Design and Implementation of FAITH, an Experimental System to Intercept and Manipulate Online Social Informatics

Ruaylong Lee, Roozbeh Nia, Jason Hsu, Karl N. Levitt, Jeff Rowe, S. Felix Wu
Computer Science Department, University of California, Davis
{rulee, rvnia, jlhsu, knlevitt, jbrowe, sfwu}@ucdavis.edu

Shaozhi Ye
Google Inc.
yeshao@google.com

Abstract — **Social informatics is the core of Facebook's business and is its most valuable asset which consists of the social graph and the private data of over 500 million users. However, without secure methods of managing this data, Facebook has become vulnerable to privacy risks and devaluation. In Facebook's model, users are asked upon access to grant applications the required permissions without sufficient knowledge of the applications' intentions. As a result, if they are deceived, users risk the exposure of sensitive and personal data. This paper presents a system dubbed FAITH (Facebook Applications: Identification, Transformation & Hypervisor) to mitigate or eliminate these issues by enhancing the management of social data. First, FAITH allows users to adjust the visibility of their social informatics for each individual application depending on how much they trust the application. Users can configure FAITH to let non-trusted applications run with the least privileges (least amount of social informatics) to minimize potential privacy leaks. Second, FAITH logs the activities of applications to assist users in making more secure decisions. Users can closely monitor each activity performed by applications to adjust their privacy settings more securely. Third, FAITH allows users to transform their social graph such that different applications see different social graphs preventing the formation of friendship inflation caused by applications. The implementation of FAITH only needs the resources and tools available to the public by Facebook and requires no further cooperation from the social network. FAITH is a prototype system: the design and concept can be extended to secure other OSNs (Online Social Networks). Currently, FAITH contains thirteen Facebook social applications and has been officially released for public usage with approximately two hundred monthly active users as of now.**

Index Terms—Facebook; social network sites; privacy protection; Facebook applications

I. INTRODUCTION & MOTIVATIONS

WHILE early OSNs (Online Social Networks) can be traced back to around 1995 or earlier, the majority of current OSNs had sprung up during the last decade. Among them, Facebook is the most successful and visited of them all. In May 2007, Facebook released the Facebook platform providing a set of APIs (Application Programming Interface) and tools to allow third-party applications and websites to leverage Facebook's social informatics which is the social graph (users and friendships) and users' stored data (photos, events, and wall posts). Since the inception of the platform, the number of Facebook users has dramatically boosted. The site reached 100 millions users in August 2008 and 500 million users by July 2010. The top three most popular applications played by these users: Farmville, Birthday Cards, and Café World reached more than 83 million, 47 million and 30 million users respectively [1]. In 2010, the number of active applications and websites reached 550,000 and 250,000 users. There are more than one million application developers from over 180 countries. Each month, more than 70 percent of Facebook users engage with platform applications, and over 100 million users engage with Facebook Connect websites [2]. While Facebook applications provide intensive online social interaction, communication and fascinating content; they also contribute immensely to Facebook's unprecedented growth. The majority of them are in fact developed by individual developers or organizations not sponsored by Facebook. To ensure that social informatics are retrieved properly and securely, Facebook adopted the OAuth 2.0 protocol to enforce authentication [3,4]. The data fetching under this protocol involves three steps. The first is user authentication, which requires users to log in with their Facebook accounts to ensure that they are exactly who they claim to be. The second step is application authorization. This step occurs when users access platform applications. Users need to grant permission to any application wishing to access information but also any extended permissions required by the application. The last step is application authentication. This ensures that only approved applications have access to the data of users. Facebook then issues access tokens to these applications. These tokens allow applications to make Web API calls on behalf of users. Although users are allowed to determine under the protocol which applications have access to which portion of their social informatics, the data is still vulnerable to privacy disclosure. Many users have been found victimized in unintended social information sharing and exposure of sensitive personal data [5].

First, when users are asked to grant applications the required permissions upon access, the majority of them possess insufficient knowledge of applications' intentions to make secure decisions. The platform misses the point that users need adequate privacy-related knowledge of applications to make these decisions. Facebook merely offers the description written by the developers and the fan pages of the applications which are inadequate to aide users. Users that are concerned about privacy are forced to reluctantly grant the required permissions. Although Facebook allows users to remove any granted permissions, without appropriate security and privacy reports, users once again encounter the same challenges. For instance, assume a user grants a free game application the permissions to access to the entire social informatics of the user offline, which allows the application to execute authorized operations on behalf of the user at any time even when the user is not logged in. While the application appears benign and provides various interesting games, its objective is to steal sensitive and personal data of the user. With the Facebook platform, the user has no method of detecting suspicious activities and would be left wondering why producing the contents require the permissions asked. What often occurs is that when users realize they have been victimized by privacy leaks, they have no clue in identifying from where and what information was leaked. This example makes clear that revealing social informatics solely according to granted permissions is bad management practice on Facebook's part. Facebook's platform allows an application with the required permissions to perform whatever they intend with the social informatics on behalf of the users without taking any other factors into account such as how many users the application have, how long the users have been using the application, and how many users have blocked the application. Under this vulnerable protocol, what is likely to happen is that when malicious applications are able to deceive users, the entire social informatics of the victims are divulged to unauthorized parties. In an effort to minimize potential disasters, a trust factor should be taken into consideration under this scenario.

