Effectiveness of Social Networks in Building Whitelists

Aditi Rajoriya
Department of Computer Science
Columbia University
ar2630@columbia.edu

December 22, 2008

Abstract

This project deals with the creation of whitelists for spam detection using social networks. Social networks are rich source of self-defined reliable contacts. The goal was to capture these contacts from social networks and analyze different sources of email. The first phase of the project involved exploration of the OpenSocial API. The purpose of OpenSocial API is to retrieve information from social websites. In second phase for the project, we evaluated the effectiveness of this whitelist, which was on the basis of friends and friends of friends information gained from the OpenSocial API. Experiments were conducted with the OpenSocial API. Unfortunately, the OpenSocial API did not allow us to access email addresses due to privacy restrictions. So user interface features of social websites such as Orkut and LinkedIn were used to gather email addresses of contacts and manual classification was done to identify sources of emails. Social networks were very useful in creating whitelist using the first degree friendship network of a person.

Contents

1 Introduction

The goal of this project is to make use of social networks in creating email whitelists for spam detection. Social networks are a rich repository of a person contacts. Moreover, these contacts are self-declared: a prospective contact can enter a person's social network only if they are explicitly approved by the person. This makes them an extremely valuable source of reliable information regarding trustworthy emails. In this project, our original intention was to mine social networks for emails of contact and use them to categorize incoming email automatically using the OpenSocial API [6]. Due to the absence of such a provision in the OpenSocial API Documentation [7], we filed a feature request [8] for the addition of such an API via the OpenSocial IssueTracker [9]. We then proceeded to extract contact lists via user interface features and used these lists to manually classify email into meaningful categories. This classification was designed to reveal the various sources of email that people may have and to quantify the effect of including whitelists induced by a social network. Our results for the one-hop social network indicate that social networks may be beneficial sources of email whitelists. At the moment, second-degree contacts have not yielded much incremental improvement.

2 Background

2.1 Email whitelists

An email whitelist is a list of email addresses which the user trusts and wishes to receive email from. Emails from these addresses are regarded as safe for their inbox and must not be sent to the trash folder. Whitelists help users to avoid spam and unwanted emails. However, it is a significant challenge to keep whitelists updated and it can be quite cumbersome to manually update them with email addresses of people who you know personally. In this project, the aim was to gather trustworthy contacts from the self-declared contacts that exists on online social networks.

2.2 Social networks

Online communities like Orkut, Facebook and LinkedIn make it easy for people to stay in contact with a diverse range of friends on a daily basis. These communities have information regarding users, their interests and activities. Members can make friendship links between themselves and their friends thereby encoding their social circle as a social network on the web. In that sense, a social network is a graph of people and their links based on their friendships. Users in an online community propose these friendship links by sending friendship requests, and the link is created if their request is accepted. Thus, friendship links are deliberately created by a user and therefore can be extremely reliable indicators of friendship and trust.

Orkut, LinkedIn, My Space, FaceBook, Twitter, Friendster and Hi5 are some of the more popular networks while several smaller websites like Ning allow users to create smaller communities that share a common set of interests or concerns. Web users have created a significant database of their friends and contacts across all of these and many more online communities. Facebook claims that they have more than 140 million active users[1] out of an estimated 360 million web users worldwide[4]. This is almost 40% of web users worldwide and is growing rapidly. If we can utilize this massive database to create email whitelist, we can potentially automate whitelist creation for a huge number of people.

2.3 OpenSocial

OpenSocial is a set of common application programming interfaces for web-based social network applications. It was started launched by Google and MySpace [14]. Several online communities, like Orkut, LinkedIn, Friendster, have joined the OpenSocial Foundation and are committed to making their applications accessible via OpenSocial APIs. These APIs help third party developers to build engaging applications such as List the Friends, Give Gifts to Friends and Poke a Friend.

2.3.1 Architecture

An OpenSocial application is a javascript program that runs inside an OpenSocial container. The OpenSocial container provides a javascript API which maybe accessed by OpenSocial applications. This container is essentially an embedded javascript interpreter. An example of an OpenSocial container is a webpage running in a web browser that can run javascript. An application running inside a webpage makes OpenSocial API calls through javascript. Google Desktop is a standalone application that has an inbuilt javascript interpreter, and therefore it can function as a OpenSocial container. One example of an OpenSocial application that runs in Google Desktop is a Google Gadget. Google Gadgets are standardized applets specified by XML files, that access OpenSocial APIs and can run in any OpenSocial container, such as a webpage or Google Desktop. The XML files are converted into executable javascript code by a gadget server which is an OpenSocial container at the server side.

