Technologies* (2006).

[6] Acquisti, A., and Grossklags, J. Privacy and rationality in individual decision making. *IEEE Security and Privacy* (2005).

[7] Barbosa, L., and Freire, J. Combining classifiers to identify online databases. In *Proceedings of the Conference on World Wide Web (WWW)* (2007).

[8] Birge, C. Enhancing research into usable privacy and security. In *Special Interest Group on Design of Communication (SIGDOC)* (2009).

[9] Bonneau, J., and Preibusch, S. The privacy jungle: On the market for data protection in social networks. In *The Workshop on the Economics of Information Security (WEIS)* (2009).

[10] Borgatti, S. P., and Everett, M. G. A graph-theoretic perspective on centrality. *Social Networks* (2006).

[11] Clauset, A., Newman, M. E. J., and Moore, C. Finding community structure in very large networks. *Physical Review E* (2004).

[12] Dhamija, R., and Perrig, A. Deja vu: A user study using images for authentication. In *Proceedings of the conference on USENIX Security Symposium* (2000).

[13] Drupal.org. Drupal - open source cms. `Drupal.org`, 2010.

[14] Dwyer, C., Hiltz, S. R., and Passerini, K. Trust and privacy concern within social networking sites: A comparison of facebook and myspace. In *Proceedings of the Americas Conference on Information Systems (AMCIS)* (2007).

[15] Egelman, S., Oates, A., and Krishnamurthi, S. Oops, i did it again: mitigating repeated access control errors on facebook. In *Computer Human Interaction (CHI)* (2011).

[16] eMarketer. Days of double-digit growth in social network users are over. `www.eMarketer.com`, 2011.

[17] Facebook. Facebook statistics. `www.facebook.com/press/statistics`, 2011.

[18] Ferraiolo, D., and Kuhn, R. Role-based access control. In *Proceedings of the National Computer Security Conference* (1992).

[19] Fowler, M. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley, 1999.

[20] Greenfield, S. Flash node. `drupal.org/project/flashnode`, 2010.

[21] Hammer-Lahav, E. E., Rocordon, D., and Hardt, D. The oauth 2.0 protocol. *`tools. ietf. org/ html/ draft-ietf-oauth-v2-08`* (2010).

[22] Hubert, L., and Arabie, P. Comparing partitions. *Journal of Classification* (1985).

[23] Iqbal, S. T., and Bailey, B. P. Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *Computer Human Interaction (CHI)* (2005).

[24] Jones, N. A., Ross, H., Lynam, T., Perez, P., and Leitch, A. Mental models: An interdisciplinary synthesis of theory and methods. In *Ecology and Society* (2011).

[25] Jones, Q., Grandhi, S. A., Whittaker, S., Chivakula, K., and Terveen, L. Putting systems into place: a qualitative study of design requirements for location-aware community systems. In *Proceedings of the conference on Computer Supported Cooperative Work (CSCW)* (2004).

[26] Jones, S., and O'Neill, E. Feasibility of structural network clustering for group-based privacy control in social networks. In *Symposium on Usable Privacy and Security (SOUPS)* (2010).

[27] Joomla! Secure coding guidelines. `docs.joomla.org/Secure_coding_guidelines`, 2010.

[28] Joomla.org. Joomla! `www.joomla.org`, 2010.

[29] Jsang, A., and Pope, S. User centric identity management. In *Asia Pacific Information Technology Security Conference, AusCERT2005, Austrailia* (2005), pp. 77–89.

[30] Jsang, A., Zomai, M. A., and Suriadi, S. Usability and privacy in identity management architectures. In *Proceedings of the Australasian symposium on ACSW frontiers* (2007).

[31] Kittler, J., Hatef, M., Duin, R. P., and Matas, J. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1998).

[32] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* (1999).

[33] Krasnova, H., Gnther, O., Spiekermann, S., and Koroleva, K. Privacy concerns and identity in online social networks. *Identity in the Information Society* (2009).

[34] Krishnamurthy, B., and Wills, C. E. On the leakage of personally identifiable information via online social networks. *Special Interest Group on Data Communications (SIGCOMM)* (2010).

[35] Kumaraguru, P., and Cranor, L. F. Privacy indexes: A survey of westin's studies. *ISRI Technical Report* (2005).

[36] Lampson, B. Privacy and security: Usable security: how to get it. *Communications of the ACM* (2009).

[37] Last.fm. Last fm platform. `www.last.fm`, 2009.

[38] Lederer, S., Hong, J. I., Dey, A. K., and Landay, J. A. Personal privacy through understanding and action: five pitfalls for designers. *Personal and Ubiquitous Computing* (2004).

[39] Lewis, K., Kaufman, J., and Christakis, N. The taste for privacy: An analysis of college student privacy settings in an online social network. *Journal of Computer-Mediated Communication* (2008).

[40] Liu, B. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer, 2007.