Friendship inflation is another problem caused by Facebook's mismanagement. Friendship inflation refers to the practice that users maintain a greater amount of friends in OSNs than they really have in real life. It is a significant OSN-related issue, which is believed to devalue the significance of social informatics and eventually lead to the decline of OSNs. Research shows that there are multiple factors which cause such inflation such as feeling uncomfortable to reject friendship requests, appearing popular with more friends in profile, getting access to private profiles of unknown users, and taking advantage of applications [6]. In this paper, we focus on application-related factors. Since, the number of users is proportional to the amount of profits, many applications create incentives to encourage users to get their friends to join the applications and some even make it a requirement to advance to the next level in certain games. Take Restaurant City, a prevalent Facebook restaurant simulation application, for instance. It is an application where users run virtual restaurants of their own, hire their friends as chefs, and add their friends as waiters or janitors. Moreover, users receive a free ingredient as a bonus when they visit the virtual restaurants of their friends for the very first time. Free coins and ingredients are also given when they help clean up their friends' restaurants. Ingredients are important as they allow users to learn new dishes and level up the dishes they have already learned. We can see from the rules that the more friends a user has to play the game with, the more free ingredients and coins the user is likely to get. While the Restaurant City is merely one of the prevalent applications with various incentives to acquire more users, the applications together create a force to push users to eagerly befriend other users even when they do not know each other in real life. This phenomenon unfortunately accelerates the formation of friendship inflation, which not only in the short term devalues the significance of the social informatics but also in the long term causes the decline of Facebook. Application incentives such as those of Restaurant City become an issue mainly because of the way Facebook manages users' social informatics. Facebook's platform disallows users from providing different social graphs to different applications. As a result, when a user intends to befriend another user for an application, the user has to do it at Facebook's level instead of at the application's level. To solve this problem, a tool to allow users to transform their social graph for each individual application is essential. Not only can such tool prevent or at least minimize the formation of friendship inflation, but can also preserve the value of social informatics.

In this paper, we propose a system, FAITH (Facebook Applications: Identification, Transformation & Hypervisor.) When applications initiate requests to access social information, these requests are sent to FAITH instead of Facebook. FAITH retrieves the necessary resources from Facebook first, logs and transforms the original social information before sending it back to applications. The logging and transforming of data ensures that social informatics is managed more securely and transparently. The paper is organized as follows: a brief introduction to the Facebook platform in Section 2, a description of the architectural design of FAITH in Section 3 followed by details of the implementation and an evaluation of FAITH in Section 4, and finally we conclude the paper in Section 5.

II. BACKGROUND

The Facebook platform offers a set of APIs and SDKs to allow third party applications and web sites to fetch social informatics programmatically through official Facebook SDKs, available in most major server and client languages like PHP, Python, ASP, and Javascript. The platform offers three different application types, FBML, IFrame and Facebook Connect. The three types differ mostly in the display mechanism. FBML Canvas applications are rendered by Facebook servers utilizing the contents of applications on remote servers. In contrary, IFrame Canvas applications and Facebook Connect are directly rendered by application servers. To clarify the differences, the following introduces the data flow of each type.

Figure 1.1 illustrates the data flow of a page load of FBML. When a user visits an FBML Canvas application, Facebook sends a request to the application for the content. While the application is processing, it may make multiple API calls to fetch the social informatics of the user. Once the task is complete, the content is delivered back to Facebook first and then to the user.

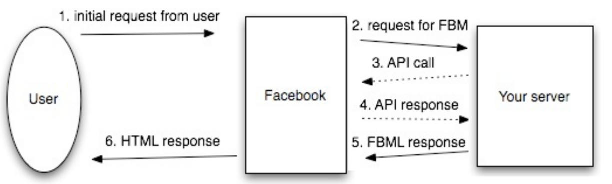


Figure 1.1 Facebook Developers, Performance, February 1, 2011 Canvas Applications Information Flow
[Online Image] http://developers.facebook.com/docs/guides/performance

Different from FBML, Facebook Connect and IFrame may also fetch the social informatics on the client side with JavaScript. In contrary, FBML only fetches on the server-side since it does not support JavaScript. The data flow of IFrame applications are demonstrated in figure 1.2. To access an IFrame Canvas application, a user would first open a browser and navigate to the URL of the application, in a form similar to http://apps.facebook.com/test_app. The initial request to Facebook causes an iframe to open in the browser for the application to display the content. The browser sends another request to the application server. Similarly, the application server may make multiple API calls to fetch the social informatics from Facebook while producing the content. Once completed, the content is delivered back to the iframe in the browser directly instead of through Facebook. Facebook Connect works the same as IFrame except the content is not shown in an iframe [7].

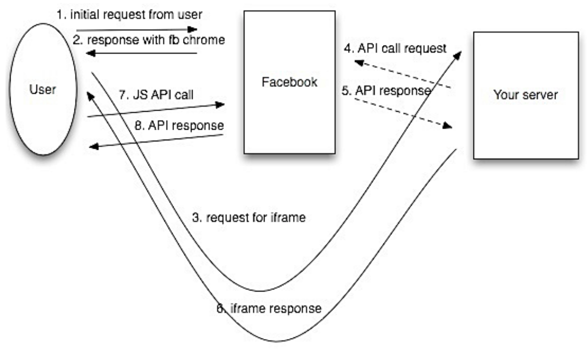


Figure 1.2 Facebook Developers, Performance, February 1, 2011 Websites and IFrame Canvas Application Information Flow
[Online Image] http://developers.facebook.com/docs/guides/performance

III. THE ARCHITECTURE OF FAITH

Figure 2 illustrates the architecture of FAITH. Applications function as social informatics consumers, which leverage social information to provide valuable online social interactions among users. In contrary, Facebook functions as a social informatics provider, which offers FAITH its social graph and informatics. FAITH functions differently depending on different points of view. To Facebook, FAITH is nothing but an ordinary application fetching social informatics. To applications, FAITH supplies the transformed social informatics upon request. From the users' perspective, FAITH is a multi-functional application-level proxy. It transforms and logs the social informatics upon users' requests to manage social information more securely and transparently. Chatroom, Social Wiki [8], Calendar and SoEmail [9] shown in Figure 2 are applications that operate behind FAITH. While there are only four applications in the figure, please note that there are no restrictions on how many applications FAITH is capable of integrating with.

