Role of Apps in Undoing of Privacy Policies on Facebook

Submitted in partial fulfillment of the requirements of the degree of

Master of Technology (M.Tech)

by

Nivia Jatain

Roll no. **15305R007**

Supervisor:

Prof. R.K. Shyamasundar

Department of Computer Science & Engineering

Indian Institute of Technology Bombay

2018

Dissertation Approval

This project report entitled **"Role of Apps in Undoing of Privacy Policies on Facebook"**, submitted by **Nivia Jatain** (Roll No. **15305R007**), is approved for the award of degree of **Master of Technology (M.Tech)** in Computer Science & Engineering.

Prof. R.K Shyamasundar
Dept. of CSE, IIT Bombay
Supervisor

.....	**......**
Dept. of CSE, IIT Bombay	Dept. of CSE, IIT Bombay
External and Internal Examiner	Internal Examiner

Date: ...... June 2018

Place: ____________________

Declaration of Authorship

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature: ....

Nivia Jatain

15305R007

Date: ...... June 2018

Abstract

Facebook is a social networking site that connects people around the world. Web based social networks (WBSNs) like Facebook are online communities that allows users to publish resources and to establish relationships of different types(“friends”, “family” etc) with other users in the network. But the question is Is this sharing of resources Secure? In this report, we will analyze how far the security and privacy policies of Facebook are preserved and kind of possible breaches that could occur. Also, we will analyze how granting various permissions to an app can lead to information leakage contrary to the privacy settings or policies specified by the user. We have presented various scenarios where granting permissions to an app results in violation of user privacy policies. All the experiments are being carried out on Facebook Developer Platform. We have also investigated Facebook platform for access to user’s data by Advertisers.

Contents

List of Figures

Chapter 1

Introduction

Facebook like large social networks helps individuals and organisations to establish digital identities and interact with each other. The social network systems like Facebook are characterized using three functions[4]:

1. Identity representation: allows users to create a profile and to provide personal information.

2. Distributed relationship articulation : organizing relationships in diffrent categories like Friends, Family, Acquaintances etc. which helps in specifying access control in a proper way.

3. Traversal-driven access : allow users to traverse social graph and access is granted to the resource based on access policy of the resource owner.

The organization of relationships in categories like "Friends", "Family" etc helps in building relevant audiences whenever there is an update from a user. Also, users have control of who can see what updates. All the information of the users and their interactions on Facebook is organized as a graph called as social graph. Every user of Facebook is represented as a node on social graph and its relationship with other nodes is determined through labelled edges. The Graph API is the primary way to get the data out of, or put the data into, Facebook's platform i.e. we use Graph API to query the social graph of Facebook.
Example query:**GET graph.facebook.com/me?fields=id,name,picture & access_token**

id represents the id of the node to be queried. Only root nodes can be queried in social graph. It is dynamic in nature meaning its state continuously changes reflecting user's actions. The social graph is consists of [1]:

- nodes - basically "things" such as a User, a Photo, a Page, a Comment
- edges - the connections between those "things", such as a Page's Photos, or a Photo's Comments
- fields - info about those "things", such as a person's birthday, or the name of a Page

Facebook allows developers to create apps on their developers platform where each app is given a unique id with which it is represented on social graph and as users install or authorize these apps, an edge is created on social graph between that user node and app node. The profiles of the users can be tracked on interacting with these applications. Now these apps after being authorized will have access to user's information and their interactions depending on the permissions granted to the app. Thus, apps in a way helps the Facebook platform to segment users in specific categories which can be used in profiling the users which further helps in building accurate audiences for the Advertisers. The Figure 1.1 shows access hierarchy in the social graph. The access control mechanism to control access to user's information is different for each layer(user, app and advertisement layer). In this report, we will show there is no coherence in policy enforcement across the layers, which undermines the privacy of the users[8]. So there is a need of regulations on privacy while tracking and to check the relationship between privacy policies and the information collected through apps (through experimentation) via Facebook. All the findings have been validated via experiments on Facebook's developers platform and using the Graph API version 3.0.

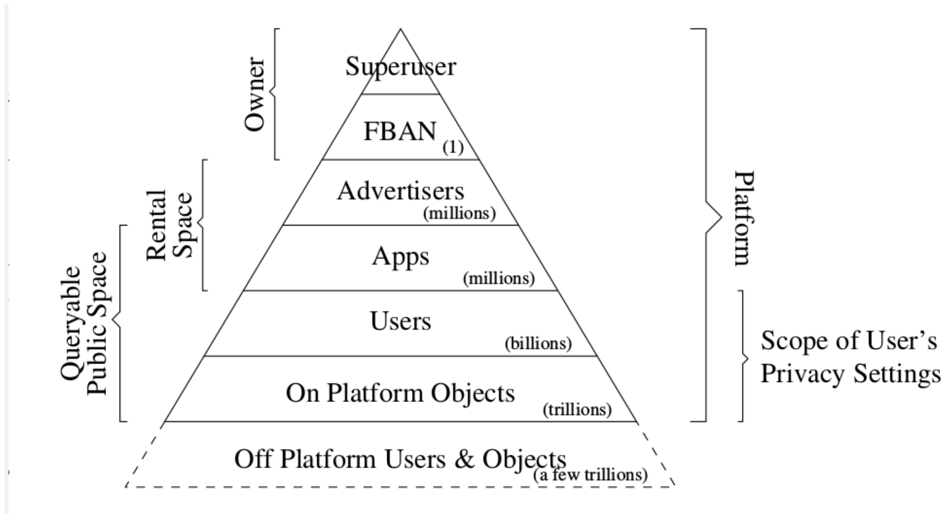


Figure 1.1: Access-hierarchy in the social graph[8]

The remainder of the report is organized as follows: Chapter 2 gives an overview of how access control works in Facebook. Chapter 3 discusses various privacy breach scenarios fro user's point of view. Chapter 4 provides an introduction to access tokens and discusses about various permissions that can be granted to an application. Chapter 5 presents the various results obtained via experimentation on Facebook developer platform. Chapter 6 discusses related work and Chapter 7 concludes the report.

Chapter 2

Access Control in Facebook

The content of Facebook is organized as follows:

- Newsfeed : gives updates to users about their friends and the people they are connected to
- Timeline : where users own content is being organized
- Graph Search : Social graph captures all the activities of Facebook users and their relationships and allows users to query the graph

Social graph - Reachability as condition for access

Social graph in Facebook represents information present on the Facebook. It is dynamic in nature meaning its state continuously changes reflecting user's actions. Every user of Facebook is represented as a node on social graph and its relationship with other nodes is determined through labelled edges. For example if two users are friends on Facebook then their relationship on social graph is represented as "Friends" edge. Therefore, edges between the nodes establishes connections between them and helps in extending their reachability in social graph. For example one user can access posts(having privacy settings as "Friends") of another user if they are connected via friends edge and if the user likes on that post then this will be reflected in social graph via likes edge between the user and the post. All such type of interactions, like users interacting with each other or user interacting with an app) are recorded in the social graph and the graph gets updated every time there is addition or deletion of a node or edge depending on user's and app's interactions with the nodes reachable to them.

List as access policies for users

Users on Facebook can specify access control policies on objects in following ways:

- Only Me : This is the label/policy where the user himself is the only member.
- Public : This is label/policy for the object that can be accessed publicly
- Friends : This is the primary list where all the friends of a user are listed
- Restricted : This is the list of friends who can access only publicly available information
- Friend of Friends : This list consists of users who have Friends relation with every member in the Friends list.

Apart from the above access control policies, a user can also specify his specify his own custom policy i.e. a user can select specific friends/make list of some specific friends(for example Close Friends) with whom he wants to share the information/object.

Capabilities as access policies for Apps

Apps are also represented as nodes on social graph and the reachability of apps in social graph depends on the permissions granted to an app. For example an app can't access posts of a user unless granted user_posts permission. There are 32 such permissions. We will discuss about few of these permissions in greater detail in later sections of this report and will also see which of these permissions results in violations of privacy policies of users.

The capabilities of the social graph are not just limited to objects representation but it also provides real-time updates to the users depending on their interactions in the social graph and this is handled by Facebook's NewsFeed algorithm and interactions of users with apps helps in attaining precision.