[41] Liu, Y., Gummadi, K. P., Krishnamurthy, B., and Mislove, A. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the Internet Measurement Conference (IMC)* (2011).

[42] Madejski, M., Johnson, M., and Bellovin, S. A study of privacy settings errors in an online social network. In *Pervasive Computing and Communications Workshops* (2012).

[43] Maliki, T. E., and Seigneur, J.-M. A survey of user-centric identity management technologies. In *Proceedings of the Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE)* (2007).

[44] Mannan, M., and van Oorschot, P. C. Privacy-enhanced sharing of personal content on the web. In *Proceeding of the conference on World Wide Web (WWW)* (2008).

[45] Meneely, A., Williams, L., Snipes, W., and Osborne, J. Predicting failures with developer networks and social network analysis. In *Proceedings of the Symposium on Foundations of Software Engineering (SIGSOFT)* (2008).

[46] Moyer, M., and Abamad, M. Generalized role-based access control. In *The conference on Distributed Computing Systems* (2001).

[47] Newman, M. E. J. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E* (2001).

[48] Newman, M. E. J. The structure and function of complex networks. *Society for Industrial and Applied Mathematics Review* (2003).

[49] Newman, M. E. J. Fast algorithm for detecting community structure in networks. *Physical Review E* (2004).

[50] Norberg, P. A., Horne, D. R., and Horne, D. A. The Privacy Paradox: Personal Information Disclosure Intentions versus Behaviors. *Journal of Consumer Affairs* (2007).

[51] Olson, J. S., Grudin, J., and Horvitz, E. A study of preferences for sharing and privacy. In *Computer Human Interaction (CHI) Extended Abstracts* (2005).

[52] OWASP. Open web application security project. `www.owasp.org`, 2010.

[53] Perols, J., Chari, K., and Agrawal, M. Information market-based decision fusion. *Management Science* (2009).

[54] Plone.org. Plone cms: Open source content management. `Plone.org`, 2010.

[55] Reeder, R. W., Bauer, L., Cranor, L. F., Reiter, M. K., and Vaniea, K. More than skin deep: measuring effects of the underlying model on access-control system usability. In *Proceedings of the conference on Human Factors in Computing Systems* (2011).

[56] Saltzer, J., and Schroeder, M. The protection of information in computer systems. *Proceedings of the IEEE* (1975).

[57] Saltzer, J. H. Protection and the control of information sharing in multics. *Communications of the ACM* (1974).

[58] Sandhu, R., Coyne, E. J., Feinstein, H. L., and Youman, C. E. Role-based access control models. *IEEE Computer* (1996).

[59] Sandhu, R., Ferraiolo, D., and Kuhn, R. The nist model for role-based access control: Towards a unified standard. In *Proceedings of the ACM Workshop on Role-Based Access Control* (2000).

[60] Seacord, R. C. *The CERT C Secure Coding Standard*. Addison-Wesley Professional, 2008.

[61] Short, J. F., and Hughes, L. A. Studying youth gangs / edited by james f. short, jr. and lorine a. hughes, 2006.

[62] Strater, K., and Lipford, H. R. Strategies and struggles with privacy in an online social networking community. In *Proceedings of the British Human Computer Interfaces Conference* (2008).

[63] Sun, S.-T., Hawkey, K., and Beznosov, K. Secure web 2.0 content sharing beyond walled gardens. In *Annual Computer Security Applications Conference (ACSAC)* (2009).

[64] Sutterer, M., Droegehorn, O., and David, K. Making a case for situation-dependent user profiles in context-aware environments. In *Proceedings of the Workshop on Middleware for Next-generation Converged Networks and Applications (MNCNA)* (2007).

[65] Swidey, N. Friends in a facebook world. *Globe Newspaper Company* (2008).

[66] Theofanos, M., and Pfleeger, S. Guest editors' introduction: Shouldn't all security be usable? *IEEE Security & Privacy* (2011).

[67] Tootoonchian, A., Gollu, K. K., Saroiu, S., Ganjali, Y., and Wolman, A. Lockr: social access control for web 2.0. In *Proceedings of the Workshop on Online Social Networks (WOSP)* (2008).

[68] VanDyk, J. K., and Buytaert, D. *Pro Drupal Development*. Apress, 2007.

[69] Vrancken, B., and Zeltsan, Z. Internet draft on oauth recursive delegation. *tools. ietf. org/html/ draft-vrancken-oauth-redelegation-00* (2009).

[70] Wang, G. A., Chen, H., Xu, J. J., and Atabakhsh, H. Automatically detecting criminal identity deception: an adaptive detection algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* (2006).

[71] Wikipedia. Social network. `en.wikipedia.org/wiki/Social_network`, 2009.

[72] Witten, I. H., and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[73] WordPress.org. Wordpress. `WordPress.org`, 2010.

[74] Xu, J., Wang, G. A., Li, J., and Chau, M. Complex problem solving: Identity matching based on social contextual information. *Journal of the Association for Information Systems* (2007).