FAITH is built on top of two existing web services [10], DSL kernel [11] and Privacy Shield [12]. FAITH communicates with these services through the SOAP protocol. The API methods of those services are included as part of FAITH SDK to give developers more resources in creating robust and interactive applications. DSL kernel is a web service built on top of the Facebook social graph. The service publishes its API via the SOAP protocol, and helps third party applications leverage the power of the Facebook social graph. DSL kernel consists of two components: trust management and social router. FAITH allows users to specify rules which transform their social informatics used by DSL kernel such that the social router utilizes the transformed social graph instead of the original Facebook social graph. One use would be giving users more options on which social paths to utilize when delivering emails. Wall posting is frequently-used by Facebook users for information sharing purposes. Facebook allows users to adjust the privacy settings of each post they make through both Facebook Wall and applications. However, users need to determine the setting each time they post a message. Facebook offers no suggestions to help users on that perspective. Privacy Shield is a web service, which suggests privacy settings for wall posts based on previous online social interactions among users and their friends. FAITH allows users to review and edit the recommendations they receive from the service which then uses those settings as privacy parameters of wall posts to control the visibility of those posts on the Facebook wall.

During the process of producing content, applications may send multiple requests to FAITH to access the social informatics of Facebook or the functionalities of web services. In the case of requesting social informatics, FAITH sends requests to Facebook and then logs and passes back the transformed informatics to applications. In the case of utilizing the functionality of the web services, FAITH sends other requests to services, and also logs and passes back the results to applications.

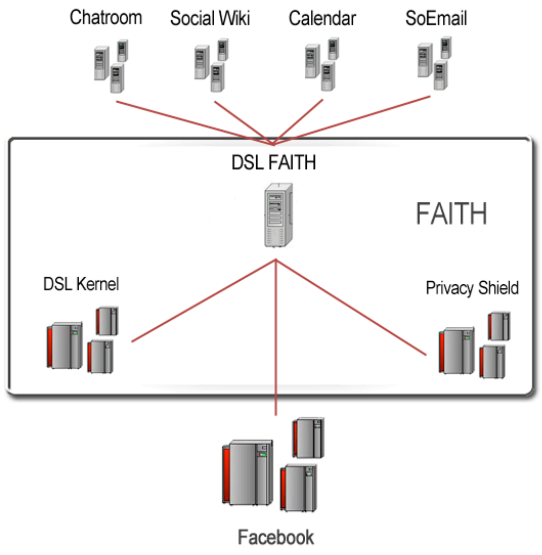


Figure 2

A. The Information Flow of FAITH Applications

The data flow of Facebook applications (illustrated in figure 1.1) is different from the flow of the FAITH-integrated applications in that FAITH is added between Facebook and the applications. Figure 3.1 illustrates the data flow of a page load of a FAITH-integrated FBML application. When a user accesses FAITH to start navigation, Facebook sends a request to FAITH, and then FAITH sends another request to the application server. While processing the content, the application may make multiple API calls to fetch the social informatics. FAITH needs to authenticate and process those calls. If valid, FAITH initiates corresponding API calls to fetch the social informatics from Facebook, and then transforms and logs the results before sending them back to the application server. Once the application completes the content, it is delivered back to the user from FAITH and then Facebook.

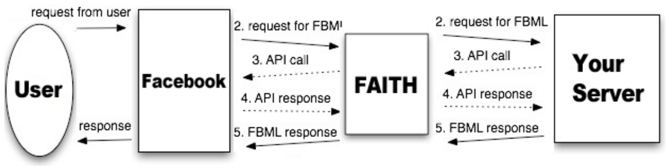


Figure 3.1 Facebook Developers, Performance, February 1, 2011 Canvas Applications Information Flow
[Online Image] http://developers.facebook.com/docs/guides/performance

The data flow of FAITH-integrated IFrame applications are demonstrated in Figure 3.2. Similar to original ones illustrated in Figure 1.2, FAITH is added between Facebook and applications. When a user accesses FAITH to start navigation with a browser, Facebook opens an iframe in the browser to display the content from the application, and then the browser sends another request to FAITH. Upon reception, FAITH sends a request to the application server. While processing the content, the application may make multiple Web API calls to fetch the social informatics. FAITH authenticates those requests. If valid, it sends corresponding API calls to fetch the data, and then transforms and records the results before sending them back to the application server. After the application completes the content, it is sent back to the browser only through FAITH.

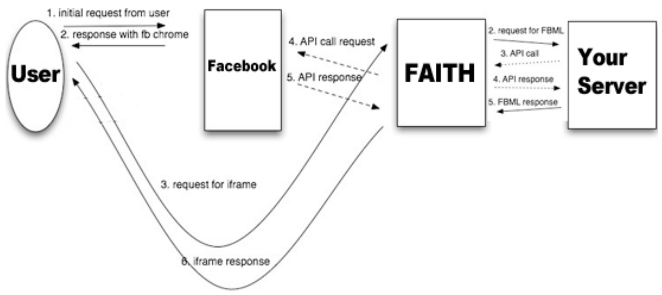


Figure 3.2 Facebook Developers, Performance, February 1, 2011 Websites and IFrame Canvas Applications Information Flow
[Online Image] http://developers.facebook.com/docs/guides/performance

B. Features and Improvements

1. Social Informatics Logging & Replay

In order to make the activities of applications transparent to users, FAITH records the social informatics coming through it including both the data sent to applications (API calls and the results of the calls) and the data produced by the applications (HTML source codes). Logging occurs when users are logged in or not (fetching the data with offline permissions granted by the users). The logs allow users to monitor and analyze the complete activities of the applications to know when, what, and how often their social informatics is retrieved. For each API log, replay allows users to simulate the API method call on behalf of the application such that the user can compare the results of the API call in different time periods.