Chapter 3

Analysis of privacy preservation in Facebook through user specified policies

In this section we will present various scenarios where user's information is leaked contrary to the privacy settings specified by the user. Assumptions are as follows : users = {A, B, C, D, E}
friendship edges = {(A,B), (B,C), (A,D), (C,E), (D,E)}
Family(B) = {A}
University = {A, C, E}
School = {A, E}
University and School are Social lists.

- **Nonrestrictive change in policy of an object risks privacy of others:** Consider the scenario presented in the Figure 3.1, Nonrestrictive change in policy from family to friends on P2 will be enforced on all the dependent nodes of P2(comments, replies, likes) and User A's comment is exposed to B's friends without the consent of A.

- **Restrictive change in policy of an object suspends other's privileges** Consider the scenario presented in Figure 3.2, Restrictive change in policy from Public to Only me on P5 locks out the users(D in Figure) from updating their own

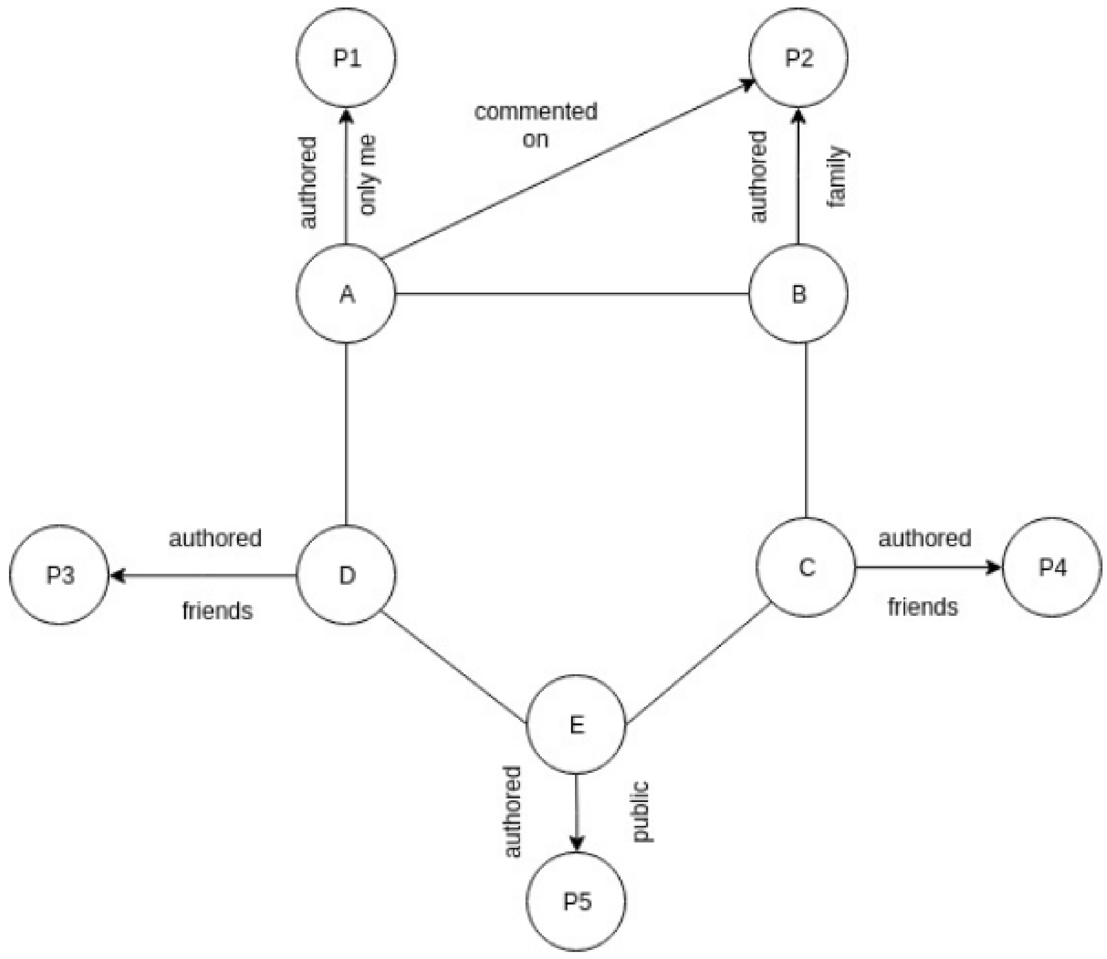

P1
P2
commented on
authored
only me
authored
family
A
B
P3
authored
friends
D
C
authored
friends
P4
E
authored
public
P5

Figure 3.1

comments and likes.

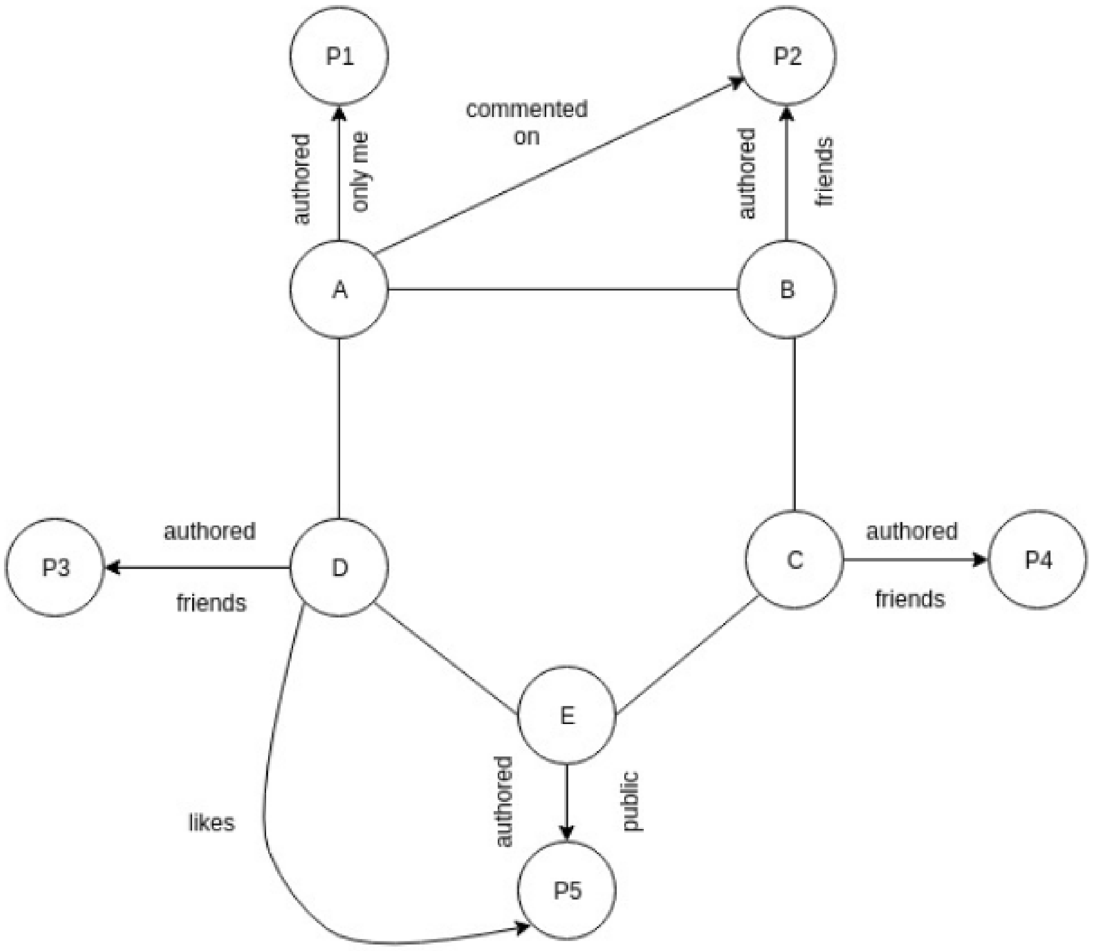


Figure 3.2

- **Like, Comment operations are not privacy preserving:**
  When a user likes or comments on an object whose visibility is set to public then the interactions of the user with that object also becomes public and social graph allows queries to public content. For example if a query like photos liked by Alice is given then this query returns all the photos liked by Alice. Similar is the case with comments. Also any user of the Facebook can make queries about any other user of the Facebook.

In Chapter 5 we will look at possible scenarios of how information leakage can happen when an app comes into the picture but first lets discuss about access tokens and some of the permissions which these apps require in order to access user's data.