The architecture of OpenSocial is described by Figure 1. There is a something on the HTTP client and something on the Http server. The OpenSocial application resides on the Client inside a OpenSocial container. Since the OpenSocial javascript API can only be used at the client side, it needs some kind of mechanism to communicate with the server. This communication over the internet is done using the OpenSocial Restful protocol and OpenSocial RPC protocol. An OpenSocial Restful API [10] can be thought of as a set of URLs which respond to clients' HTTP GET requests with various kinds of information. It consists of a set of services provided by specific URLs. Each API is capable of responding to requests with data in all of three different representa-

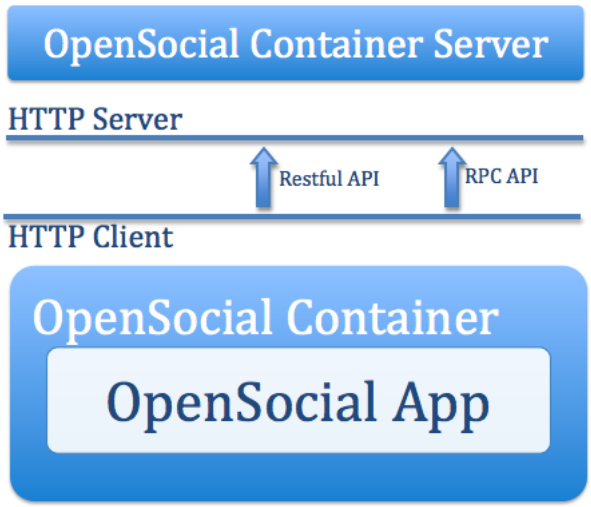

Figure 1: OpenSocial Architecture

tions: JSON, Atom Pub XML, and a custom Open Social XML. These three data formats are provided in order to make the API easier to use for applications with different requirements. An OpenSocial RPC protocol [16] is used to optimize the transfer of information, by batching queries and preserving state between calls.

2.4 Facebook API

Facebook has its own API for building applications. The Facebook API allows an application developer to add some social context to an application by providing access to profile and friend data. The Facebook API was present before Open Social API. However, the main focus in this project was to use OpenSocial because OpenSocial is a common platform used by several social networks. OpenSocial works on the principle of build once and run everywhere. Their are certain interesting differences in OpenSocial API and Facebook API. The Facebook API provides a RESTful interface for making method calls over the internet by sending HTTP GET or POST requests to Facebook servers. Nearly any computer language can be used to communicate over HTTP with the REST server. Unfortunately, Facebook also hides email addresses of friends and friends of friends due to privacy and security issues.

2.5 OpenSocial API vs Facebook API

OpenSocial applications can be both client side and server side. These typically require a high competence in HTML and especially in Javascript. The applications can also be server side but this does not seem to be a predominant

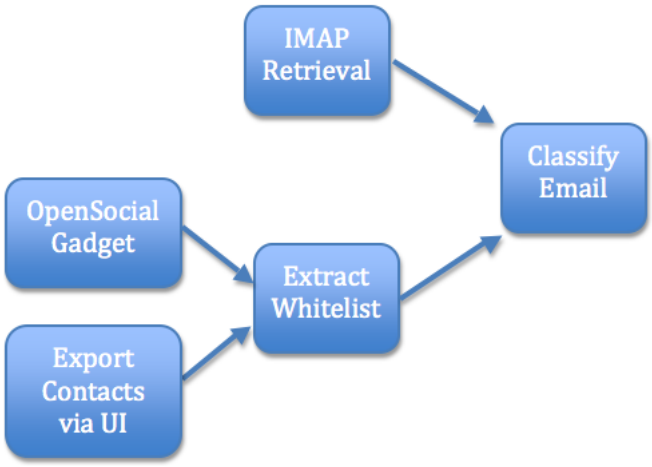

Figure 2: System Architecture

mechanism of application serving as yet. OpenSocial mainly focuses to build applets i.e small focused functionality. Open Source application run everywhere.

Facebook applications are all hosted on a web site. These typically require a reasonable amount of server side development using a variety of languages. PHP and Java client libraries are provided and supported by Facebook. Facebook focuses on building applications. Facebook applications run only on Facebook.