2. Privacy Control at the level of API & Application

FAITH allows users to disable any individual FAITH-supported API methods: Old REST API, Graph API [3] and DSL API [11]. Once disabled, applications are no longer able to utilize the methods to retrieve the social informatics. FAITH also adds more flexibility to the control. After blocking an API method, users may exclude any applications such that the excluded ones can continue to utilize the disabled API method to fetch the social informatics of the users. Blocking applications functions the same except that it is in the scope of the application. When blocked, the applications are no longer able to fetch the social informatics.

3. Social Graph Transformation (SGT)

The concept of SGT is to transform Facebook's social graph in ways specified by users such that applications, websites and devices see the transformed social graph instead of the original one. The social graph is a graphical representation of social relationships and links all of Facebook's users together. SGT is a technique to defend against privacy breaches as well as to protect the value of social informatics. The following illustrates how SGT works in FAITH. Assume Alice and Bob are Facebook users, but not Facebook friends. If Alice virtually befriends Bob through FAITH, and Bob has confirmed the request, applications under FAITH see Alice and Bob as Facebook friends after SGT. If Alice and Bob are Facebook friends, and Alice hides from Bob through FAITH, applications see them as not Facebook friends. Different from adding, hiding does not need confirmation to become effective. In addition, for each virtual friendship and hidden real friendship, users are allowed to exclude any applications and friends such that the virtual and hidden friendships are not applied to the excluded ones. To better illustrate the concepts, the following examples demonstrate various scenarios of SGT. Assume Calendar is an application behind FAITH, and Alice, Bob and Carol are Facebook users (only Alice and Bob are Facebook friends). The social graph is as shown in Figure 4.1. (1) If Alice virtually befriends Carol through FAITH, and Carol has confirmed the request, the social graph presented to applications becomes as illustrated in Figure 4.2, where Alice are friends with both Bob and Carol. (2) Assume the same situation as the previous example, but Alice excludes Bob from the virtual friendship (Alice may exclude any of her friends including herself). When Bob accesses any application through FAITH, the applications see the Facebook social graph as illustrated in Figure 4.1 while other users see the social graph as Figure 4.2 through any application. (3) Assume the same situation as (1), but Alice excludes Calendar from the virtual friendship (Alice is able to exclude any application). When any user accesses Calendar, it sees the social graph as illustrated in Figure 4.1 while other applications see the social graph in Figure 4.2 with any user. (4) If Alice hides from Bob through FAITH, the social graph presented to applications becomes as illustrated in Figure 4.3, where Alice is not friends with both Bob and Carol. (5) Assume the same situation as the previous example, but Alice excludes Bob from the hiding (Alice may exclude any of her friends including herself). When Bob accesses any applications through FAITH, the applications see the social graph as illustrated in Figure 4.1 while other users see the social graph as Figure 4.3 through any application. (6) Assume the same situation as (4), but Alice excludes Calendar from the hiding (Alice may exclude any application). When any user accesses Calendar, it sees the social graph as illustrated in Figure 4.1 while other applications see as Figure 4.3 with any users.

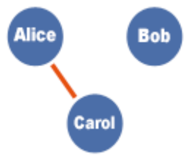


Figure 4.1

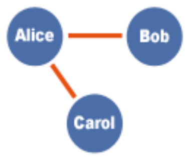


Figure 4.2

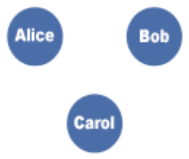


Figure 4.3

4. *Social Informatics Management Improvements*

FAITH is effective in defending against privacy breaches and social informatics devaluation caused by Facebook's mismanagement. First, FAITH allows users to adjust the visibility of their social informatics for each individual application depending on how much they trust the application such that non-trusted applications run with the least privileges (or least amount of social informatics). With the Facebook platform, if malicious applications are able to deceive users, the social informatics of the victims is completely divulged to unauthorized parties. In contrary, privacy control and the SGT of FAITH can work together to allow users to restrict the visibility of their social information for new and non-trusted applications. Users can use the privacy control of FAITH to disable certain API methods that are often utilized to fetch more sensitive social informatics, and use SGT to restrict the visibility of their social graph. Instead of showing the entire social graph, users may just reveal a small portion to unknown applications while set no limitations on trusted ones. Take the previously mentioned free game application for instance. With the Facebook platform, users upon accessing the application have to either grant all the permissions or leave the application. Both choices are neither desirable nor favorable. With FAITH, users with special privacy concerns on their Facebook events for example can simply disable events.get API so that the application has no access to the events of the users. Users who care about the privacy of their friends may hide their real Facebook friendships so that the application sees only a small portion of their friends. In this way, even if the application has permissions to those resources, it will still see the restricted data instead of all of it minimizing potential damages.

Moreover, FAITH offers complete application log data to assist users in making relevant privacy decisions. FAITH closely monitors and records the activities of applications to give users the transparency needed to be aware of when, what and how often their social information is fetched. If the users find suspicious activity, they may further disable those API methods or applications to prevent potential damages, and open more resources if applications are found to be benign. For instance, if the free game application is running under FAITH, users would be able to identity suspicious activities with social informatics logging. Users may find that the social informatics fetched is not relevant to the content shown and the application excessively fetches those data which is abnormal to regular game applications. Although FAITH is unable to prevent privacy breaches that have already occurred in the first place, the system helps to minimize potential future damages. Moreover, replay allows users to test the current privacy settings by simulating the API calls on behalf of the applications. Users can compare the results of the calls and adjust the settings to meet their privacy demands.