Chapter 4

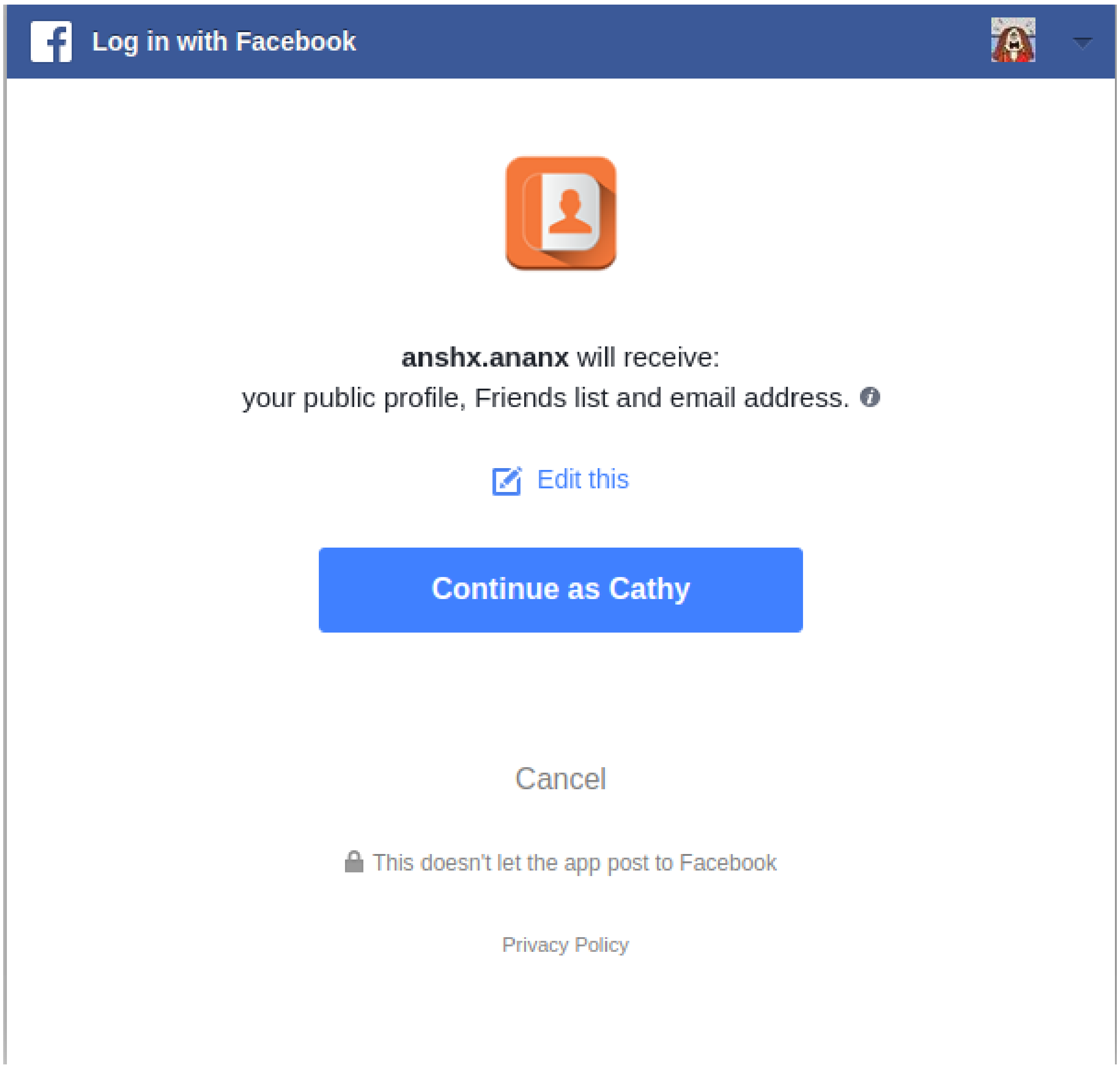
Facebook Login

Facebook login for apps is a way via which people can log into our app across various platforms as shown in Figure 4.1. When a user logs into an app via Facebook login then at the time of login an app can ask for various permissions.An app can have a total of 32 permissions. Out of these the app have two permissions by default email and public_profile. All the other permissions except these will require a review from facebook. In the following Section 4.2, we will discuss in detail about permissions that we have experimented on.

4.1 Access Tokens

An access token is an opaque string which is used for identifying a user, an application, and a page. Applications use access tokens when they have to make graph API calls. The access token contains information regarding token expiration time and the app which generated the token. Access tokens are required while making calls to the graph API. There are different types of access tokens:

- **User Access Token:** This access token is required when an app makes Graph API calls on behalf of a particular user in order to read, modify or write that user's Facebook data. This token is obtained through the login dialog and requires the user to grant necessary permissions to the app. **The permissions used to set the access token will be the only permissions that can be accessed. Altering**

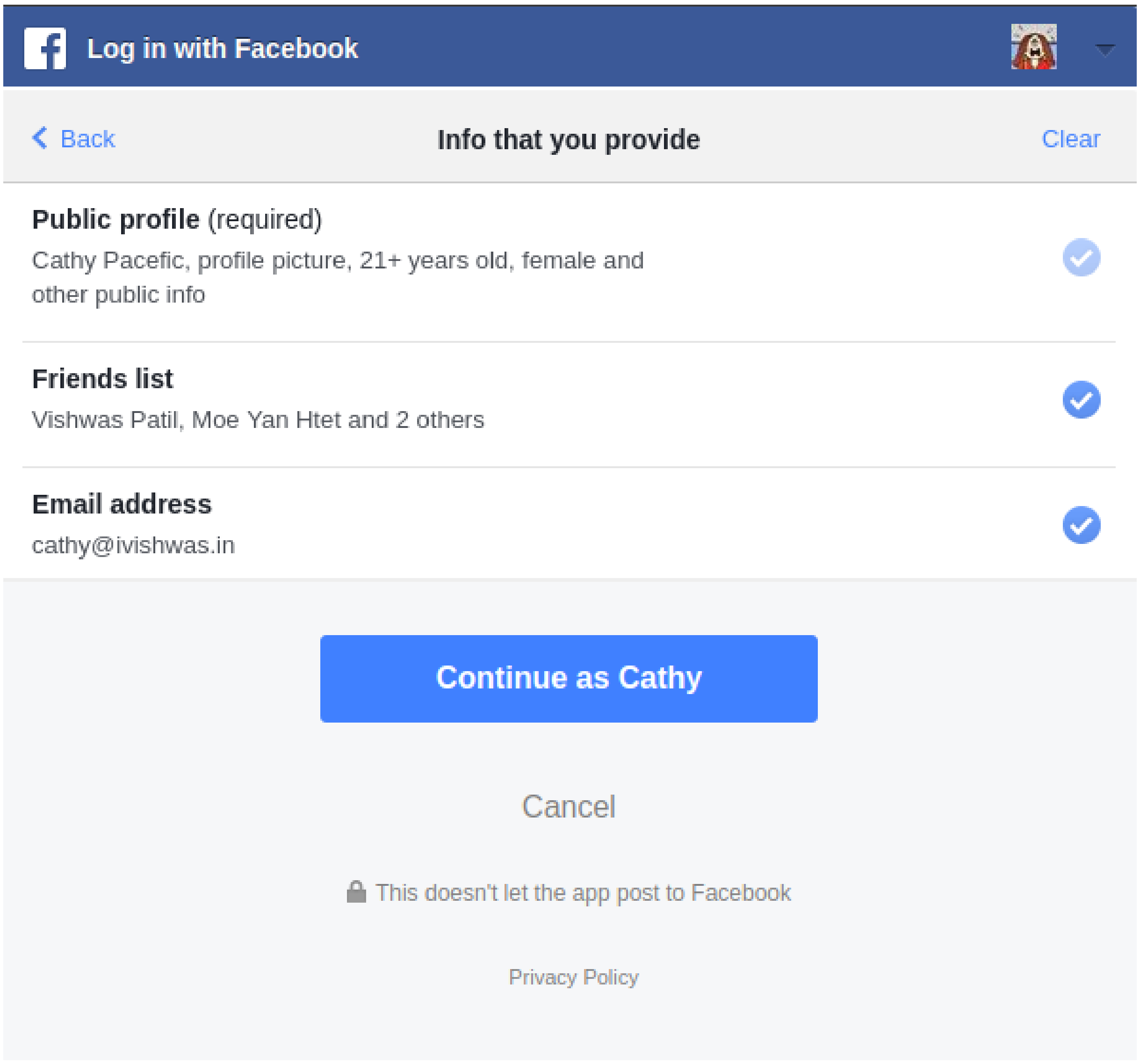


Log in with Facebook

anshx.ananx will receive:
your public profile, Friends list and email address.