3 System architecture

The system architecture for this project includes four major modules. Each of the module has been described in detail below. Basically the IMAP module and Extraction of whitelist modules integrate with the Email classifier module to give the statistics. All these modules are return using Java language.

3.1 Extraction of whitelist using OpenSocial

A whitelist of social friends is retrieved from social websites such as Orkut and LinkedIn. OpenSocial provides access to the profile information of users so I created a javascript application that makes requests for a user and user's friends. This program is hosted on the Orkut Sandbox[11] and is known by the name *List My Friends*. Next, I created another application to get a list of friends and their email addresses. This is application is called *list of Friends Name and Emails* and is also available via the Orkut Sandbox [12]. This application returns null values for email addresses. This gave me an inkling that perhaps the OpenSocial API restricted access to email addresses of users. Discussions of this issue on the OpenSocial API mail group confirmed whether that OpenSocial does not

allow access to email addresses. The links to the discussion are available online [2], [15], [3].

3.2 Extraction of whitelist through user interface

Since the OpenSocial API did not allow us to retrieve email addresses and generate whitelists automatically, we thought of using the user interface provided within social networking applications for evaluation purposes. Applications provide a user interface to allow to save their contacts and import them into any other application, such as a mail client. I wrote a module to parse this file and to extract all the individual email addresses of my contacts.

3.3 IMAP retrieval

The main purpose of this module was to retrieve non spam emails and parse the email header's *to* and *from* fields. This module uses the JavaMail API[5]. This API provides the framework for sending and retrieving electronic messages. It provides access to the session object which is used to create a Store object. object. This object provides connectivity to the mail server and also a mechanism for accessing stored email. Through this object we can access mail folders, which can further be used to retrieve the messages that they contain.

This module processes the header of the emails in the Inbox and extracts the *to* and *from* fields and outputs a list of addresses from which mails in the inbox originate or end up.

3.4 Email classifier

The main purpose of this module is to classify the non spam emails and generate the statistics. It calculates the percentage of messages a person receives from his social friends and friends of friends on social network. This module also calculate the number of messages a person send to his friends from social networks. Both these statistics help us to evaluate the contribution of social networks in building whitelists.

4 Limitations of OpenSocial

4.1 Optional Information

Online communities give the user the option to control disclosure of profile information. The user can exercise this choice by refusing to enter any information by leaving that text-box empty. They can also restrict access to their profile of specific profile information by allowing only their friends, or the friends of their friends, to view it. Some users do restrict access to their personal email addresses, and we were not able to include those users in this study.

4.2 Scalability Problems with Friends of friends

Opensocial does not provide access to information of users two hops away from the user in question. Henceforth, these users will be known as *friends of friends* (FOF). The OpenSocial People and Friends API has a proposal that will provide information about friends of friends information but this API has not been released yet.

We considered using friends at higher than 2 degrees of separation as well. There were many discussions over the google groups with the basic import that this information might actually be of limited use. The main argument was that when we travel to all people at 6 degrees of separation from the user in question, we expect to have included a large fraction, if not all, of the entire social graph. This will cause an exponential explosion that can even result in the destruction of database server of the social website. Therefore, even if the proposal for an addition to the People and Friends API gets passed, there is a some doubt that some containers will choose not to support accessing friends of friends information. Moreover, there is very high computational cost to support friends of friends of friends.

Another factor is the fact that social websites continuously discover and suggest potential friends through mutual friends. As a result, many users end up making a direct link to most people who could be two-hops away in the real world.

4.3 Privacy restriction

Opensocial has email address as a field in its specification. Unfortunately, it's use is currently restricted due to concerns of privacy violation. This was confirmed by a message post at the OpenSocial application development group. Links of the communication with the OpenSocial developer are present in the appendix. Also, OpenSocial does not provide access to other contact related information such as home address, phone number, zip code etc.

5 Proposal for addition of API: Hashed email address

5.1 Proposal

Due to privacy limitations of OpenSocial API, it was not possible to get the email addresses of friends from the social networks. This triggered a need to outline a proposal to the Open Social Developers to generate hash code using hashing algorithm SHA 1[13] for email addresses. The hash code approach will allow developers to see only the hash code rather than the actual email address. Further string matching operations can be performed using that hash code as a substitute for email addresses. A feature request proposal [8] for incorporating

this hashing algorithm for the email addresses was proposed to the OpenSocial Foundation via the OpenSocial IssueTracker.