The capability of minimizing the formation of friendship inflation caused by applications is another management improvement of FAITH. While Restaurant City was an appropriate example, FarmVille, an extremely prevalent social Facebook application, is another example, which illustrates the significance of the issue. FarmVille is a farm simulation application. It allows users to manage virtual farms, and to interact with their friends when virtually farming. FarmVille encourages users to acquire as many farm neighbors as possible when advancing to the next level as incentives. However, those farm neighbors need to be Facebook friends of the users, which push them to eagerly seek more Facebook friends. Research reveals that a great portion of social game users, such as FarmVille, are likely to have more than 95% of their Facebook friendships established solely due to gaming purposes instead of being close friends in real life [13]. The SGT of FAITH allows users to transform their social graph such that different applications see different social graphs, and that each application sees the most optimal one. In FAITH, instead of befriending others in Facebook for any application-related purposes, users may create virtual friendships through FAITH. For instance, if FarmVille is under FAITH, users can establish virtual friendships with any other user or hide any real Facebook friendships for any gaming purposes.

IV. DESIGN & IMPLEMENTATION

A. *SDKs*

The architecture of FAITH requires two different types of SDKs: (1) the FAITH Client SDK and (2) the FAITH Server SDK. Both are derived from the official Facebook SDK.

1. *FAITH Client SDK*

The objective of the FAITH Client SDK is to allow applications, which include the SDK, to fetch social informatics from FAITH. To convert the official Facebook SDK into the FAITH Client SDK, there are three major areas requiring modifications.

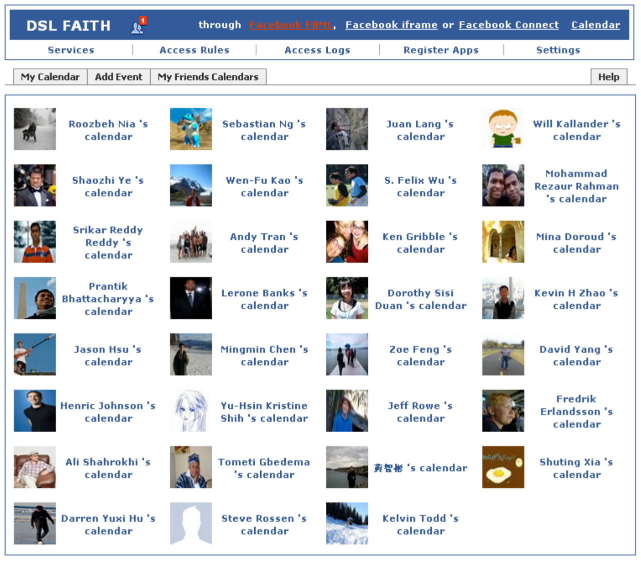
1) Delivering API call requests to FAITH instead of Facebook: From the application's perspective, FAITH takes the role of Facebook in providing social information. As such the SDK needs to send requests from the application to FAITH instead of Facebook.
2) Managing the data from FAITH: Since FAITH becomes the platform interacting with applications, extra variables are required in addition to the existing variables in the official SDK. These variables need to be initialized by data from FAITH. For instance, the encrypted Facebook session key is a new variable which needs to be initialized in new SDK. To prevent applications from contacting Facebook directly, FAITH encrypts the session key before sending it to applications.
3) Sending additional data to FAITH when making API calls: When FAITH receives API calls; it needs to send more data than an official SDK would for various purposes. The encrypted Facebook session key is an example of that. When FAITH receives this data, the session key can be decrypted to allow FAITH to authenticate with Facebook when fetching social informatics.

2. FAITH Server SDK

In comparison with the FAITH Client SDK, the FAITH Server SDK requires less modification of an official SDK. The main purpose is to allow (1) the UI of FAITH to fetch social informatics and (2) Facebook session key reuse. The first case requires no modifications to an official SDK as it already does that. On the contrary, session key reuse requires some changes to an official SDK. When each application initiates a Web API call, FAITH processes the request, and initiates another API call to fetch the social informatics from Facebook. However, without a valid Facebook session key, FAITH is unable to complete this. To solve this problem, the FAITH SDK needs to allow session key reuse. Whenever FAITH contacts an application, FAITH encrypts valid session keys received from Facebook, and attaches the encrypted keys to each request sent to the application. When the applications initiate API calls, the FAITH Client SDKs sends the encrypted keys back to FAITH. Encryption is mandatory since it prevents the applications from contacting with Facebook directly.

B. User Interface

The user interface built into FAITH is the primary method that users interact with the system and configure its many settings. As part of privacy control, FAITH allows users to control the usage of supported API methods. For instance, friends.getAppUsers is an Old REST API method to fetch the user IDs of the user's friends who have authorized the calling application. Figure 5 shows a screenshot of an application, Calendar, under FAITH. It is a social calendar application, which allows users to create daily events and share the events with their friends by allowing access to the calendars of their friends. Figure 5 is the page where users access the friends' calendars. After the user blocks friends.getAppUsers, Calendar is no longer able to retrieve the user IDs of the user's friends who have authorized Calendar. The page accessing the friends' calendars becomes as illustrated in Figure 6.