This doesn't let the app post to Facebook

Figure 4.1: Login dialog asking for default permissions


Log in with Facebook

Back
Info that you provide
Clear

Public profile (required)
Cathy Pacefic, profile picture, 21+ years old, female and other public info

Friends list
Vishwas Patil, Moe Yan Htet and 2 others

Email address
cathy@ivishwas.in

Continue as Cathy

Cancel

This doesn't let the app post to Facebook

Privacy Policy

Figure 4.2: Login dialog showing editing of permissions

the set of permissions alters the access token. The working of Access Tokens is shown in Figure 4.3.

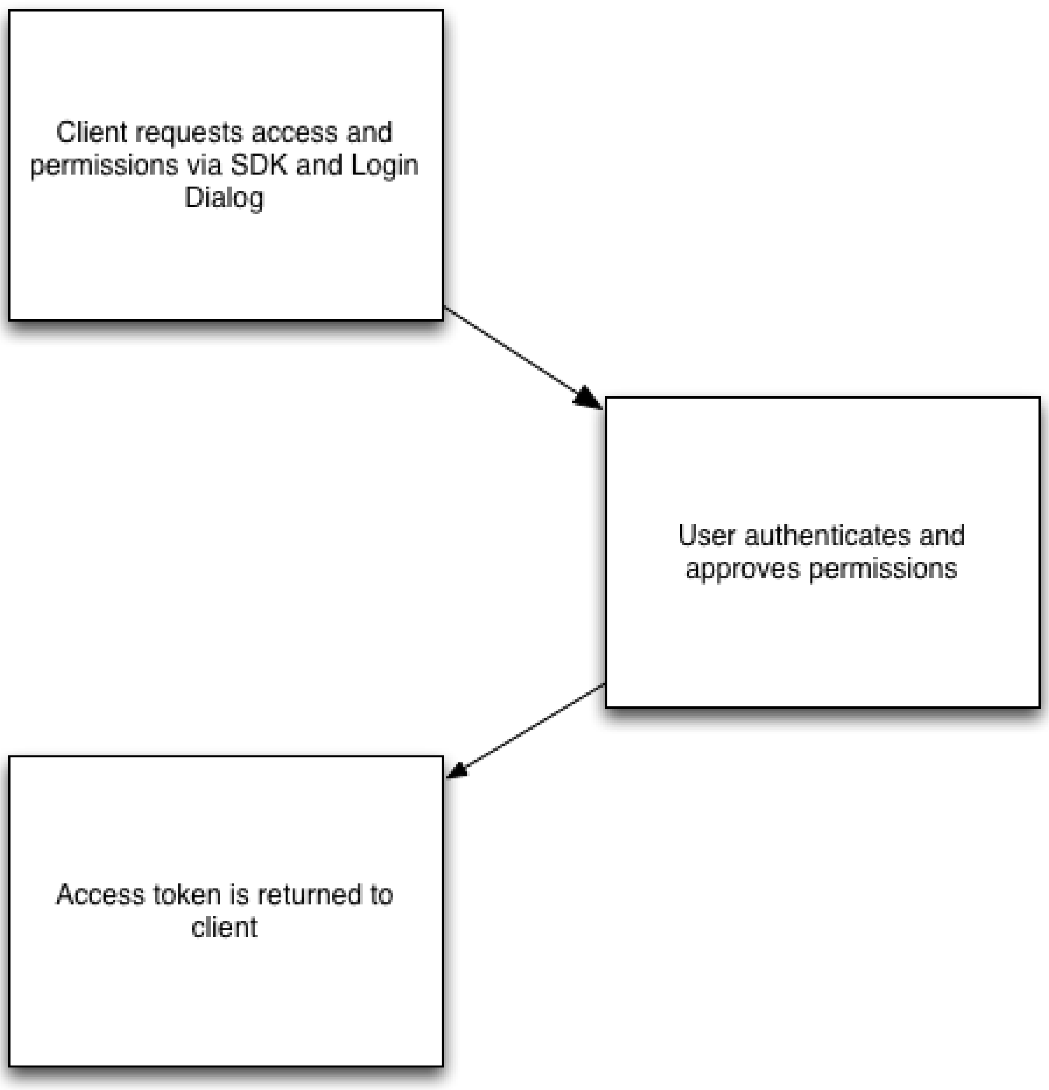
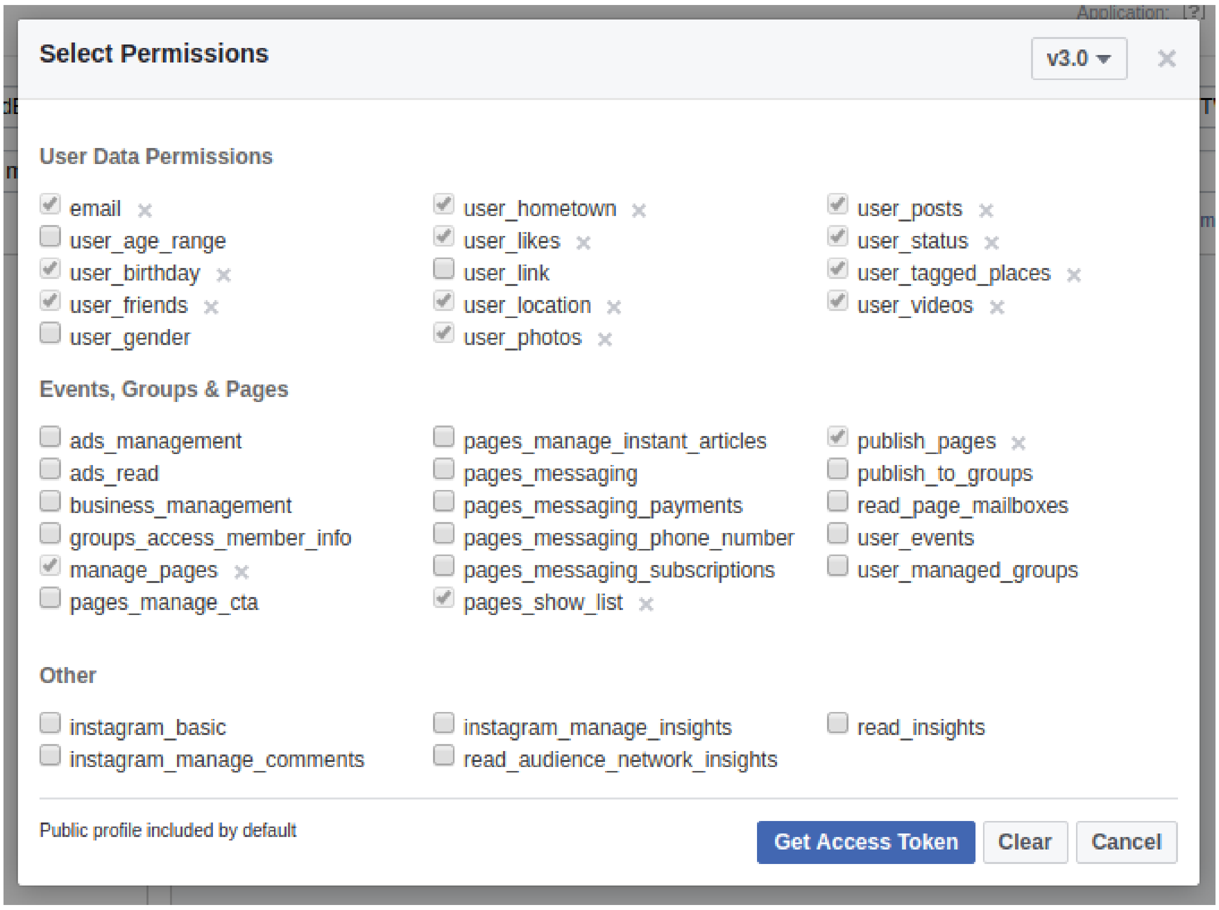

Client requests access and permissions via SDK and Login Dialog
User authenticates and approves permissions
Access token is returned to client

Figure 4.3: How Access Tokens Works[1]

- **App Access Token:** In order to modify and read an app's settings this access token is required.
- **Page Access Token:** This access token is required when API calls are made in order to read, write or modify data of a Facebook page. In order to obtain a page access token, there is a need to obtain user access token first and then asking for manage_pages permission.