5.2 Discussions

There are many ongoing discussions on the OpenSocial mailing lists regarding this feature. People stated that initially that there were some arguments against the hash coding algorithm. A strong consensus is required regarding which of ShA-1, MD2 or MD4 should be used as the hashing algorithm. Another issue is that some people think these hashes are not secure enough because they still allow correlation across sites (site A and site B publish the same hash, so I can tell it's the same user on both places). In order to enable secure discovery, there is a need to have site specific hash i.e a hash of the user's email concatenated with the requesting site's URL. That way, if the site already has the raw email address, they can re-compute the hash and verify the match, but they can't reuse the site-specific hash anywhere else. This is essentially what MicroID does (hash of email and url can be safely published on a profile URL), but this has yet to be widely adopted.

6 Evaluation

The main objective of the evaluation was to measure the effectiveness of social networks in building whitelists i.e total number of emails a person receive from his friends and friends of friends on social websites. Due API limitations, it was not possible to retrieve user data from social websites. We decided to go ahead and perform evaluation using the user interface option on social websites to export contact information via csv files. The limitation of this approach is that we were not able to perform calculations for the second degree of friends. The second degree calculations were done manually by asking friends to export their contact.csv files and email them to use for analysis.

1. Total Number of Non Spam Messages : 8531
2. Total Number of Spam Messages : 3683

6.1 Manual Classification of Non Spam Emails

The non spam emails were categorize manually in order to analyze the sources of email addresses.

6.1.1 Mailing Lists

This category contains all mails directed towards mailing lists. The fact that most of these email addresses contain the words list or listings aids the categorization process. Examples: MailingList@exhedra.com, jobs-listings@linkedin.com. Moreover, the sender header of such an email contains the list information and

list-related headers, such as the List-Id head, List-subscribe. This category was rather significant in my inbox, since I have subscribed to 71 mailing lists.

6.1.2 Auto Lists

Lists All email addresses which contain any of the words 'auto', 'auto-email','auto-confirm','autoreply','noreply' or 'support' in their From fields are binned into this category. These emails are auto generated and are informational in nature. Examples: autoreply@geowebnews.com, autoemail@noreply.buy.com. These lists also contains the set of web related email addresses. Web related email addresses are the addresses from the websites, where user had already gone before and left his email address while creating an account for that website. They also contain the transactional email addresses regarding confirmation of your payments and recipt.

6.1.3 Google Groups Contacts

Email groups on popular services such as googlegroups and yahoogroups also generate a huge number of emails. In such emails, the from field contains the name of the sender, and the to field contains the email address of the group. This category also generates a huge number of emails since people tend to discuss issues of important to them. This category is different from mailing lists in that mailing lists are usually used for one-way announcements, new releases, newsletters. Email groups are frequently a channel for vigorous mutliway discussions.

6.1.4 Gmail Friends

This is the list of my contacts on Gmail. This list can also be created by importing contacts from any of the social networks.

6.1.5 User Owned Domain Friends

People usually have one email address that is associated with an organization such as their workplace or their university. As a consequence of their participation in the functions of their organization , they tend to have a huge amount of communication with people with emails addresses within the domain. As a student of Columbia University, several of my friends and collaborators are from Columbia and for a lot of the university related communications occurs within emails on that same domain. This category can also be broadly categorized as the list of known legitimated domains such as IBM.

6.1.6 Orkut Friends

This is the list of email addresses from friends on Orkut. This list was retrieved using the export feature of user interface.

6.1.7 LinkedIn Friends

This is the list of email addresses from friends on LinkedIn. This list was retrieved using the export feature of user interface. Since I am not active on this network, I have very few friends on this network.

6.1.8 Orkut Friends of Friends

This is the list of email addressses from friends of friends on orkut. This list was retrieved using the export feature of the user interface. This list was gathered manually by asking people to send their contacts.csv. I was successful in convincing only one person to give his csv. So the results are based on a single friends of friends information. These results could have been much much better if OpenSocial API supported the friends of friends information.

6.1.9 Good Strangers

This is the list of those email addresses who are not known initially but were mailing for the first time. They had been in communication either through a social gathering or mobile. But later they became friends on social networks. The frequency of messages increased once I came to know the background information of the person. This category varies according to the different inboxes of different users. In my case the contents of these email addresses gives the background information about this person and his interests.

6.1.10 Bad Strangers

This is the list of unwanted email addresses of unknown people and domain. This category has a single frequecy of email from each of the email addresses. That means these emails were never successful in starting a two way communication.