DSL FAITH
through Facebook FBML, Facebook iframe or Facebook Connect Calendar
Services | Access Rules | Access Logs | Register Apps | Settings
My Calendar | Add Event | My Friends Calendars | Help

Roozbeh Nia 's calendar
Sebastian Ng 's calendar
Juan Lang 's calendar
Will Kallander 's calendar
Shaozhi Ye 's calendar
Wen-Fu Kao 's calendar
S. Felix Wu 's calendar
Mohammad Rezaur Rahman 's calendar
Srikar Reddy Reddy 's calendar
Andy Tran 's calendar
Ken Gribble 's calendar
Mina Doroud 's calendar
Prantik Bhattacharyya 's calendar
Lerone Banks 's calendar
Dorothy Sisi Duan 's calendar
Kevin H Zhao 's calendar
Jason Hsu 's calendar
Mingmin Chen 's calendar
Zoe Feng 's calendar
David Yang 's calendar
Henric Johnson 's calendar
Yu-Hsin Kristine Shih 's calendar
Jeff Rowe 's calendar
Fredrik Erlandsson 's calendar
Ali Shahrokhi 's calendar
Tometi Gbedema 's calendar
黃智彬 's calendar
Shuting Xia 's calendar
Darren Yuxi Hu 's calendar
Steve Rossen 's calendar
Kelvin Todd 's calendar

Figure 5

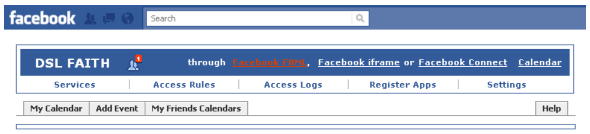
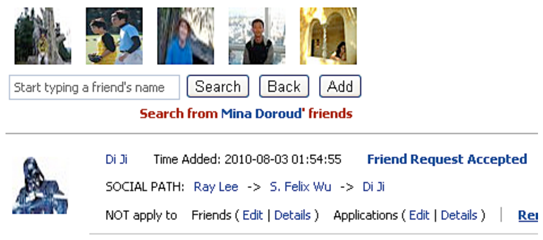

facebook
Search
DSL FAITH
through Facebook FBML, Facebook iframe or Facebook Connect Calendar
Services | Access Rules | Access Logs | Register Apps | Settings
My Calendar | Add Event | My Friends Calendars | Help

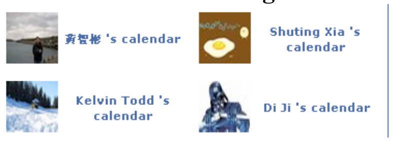
Figure 6

Figure 7 is a screenshot of the adding virtual friendships page. Before users can virtually befriend any other Facebook users through FAITH, they need to find a social path to each one of them. Figure 7 shows that Ray Lee has virtually befriended Di Ji, and the social path of the friendship is from Ray Lee (the user who initialized the friend request) to S. Felix Wu then to Di Ji. The process of friendship requests is similar to Facebook's. Users receive a notification for confirmation from FAITH when other users want to befriend the user. Figure 8 shows a screenshot of Calendar after Ray Lee and Di Ji has established a virtual friendship which has been confirmed by Di Ji. In comparison with Figure 5, Figure 8 shows that Calendar sees Di Ji as a friend of Ray Lee. Moreover, since Ray Lee is the one who initiated the virtual friendship, he has the privilege to exclude any users and applications from the virtual friendship.


Start typing a friend's name
Search
Back
Add
Search from Mina Doroud' friends

Di Ji Time Added: 2010-08-03 01:54:55 Friend Request Accepted
SOCIAL PATH: Ray Lee -> S. Felix Wu -> Di Ji
NOT apply to Friends (Edit | Details) Applications (Edit | Details) | Re

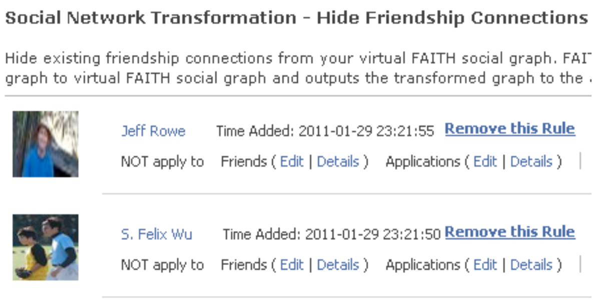
Figure 7



黃智彬 's calendar
Shuting Xia 's calendar
Kelvin Todd 's calendar
Di Ji 's calendar

Figure 8

Hiding real Facebook friendships works the opposite to adding virtual friendships except hiding does not need confirmation to be effective. When users choose to hide their real Facebook friendships, the hidden friendships becomes invisible to applications behind FAITH. Figure 9 is a screenshot of the hiding friendships page of FAITH. The figure shows that Ray Lee has hidden himself from Jeff Rowe and S. Felix Wu. Upon visiting Calendar again, Figure 10 shows that the application sees both Jeff and Felix as not friends of Ray Lee. Similar to adding, Ray Lee may exclude any of his Facebook friends and applications from the rules, and the excluded users and applications can continue to see the friendships when fetching the social informatics from FAITH.



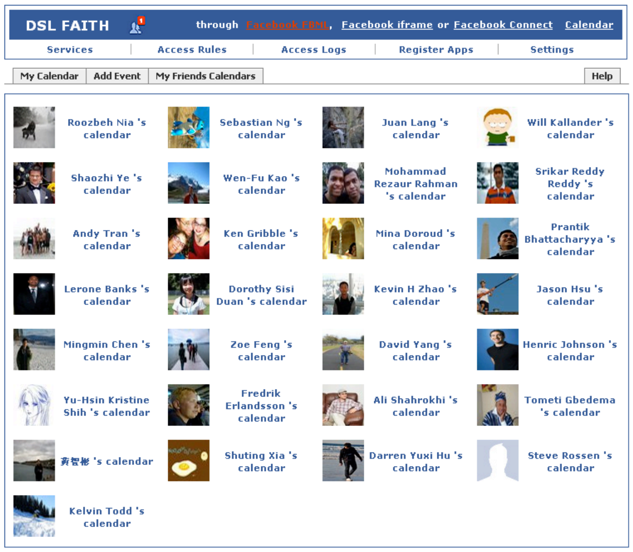
Social Network Transformation - Hide Friendship Connections