4.2 Permissions

In this section we will discuss in detail about few permissions which we have used while performing experiments. The following Figure 4.4 shows all the permissions available to an App. The permissions are as follows:



Select Permissions v3.0

User Data Permissions

[✓ email]	[✓ user_hometown]	[✓ user_posts]
user_age_range	[✓ user_likes]	[✓ user_status]
[✓ user_birthday]	user_link	[✓ user_tagged_places]
[✓ user_friends]	[✓ user_location]	[✓ user_videos]
user_gender	[✓ user_photos]	

Events, Groups & Pages

ads_management	pages_manage_instant_articles	[✓ publish_pages]
ads_read	pages_messaging	publish_to_groups
business_management	pages_messaging_payments	read_page_mailboxes
groups_access_member_info	pages_messaging_phone_number	user_events
[✓ manage_pages]	pages_messaging_subscriptions	user_managed_groups
pages_manage_cta	[✓ pages_show_list]	

Other

instagram_basic	instagram_manage_insights	read_insights
instagram_manage_comments	read_audience_network_insights	

Public profile included by default

Figure 4.4: Set of Permissions

1. **public_profile:** When a user grants this permission then an app can have access to the user's data which is part of user's public profile. By default public_profile includes id, name, first_name, last_name, middle_name, short_name, picture.

2. **user_friends:**This permission gives access to list of friends who also use your app i.e. friends who has authorized the app. This means all the friends in the list must have granted user_friends permission to the app. These friends can be found on friends edge on the user object.

3. **user_birthday:**This permission provides access to date and month of a user's birthday which may or may not include year of birth depending on privacy settings of a person and the access token used to query the birthday field.
4. **user_events:**This permission provides read only access to user's events.
5. **user_hometown:**This permission provides access to person's hometown location using hometown field on user object.
6. **user_likes:**This permission provides access to all the Facebook pages and objects that a user has liked.
7. **user_location:**This permission provides access to the current city of the user using the location field of the user object.
8. **user_photos:**This permission provides access to all the photos that a user has uploaded or tagged into.
9. **user_posts:**This will provide access to posts made by the user and the posts in which a user is tagged.
10. **user_relationships:**This will provide access to user's relationship status.
11. **read_custom_friendlists:**This will provide names of the custom friend lists created by a user. For example family, close friends. It is deprecated now.
12. **publish_to_groups:**An app can publish to a group using this permission.

publish_actions has been deprecated in Facebook Graph API version 3.0. This permission allowed an app to post on a user's timeline.

Chapter 5

Experiments and Results : How granting permissions to an App violates user's privacy policy?

5.1 System setup

To conduct the experiments Apps are created on Facebook developers platform and have implemented facebook login which allows the users to login to the app. The login permissions which are approved by default are email, and public_profile. In order to carry out experiments test users(which are temporary facebook accounts created to test various features of the app) are created on Facebook developers platform and obtained information corresponding to the permissions for these users. The facebook graph api v3. 0 is used. In order to collect any kind of information for a user an app first obtains an access token for that user(this is obtained when a user logs into the app via Facebook Login) and then queries the graph API on behalf of that user using the information provided in the access token.

5.2 Social Graph in consideration

The following graph Figure 5.1 has been constructed using test users for App1(TestApp) and App2 (anshx.ananx) : The Figure 5.1 represents the social graph considered in

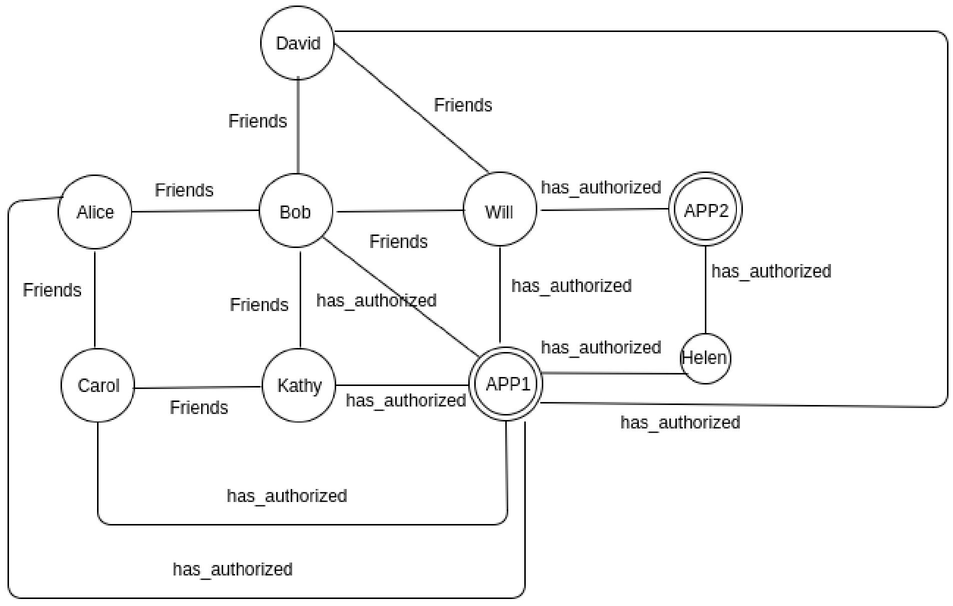


Figure 5.1: Social Graph in Consideration

carrying out the experiments. The friends edge depicts that two users are friends and has_authorized/installed edge represents that a user has installed the app and granted it various permissions. The following table represents the test users and permissions granted to them.

Permissions are granted as follows:

Alice : user_likes, user_photos

Carol : user_birthday, user_location, user_hometown, user_posts

Kathy : publish_actions, user_relationships, user_friends, user_managed_groups

Bob : email, user_friends

Will : user_posts, publish_actions

David : user_posts, publish_actions

Betty:user_posts,user_managed_groups(deprecated),user_posts,group_access_member_info,read_group

Helen : publish_actions, user_posts

Test users associated with both the apps(TestApp and anshx.ananx) : Will Will, Helen

Assumptions

1. All the queries will be from app side(TestApp) unless otherwise stated.
2. Facebook API version used to carry out the experiments : 2.12, 3.0
3. Javascript and php SDK is used to make Facebook API calls.
4. For simplicity we will consider the subgraph of the above social graph in Figure 5.1 in order to demonstrate each individual experiment.

5.3 Experiments Conducted and Results Obtained

Experiment 1: Retrieving the posts shared by the user

Consider the Figure 5.2 which is subgraph of social graph shown in Figure 5.1

- David authorizes App1 and grants permission user_posts.
- David shares the post P1 made by Will.
- Facebook API call used by App1 to retrieve user's(here David) posts is:

-
  ```
  FB.api( "/me/feed",function (response) {
        if (response && !response.error) {
          /* handle the result */
               }
      }
  );
  ```

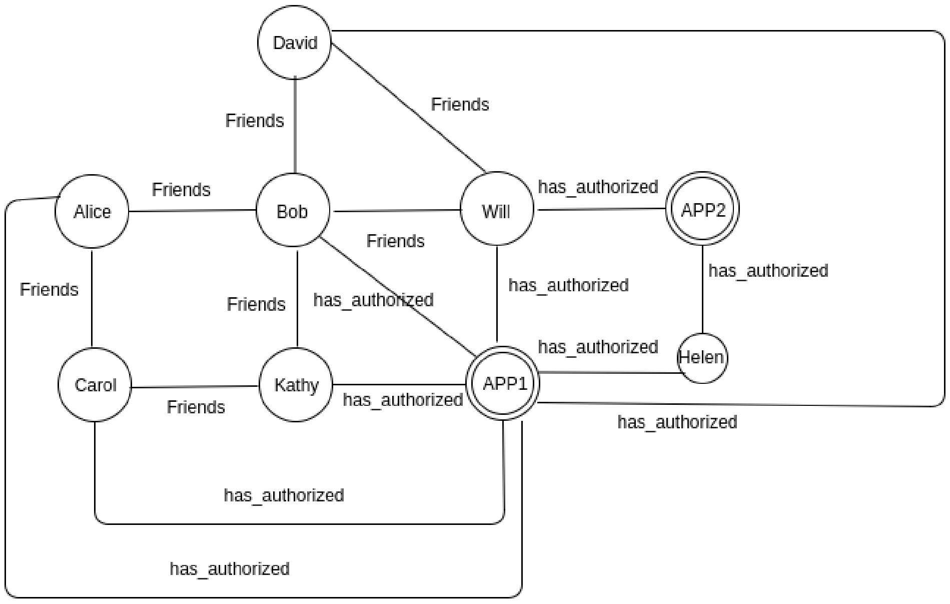


Figure 5.2

- Above API call will return all the posts on user's(David) timeline including the post shared by the user.

```
created_time:"2018-04-09T20:36:41+0000"
id:"113415586159959\_150524329115751"
story:"David David shared Will Will's post."
```

- Result here will contain created time, id and the story of the post.The story field in the result will give the information that the post is a shared post.
- We won't be able to retrieve comments on the shared post as the post id for the user who has shared the post will be different.