6.2 Results

The results of categorization of email from my GMail Inbox into the bins described above are as follows. The results are available in tabulated form in Table 1. A pie chart describing this data is shown in Figure 3.

6.2.1 Legitimate Emails from Strangers

1. Number of Legitimate Emails from Strangers = 483
2. Number of total Emails = 8531
3. Percentage of Legitimate Emails = 5.661 %

Table 1: Classification of 8525 non-spam emails

Category	Number of Email Addresses	Number of Messages	Percentage
Mailing Lists	71	3544	41.55 %
Auto Lists	99	1041	12.20 %
Google Contacts	56	971	11.38 %
Gmail Friends	82	803	9.41 %
Bad Strangers	683	683	8.006 %
Orkut Friends	110	680	7.98 %
Good Strangers	277	483	5.661 %
User Owned Domain Friends	56	248	2.9 %
Orkut Friends of Friends	10	72	0.844 %
LinkedIn Friends	4	6	0.07 %
Total	1448	8531	100.00 %

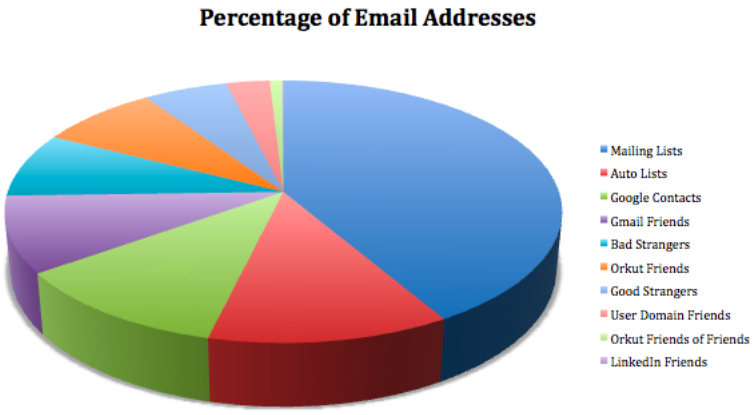

Figure 3: Results of Email Categorization

6.2.2 Non Legitimate Emails from Strangers

1. Number of Non Legitimate Emails from Strangers = 683
2. Number of total Emails = 8531
3. Percentage of Non Legitimate Emails = 8.006 %

6.2.3 Sent Mail Stats

1. Messages sent to Orkut Friends = 56
2. Number of emails in sent mail folder Gmail Inbox = 219
3. Ratio of email messages from the orkut friends in Gmail Inbox to the total number of sent email messages in Gmail Inbox = 56/219 = 25.5 %

6.2.4 Friends Stats

1. Number of Orkut Friends = 110
2. Number of Messages from these Friends = 680
3. Ratio of email messages from the orkut friends in Gmail Inbox to the total number of email messages in Gmail Inbox = 680/8531 = 7.98 %

6.2.5 Friends of Friends

1. Total Number of Orkut Friends of Friends = 10
2. Total Number of Messages from these Friends of Friends = 72
3. The ratio of email messages from the orkut friends in Gmail Inbox to the total number of email messages in Gmail Inbox = 72/8525 = 0.844 %

7 Conclusion

This project dug up important details and limitations of OpenSocial API that cast doubt over its usefulness for whitelist creation. We set the ball rolling, towards speedy resolution of these concerns by creating and submitting a proposal for a hash code mechanism for email addresses. Secondly, the project was successful in the identification and categorization of emails into meaningful categories. We evaluated the improvement in whitelists created using social networks at using friends one and two hops away. Our finding was that while the first degree social network can be extremely useful for generating whitelists, the effectiveness of the second degree network is not yet validated. Also those emails were classified whose source was unknown initially.

References

[1] Facebook statistics.

[2] Google group open social thread.

[3] Google group open social thread.

[4] Internet world stats.

[5] Java mail api documentation.

[6] Opensocial api.

[7] Opensocial api specification.

[8] Opensocial feature request: Hash code for email addresses.

[9] Opensocial issuetracker.

[10] Opensocial rest protocol.

[11] Orkut friends.

[12] Orkut friends and emails addresses.

[13] Sha1 secure hash algorithm - version 1.0.

[14] Wikipedia article: Opensocial.

[15] http://www.mail-archive.com/opensocial api@googlegroups.com/msg03448.html. Opensocial discussion group.

[16] http://www.opensocial.org/Technical-Resources/opensocial-spec-v081/rpc protocol. Opensocial rpc protocol.