Hide existing friendship connections from your virtual FAITH social graph. FAI
graph to virtual FAITH social graph and outputs the transformed graph to the

Jeff Rowe Time Added: 2011-01-29 23:21:55 Remove this Rule
NOT apply to Friends (Edit | Details) Applications (Edit | Details) |

S. Felix Wu Time Added: 2011-01-29 23:21:50 Remove this Rule
NOT apply to Friends (Edit | Details) Applications (Edit | Details) |

Figure 9



DSL FAITH through Facebook FBML, Facebook iframe or Facebook Connect Calendar

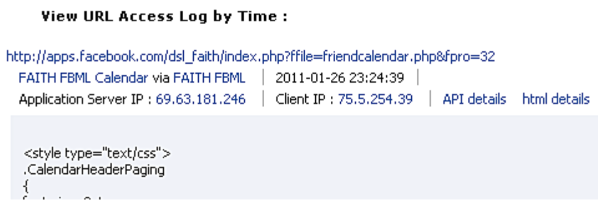
Services | Access Rules | Access Logs | Register Apps | Settings

My Calendar | Add Event | My Friends Calendars | Help

Roozbeh Nia 's calendar
Sebastian Ng 's calendar
Juan Lang 's calendar
Will Kallander 's calendar
Shaozhi Ye 's calendar
Wen-Fu Kao 's calendar
Mohammad Rezaur Rahman 's calendar
Srikar Reddy Reddy 's calendar
Andy Tran 's calendar
Ken Gribble 's calendar
Mina Doroud 's calendar
Prantik Bhattacharyya 's calendar
Lerone Banks 's calendar
Dorothy Sisi Duan 's calendar
Kevin H Zhao 's calendar
Jason Hsu 's calendar
Mingmin Chen 's calendar
Zoe Feng 's calendar
David Yang 's calendar
Henric Johnson 's calendar
Yu-Hsin Kristine Shih 's calendar
Fredrik Erlandsson 's calendar
Ali Shahrokhi 's calendar
Tometi Gbedema 's calendar
龚智彬 's calendar
Shuting Xia 's calendar
Darren Yuxi Hu 's calendar
Steve Rossen 's calendar
Kelvin Todd 's calendar

Figure 10

Access logs can be viewed by URL request. FAITH stores the HTML source codes of each page of applications they have previously visited and the API methods initiated by the applications during those visits. Figure 11 is a screenshot of the URL logging page of FAITH. The HTML source code shown in the light blue square is the code shown to the user when visiting the page. Figure 12 shows the Old REST API initiated by the application in the page. It shows that Calendar made an API call, friends.getAppUsers, at 2011-01-16, and access was allowed. If the user has blocked the API method, FAITH would still capture and log the call but it would return an empty list to the application and display "Access Denied" in the log instead. With FAITH, users know exactly what social informatics is fetched and utilized by each application for each page they visit.

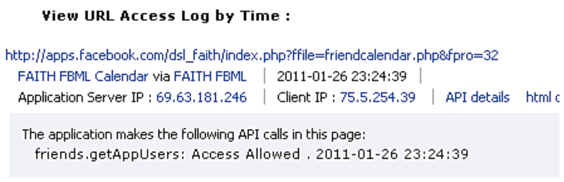


View URL Access Log by Time :

```
http://apps.facebook.com/dsl_faith/index.php?ffile=friendcalendar.php&fpro=32
  FAITH FBML Calendar via FAITH FBML  |  2011-01-26 23:24:39  |
  Application Server IP : 69.63.181.246  |  Client IP : 75.5.254.39  |  API details  html details

<style type="text/css">
.CalendarHeaderPaging
{
```

Figure 11



View URL Access Log by Time :

```
http://apps.facebook.com/dsl_faith/index.php?ffile=friendcalendar.php&fpro=32
  FAITH FBML Calendar via FAITH FBML  |  2011-01-26 23:24:39  |
  Application Server IP : 69.63.181.246  |  Client IP : 75.5.254.39  |  API details  html c

The application makes the following API calls in this page:
  friends.getAppUsers: Access Allowed . 2011-01-26 23:24:39
```

Figure 12

Access logs can be viewed by API call. API logs capture the social informatics returned in response of the API calls. Each API log includes the name of the API method, application, time, access status (allowed or blocked), the IP addresses of both application server and client as well as the social informatics returned to the application.

C. FAITH Server

The job of the FAITH Server is to respond to API calls initiated by applications and deliver the transformed social informatics back to applications. It is also the place where social informatics logging, privacy control and social graph transformations are implemented. To provide a deeper understanding, we detail the design and implementation of the component as follows. When an API call request arrives at FAITH Server, it first examines both HTTP POST variables and HTTP GET variables from the application. If valid, FAITH Server gets the encrypted Facebook session key from the data, and decrypts the key to fetch the social informatics from Facebook. Before passing the data back to the calling application, FAITH Server logs the social informatics if API logging is enabled. The next step of the process is to implement privacy controls. FAITH Server checks if the user allows the API call. If not, it returns an empty list to the application. Otherwise, continue to the last step of the process, SGT. FAITH Server transforms the social informatics as specified by the user and passes the transformed data back to the calling application. The implementation of SGT involves modifications over the results of those API methods according to user settings before delivering the social informatics back to applications. For instance, assume that Alice and Bob are not Facebook friends, but they have established a virtual friendship through FAITH. If Alice visits an application, it initiates a friends.get API call to request the user IDs of Bob's Facebook friends. In response to the call, FAITH generates another request to fetch Bob's friends from Facebook. When FAITH receives the list of Bob's real Facebook friends, Alice certainly is not in the list since they are not real Facebook friends. FAITH needs to transform the list. It is now that Alice is added to the list.