Experiment 2: Given post id, getting the details of the application which published the post on user's timeline : App can find out other Apps installed by the user

Consider the Figure 5.1

- Permission required : user_posts
- User Helen has authorized both the apps,App1 with permission user_posts and App2 with permissions publish_actions, user_posts.
- App2 publishes a post on Helen's timeline, App1(TestApp) can now use the following API call to retrieve the details of the application which has published the post.

```
FB.api( "/{post-id}?fields=application,created_time,message",
    function (response) {
      if (response && !response.error) {
        /* handle the result */
              }
    }
);
```

- application field here gives the details of the application which has published the post and that includes category, id, link of the app, name and namespace.
- created_time provides the time of creation of the post and message field represents the status message in the post.
- Result of the API call when App2 queries about App1 is:

```
    application:
category:"Education"
id:"335320743592541"
link:"https://apps.facebook.com/anshxananx/"
name:"anshx.ananx"
namespace:"anshxananx"
__proto__:Object
created_time:"2018-04-11T13:07:29+0000"
id:"108348820021337_108175360038683"
message:"Post by Anshx"
```

- So, with this result App1 can come to know about which other apps(here App2) the user Helen has installed and vice versa.
- publish_actions permission has been deprecated for version 3.0 whereas the above results are valid for Facebook Graph API version 2.12.

Experiment 3: Retrieving daily and monthly active users for an app, given an app id:

Consider the Figure 5.1

- We can retrieve daily and monthly active users of an app given its app id.
- Also, the user Will(Figure 5.1) here has authorized both the apps the one which is querying(App1:TestApp) as well as the one which is being queried(App2:anshx.ananx).
- Facebook API call used is:

```
FB.api("/{app-id}?fields=app_name,link,daily_active_users,category,monthly_act
    function (response) {
      if (response && !response.error) {
        /* handle the result */
        console.log(response);
      }
    }
);
```

- The result is as follows:

```
category:"Education"
daily_active_users:"0"
id:"335320743592541"
link:"https://apps.facebook.com/anshxananx/"
monthly_active_users:"21"
```

Experiment 4: Given a post id, retrieving comments on a post

- As shown in the Figure 5.3 below, David makes a post P1 with privacy as "friends"

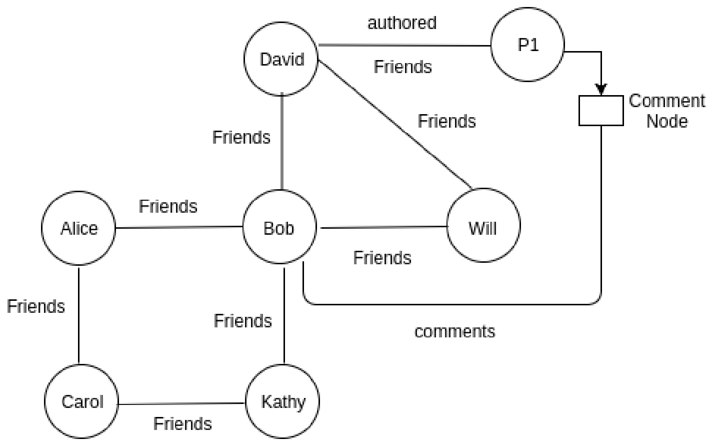


Figure 5.3

- Bob comments on P1 with comment "Test Comment by Bob"
- David logs into App 1 and the app can use the below Facebook API call to retrieve comments on a post:

```
FB.api( "/{post-id}/comments",
    function (response) {
      if (response && !response.error) {
        /* handle the result */
}});
```

- Result of above API call is as follows:

```
created_time:"2018-04-12T10:36:14+0000"
from:
id:"104812420348221"
```

```
name:"Bob Greenewitz"
id:"152355935599257_152356518932532"
message:"Test comment by Bob"
```

- from field here gives the name and id of the user who has commented on the post and message field gives message in the comment.
- Conclusion : The comments on a post can be retrieved irrespective of the privacy policy of the post(only me,friends or anything else). An app can only update the post which is published by it. An app can't make any kind of changes(other than reading) to the posts which are not published by it.

1. An app cannot publish comments on behalf of a user.
2. There is no API call to share a post for an app.
3. An app can publish comments on a page.

Experiment 5: Given a post id, retrieving privacy settings of the post

Scenario 1: When the user makes a post with only me privacy setting

- User David makes a post P1 with privacy settings as only me as shown in the Figure 5.4 below
- App 1 can use the following API call to retrieve the privacy settings of P1:

```
FB.api( "/{post-id}?fields=privacy,created_time,message",
    function (response) {
      if (response && !response.error) {
        /* handle the result */
        console.log(response);
```

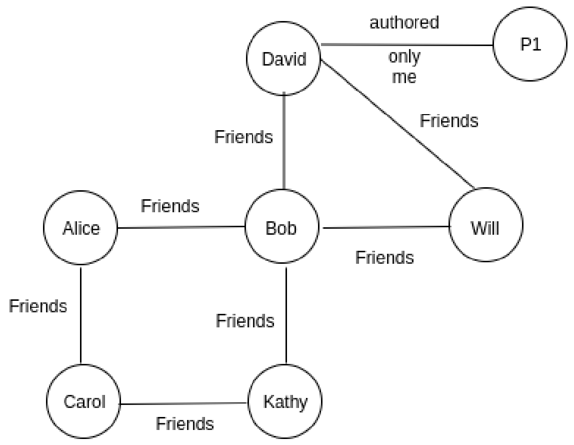


Figure 5.4

```
        }
      }
  );
```

- Below is the result of the above API call :

```
created_time:"2018-04-07T07:14:44+0000"
id:"113415586159959_148447715990079"
message:"Modified Post"
privacy:
allow:""
deny:""
description:"Only me"
friends:""
value:"SELF"
```

Scenario 2: Custom policy as privacy setting

- Family and IITB are custom Friend Lists of David
- Family = {Alice}
- IITB = {Bob, Alice}

- Family is the label defined by Facebook whereas IITB is the custom list created by David
- Now P1's privacy settings are : Friends - Family as shown in the Figure 5.5 below:

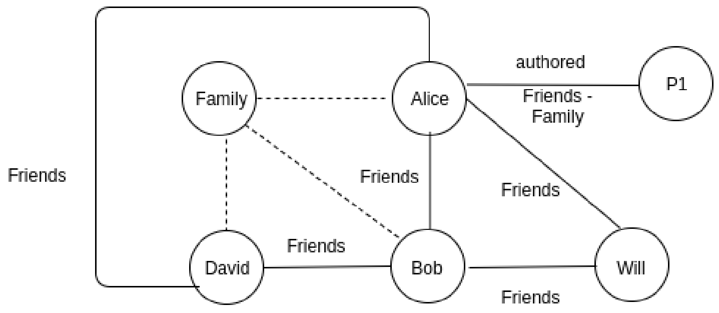


Figure 5.5: Social Graph in Consideration

- Now the result of the query is as follows:

```
privacy:
allow:""
deny:"150535245781326"
description:"Friends"
friends:"ALL_FRIENDS"
value:"CUSTOM"
```

- deny field here gives the id of the Friend List "family" which is excluded from accessing the post P1. This id will remain same even if we add or remove members from this list.
- Now if P1's privacy settings are : Friends - IITB then the result of the above query will be:

```
privacy:
allow:""
deny:"151612022340315"
description:"Friends"
```

```
friends:"ALL_FRIENDS"
value:"CUSTOM"
```

- In the result above, we can see that the Friend List "IITB" created by the the user David will also have a id and this id will also remain same irrespective of addition or deletion of members from the list.
- Conclusion : If for a particular post the privacy setting is only me, or custom then the app can easily get to know the privacy setting for that post leading to breach of privacy settings for a user. This is a serious privacy breach.

Experiment 6: Given a post id, retrieving reactions on the post

- David makes a post P1 and Bob reacts on the post with reaction "wow" as shown in the Figure 5.6 below:

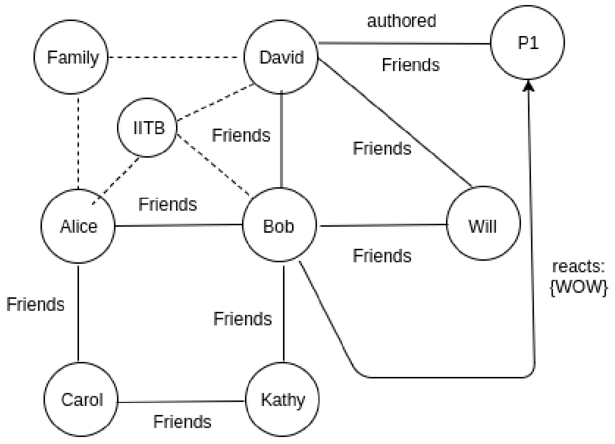


Figure 5.6: Social Graph in Consideration

- App1 can use the following Facebook API to retrieve the reactions on a post:

```
FB.api( "/{post-id}/reactions",
    function (response) {
      if (response && !response.error) {
            /* handle the result */
          }
        }
    );
```

- Result of the above API call is shown below:

```
    id:"104812420348221"
    name:"Bob Greenewitz"
    type:"WOW"
```

Experiment 7: Getting device info of the user

In order to retrieve device info of a user no explicit set of permissions are required. Suppose the user David in Figure 5.1 uses Facebook on his android device then App1 can use the Facebook Graph API in order to retrieve device info of David.
The API call used is as follows:

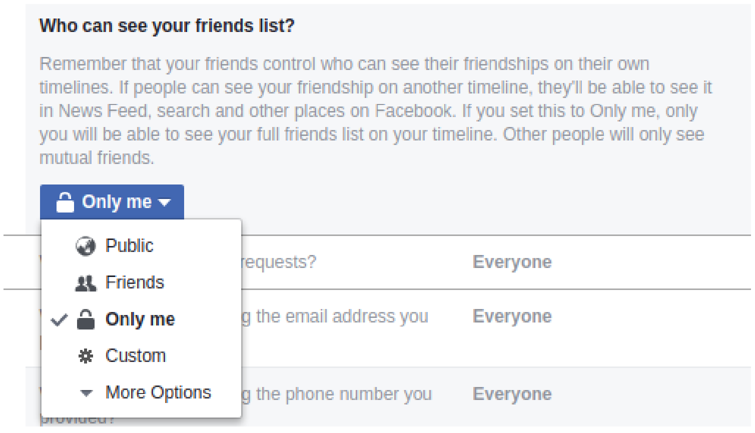
```
FB.api(
    "/me?fields=devices",
    function (response) {
      if (response && !response.error) {
        /* handle the result */
        console.log(response);
      }
    }
);
```

Result when the user say David is using Facebook on his Android device(using php sdk to query):

```
["devices"]=>
    object(Facebook\GraphNodes\GraphNode)#16 (1) {
      ["items":protected]=>
      array(1) {
        [0]=>
        object(Facebook\GraphNodes\GraphNode)#15 (1) {
          ["items":protected]=>
          array(1) {
            ["os"]=>
            string(7) "Android"
          }
        }
      }
    }
```

Experiment 8: Retrieving user's friends

- permission required : user_friends
- Access control policy of the user for friends list: only me(as shown in Figure 5.7


Who can see your friends list?

Remember that your friends control who can see their friendships on their own timelines. If people can see your friendship on another timeline, they'll be able to see it in News Feed, search and other places on Facebook. If you set this to Only me, only you will be able to see your full friends list on your timeline. Other people will only see mutual friends.

Only me

Public
Friends
✓ Only me
Custom
More Options

requests?	Everyone
g the email address you	Everyone
g the phone number you provided?	Everyone

Figure 5.7

- Information retrieved : friends who have also authorized the app.
- This also gives total count of user's friends.

- When the app queries the social graph for user's friends then the "/user-id/friends" edge provides the required information to the app.
- The API call used is as follows:

```
FB.api(
          '/me/friends',
          {fields: 'name,id,birthday,gender'},
          function(response) {
            console.log(response);


 });
```

- Result of the above query for user David, considering Figure 5.1 is as follows:

```
data:Array(4)
0:{name: "Will Will", id: "101400597342226"}
1:{name: "Alice Alice", id: "117622519064102"}
2:{name: "Bob Greenewitz", id: "104812420348221"}
length:3
summary:{total_count: 3}
```

- Now as the app has information regarding some of the friends of the user say A, it might reveal this information to its other users via which other users will also get an idea of some of the friends of the user which might lead to violation of access control policy for the user.
- An application can not only get name and id of user's friends but it can also get other information such as user's birthday, location given that user's friends have granted required permissions to the app.

App Finds out User's Friends:

Here we will see how App can deduce friends of a user other than those who have also installed the same App.

Consider the Figure 5.8 below:

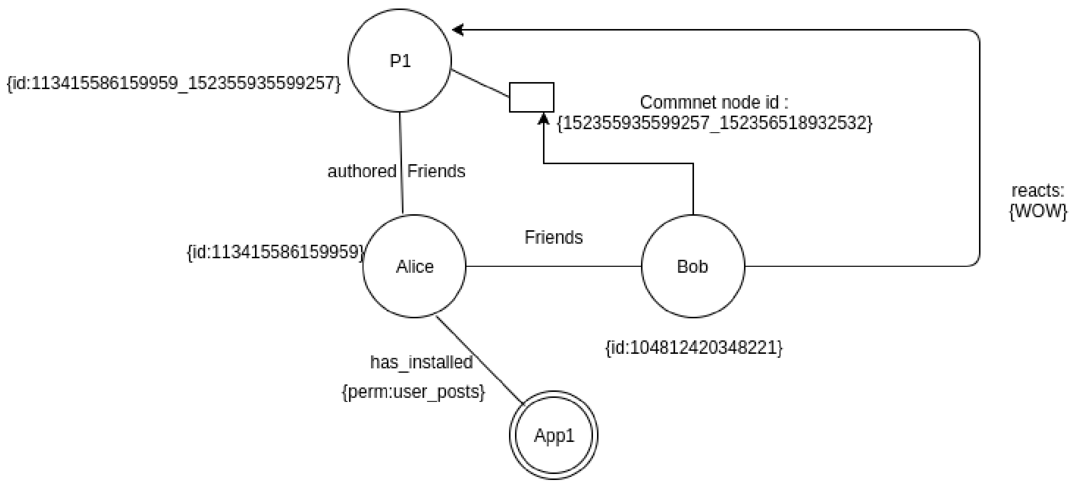
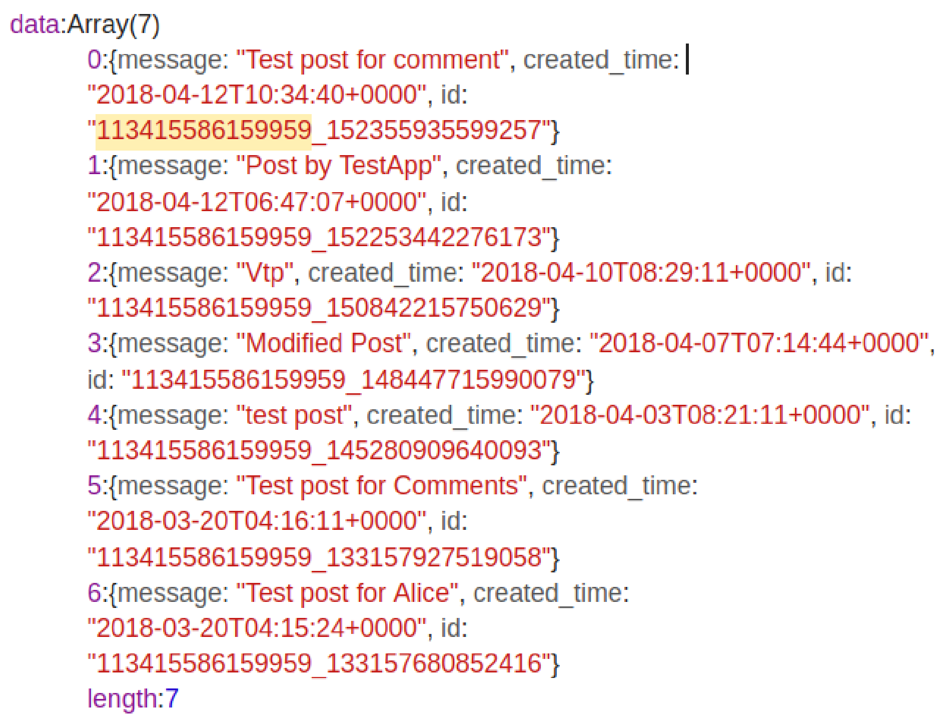