D. Integration with FAITH

To integrate with the Facebook Platform, each application first needs to register with Facebook to get a unique ID and secret this is used in the SDK to fetch the social informatics from Facebook. Integration with FAITH works very similarly. Each application needs to register with FAITH as well. New applications can utilize a FAITH Client SDK during development of the application while existing applications only need to replace official Facebook SDKs with FAITH Client SDKs.

E. EVALUATION

Although FAITH has reached approximately two hundred active users, we hope to evaluate the usability of the system when it is utilized more widely. In this section, we instead evaluate the performance and overhead of the system. In our experiments, we measured the processing time of two web pages of identical source code with one in a FAITH application and the other in a Facebook application. The same experiment was conducted for both FBML and IFrame, and averages are calculated based on 20 occurrences. When both pages of FBML applications initiate a API method, which fetches twenty friends of current active users, the average processing times are 0.062126 seconds (Facebook app) and 0.074926 seconds (FAITH app). The overhead of FAITH causes a 20% increase in processing time. When both pages initiate the same methods 10 times, the average processing times are 0.564256 seconds (Facebook app) and 0.680208 seconds (FAITH app), also a 20% increase. For IFrame applications, when the same method is initiated once, the average process times are 0.084166 seconds (Facebook app) and 0.087507 seconds (FAITH app), a 4% increase. For the case of calling the method 10 times, the average processing times are 0.702290 seconds (Facebook app) and 0.825028 seconds (FAITH app), a 17% of increase.

F. Issues

1. Existing Applications NOT Using SDKs

An SDK is not the only way for applications to fetch the social informatics from Facebook. The Facebook Platform also supports OAuth 2.0 protocol [4], which utilizes access tokens for authentication. Applications are able to utilize access tokens to retrieve the social data through URLs without SDKs. Since direct retrieval from Facebook is not permitted, FAITH encrypts access tokens before passing to applications. As a result, applications utilize access tokens need to modify the source codes to use SDKs only to integrate with FAITH.

2. FAITH SDKs in Languages Other Than PHP

Currently, FAITH server SDKs and FAITH client SDKs are all in PHP for it is the most prevalent server language of Facebook applications. While we are in the process of developing SDKs in other languages, the system can only integrate with applications in PHP at this time.

V. CONCLUSION & FUTURE WORK

In this paper, we presented the design and implementation of FAITH (available -- http://apps.facebook.com/dsl_faith/). Here, applications are regarded as processes and Facebook social informatics as resources similar to an operating system. In that respect, FAITH functions as an application-level social-centric operating system kernel. FAITH augments the Facebook platform’s management mechanisms of social data defending against privacy leaks and devaluation of social informatics. This paper has described how FAITH can be utilized to accomplish this. From the feedbacks we received, many users found FAITH useful in defending the privacy of their social data. We have also received many feedbacks about increasing the number of applications integrated with FAITH, which is certainly our goal. While we have received no complaints regarding performance during development, performance testing has demonstrated acceptable delays in processing time. In the future, we hope to further decrease these delays by utilizing data caching. Because making an API call is time consuming and cumbersome, caching the results of frequently-used API methods is an appropriate technique to improve efficiency.

ACKNOWLEDGEMENT

This research was supported in part by NSF CNS-0832202, BBN-GENI, Army Research Lab (Network Science CTA), ARO MURI (Arsenal), AFOST MURI (Helix), and Intel.

REFERENCES

[1] Top 25 Facebook Games for Feburary 2010 http://www.insidefacebook.com/2010/03/02/top-25-facebook-games-for-february-2010/

[2] Facebook Press Room Statistics http://www.facebook.com/press/info.php?statistics

[3] Facebook Developers http://developers.facebook.com/docs

[4] The OAuth 2.0 Protoccol Framework http://tools.ietf.org/html/draft-ietf-oauth-v2-11

[5] Facebook and Privacy Issues http://articles.sfgate.com/2010-10-19/opinion/24141616_1_mafia-wars-facebook-privacy-rules

[6] Huang, Y, Supporting Meaningful Social Networks. Retrieved Feburary, 2011, from http://eprints.ecs.soton.ac.uk/17180/2/thesis.pdf

[7] Performance http://developers.facebook.com/docs/guides/performance

[8] Haifeng Zhao, Shaozhi Ye, Prantik Bhattacharyya, Ken Gribble, Jeff Rowe, S. Felix Wu, SocialWiki: Bring Order to Wiki Systems with Social Context. In SocInfo '10: Proceedings of the 2nd IEEE International Conference on Social Informatics, Laxenburg, Austria.

[9] Thomas Tran, Jeff Rowe, S. Felix Wu, Social Email: A Framework and Application for More Socially-Aware Communications. In SocInfo '10: Proceedings of the 2nd IEEE International Conference on Social Informatics, Laxenburg, Austria

[10] Web Service http://en.wikipedia.org/wiki/Web_service

[11] Matt Spear, Xiaoming Lu, S. Felix Wu, Davis Social Links or: How I Learned To Stop Worrying And Love The Net. In SCA '09: Proceedings of the International Symposium on Social Computing Applications, held in conjunction with IEEE SocialCom, Vancouver, Canada, August, 2009.

[12] Lerone D. Banks and S. Felix Wu, Toward a Behavioral Approach to Privacy for Online Social Networks. Proceedings of the Second international conference on Social informatics. 2010.

[13] Nazir, A., Raza, S., and Chuah, C. 2008. Unveiling facebook: a measurement study of social network based applications. In Proceedings of the 8th ACM SIGCOMM Conference on internet Measurement (Vouliagmeni, Greece, October 20 - 22, 2008). IMC '08. ACM, New York, NY, 43-56