Figure 5.8

Facebook has deprecated Apps to access App1 its user's custom friend lists. Consider a scenario as shown in Figure 5.8, in which Alice has set her list of friends to private in her privacy settings. This setting sets an expectation that Alice's friend list will not be available to others. Alice installs App1 with permission user_posts. This permission allows App1 to reach all of Alice's posts and their fields (comments, reactions, post privacy settings). Figure 5.9 shows the list of posts retrieved by App1 from Alice's timeline. Retrieval of comment & reaction on the first post P1 is shown below. FB's NewsFeed function presents updates from Alice's timeline to her friends (Bob). When a friend interacts with the post, App1 can observe it and deduce with high probability that Bob is Alice's friend. Similarly, depending on post's permission policy setting, App1 can reason about other lists(such as Family) as well.

- Result when the App1 queries for the posts of Alice {id: 113415586159959}
- Result for Comments on P1:

```
data:Array(1)
0:
created_time:"2018-04-12T10:36:14+0000"
from:
```



```
data:Array(7)
    0:{message: "Test post for comment", created_time:
    "2018-04-12T10:34:40+0000", id:
    "113415586159959_152355935599257"}
    1:{message: "Post by TestApp", created_time:
    "2018-04-12T06:47:07+0000", id:
    "113415586159959_152253442276173"}
    2:{message: "Vtp", created_time: "2018-04-10T08:29:11+0000", id:
    "113415586159959_150842215750629"}
    3:{message: "Modified Post", created_time: "2018-04-07T07:14:44+0000",
    id: "113415586159959_148447715990079"}
    4:{message: "test post", created_time: "2018-04-03T08:21:11+0000", id:
    "113415586159959_145280909640093"}
    5:{message: "Test post for Comments", created_time:
    "2018-03-20T04:16:11+0000", id:
    "113415586159959_133157927519058"}
    6:{message: "Test post for Alice", created_time:
    "2018-03-20T04:15:24+0000", id:
    "113415586159959_133157680852416"}
    length:7
```

Figure 5.9: Posts retrieved by App1 from Alice's timeline

```
id:"104812420348221"

name:"Bob Greenewitz"

__proto__:Object

id:"152355935599257_152356518932532"

message:"Test comment by Bob"

__proto__:Object

length:1
```

- Result of Reactions on P1:

```
data:Array(1)

0:

id:"104812420348221"

name:"Bob Greenewitz"

type:"WOW"

__proto__:Object

length:1
```

- For App1 to query for node Bob, it requires access token of the user Bob which App1 can get only when Bob authorizes or installs App1.

Experiment 9: Understanding the behaviour of App Scoped ID

Each Facebook user is being addressed by a unique ID whose scope is restricted to the context for which it is generated. For example, App1 will generate a scope id,which is different from the scope id generated by App2. Thus App1 and App2 or their parent advertiser cannot interlink users across contexts. However, we have observed that, as of now, these scope IDs are publicly resolving to the real user for whom the scope IDs were generated. For example, https://fb.com/100007460080360, https://fb.com/2051781625080487, and https://fb.com/1708004396124880 reveal the actual Facebook user behind these scope IDs.

Experiment 10: Retrieving User's birthday:

- Permission required : user_birthday
- Access control policy of user : only me
- Information retrieved : birthday of the user in MM/DD/YYYY format, An application if granted user_birthday permission allows an application to access user's birthday. An application might reveal this information to user's friends which in one way or the other violates the access control policy of the user if the policy is only me.
- API call used is :

```
FB.api(
         '/me?fields=birthday',
          function(response) {
           console.log(response);
});
```

Experiment 11: App & Advertiser Can Identify Users: Linkability

[8]

Advertisers(example app advertisers) can target audience on facebook by sending the data of the users who use their app to facebook. The data includes email, dob, name, gender, locations etc. The data is shared in hashed format in order to maintain the privacy. Now using this data Facebook sees if more people can be added to ad's audience. Facebook Analytics is a useful tool used by the advertisers to target audience based on various app events.

Summary of privacy violations from the above scenarios:

1. App finds out user's friends despite user setting it private.
2. App can access user objects with "Only Me" policy.
3. App can find out what other apps are installed by its users.
4. Linkability: App and advertiser can identify their audience from the analytics data.

Chapter 6

Related Work

Web based social networks like Facebook, Twitter etc have shown tremendous growth in last decade as they helps users to establish digital identities and interact with each other. Recently, the usage of WBSNs have been increasing rapidly with about 300 websites collecting information of more than 400 million registered users [2] and it is important to protect this information. Studies have been done in past on social networks and [5] presents a survey on web based social networks. Privacy in social networks is important and can't be ignored. Research in this area has been done quite for some time focusing on privacy implication of connectivity [6]. With the recent Cambridge Analytica [3] Scandal(wikipedia refer) it is evident that more emphasis should be laid on privacy in social networks like Facebook. When an app comes into picture privacy-preservation becomes more relevant as knowingly or unknowingly users grant various permissions to app which might impact their privacy and can result in information leakage. This is discussed in [8]. Facebook platform can borrow concepts for privacy-preservation in an app ecosystem from [7] for mobile platforms.

Chapter 7

Conclusion

In this report, we presented how Apps play an important role in tracking and profiling users on Facebook platform. We also saw various scenarios where granting various permissions to an App resulted in violation of privacy setting of the user. This can not only compromise privacy of users but also their security. Therefore, App permission management need to be made understandable to the users and based on our findings we can say that the scope of user privacy policies across user layer, app layer, and beyond demands expansion.

Bibliography

[1] 2018. Facebook developers site. `https://developers.facebook.com/docs/`. Accessed: 24th June, 2018.

[2] 2018. Wikipedia site. `http://en.wikipedia.org/wiki/List_of_social_networking_websites`. Accessed: 24th June, 2018.

[3] 2018. Wikipedia site. https://en.wikipedia.org/wiki/Cambridge_Analytica.

[4] D.M. Ellison N.B. Boyd. 2007. Social network sites: Definition,history, and scolarship. In *Journal of Computer-Mediated Communication*. pages 210–230.

[5] Golbeck J. 2005. Web-based social networks: a survey and future directions.technique report .

[6] A Juels. 2001. Targeted advertising ... and privacy too. In *CT-RSA 2001.*. pages 408–424.

[7] S. Wong E.L. Goel D. Dahlin M. Shmatikov V Lee. 2013. Box: A platform for privacy-preserving apps. In *NSDI 13*. USENIX, pages 501–514.

[8] Patil Vishwas, Jatain Nivia, and R.K. Shyamasundar. 2018. Role of apps in undoing of privacy policies on facebook. DBSec.

Acknowledgements

I would like to thank my advisor, Prof.R.K. Shyamasundar, for his guidance during this research. The research discussion meetings have been a great learning experience for me. He was available whenever I needed help. He has helped me whenever I faced difficulties in my research work. I am indebted to him for his help.

Signature: ....

Nivia Jatain

15305R007

Date: ...... June 2018