This is the published version


Available from Deakin Research Online

http://hdl.handle.net/10536/DRO/DU:30060007

Reproduced with the kind permission of the copyright owner

Copyright: 2014, Academy Publisher
Investigating Facebook Groups through a Random Graph Model

Dinithi Pallegedara, Lei Pan
School of Information Systems, School of Information Technology,
Deakin University, 221 Burwood HWY, Burwood, VIC 3125, Australia
Email: {dpallege, l.pan}@deakin.edu.au

Abstract—Facebook disseminates messages for billions of users everyday. Though there are log files stored on central servers, law enforcement agencies outside of the U.S. cannot easily acquire server log files from Facebook. This work models Facebook user groups by using a random graph model. Our aim is to facilitate detectives quickly estimating the size of a Facebook group with which a suspect is involved. We estimate this group size according to the number of immediate friends and the number of extended friends which are usually accessible by the public. We plot and examine UML diagrams to describe Facebook functions. Our experimental results show that asymmetric Facebook friendship fulfills the assumption of applying random graph models.

Index Terms—Facebook groups; random graph; system analysis; UML.

I. INTRODUCTION

Facebook is the most popular social networking website on the Internet with more than 1.15 billion active users in March 2013 [1]. Improvements and advancements are massive and still growing at the time when this paper is written. Meanwhile, the large amount of privately shared information on Facebook has raised social security concerns which incurred suspicious involvement with government-initiated surveillance programs. No matter whether Facebook provides personal information to governmental agencies, most police in many countries outside the U.S. encounter difficulties when obtaining Facebook user information according to Acquisti and Gross [2].

Specifically, organized criminal groups started to utilize Facebook as a communicating channel [3], [4]. According to Varese [4], Mafia groups use Facebook to store and disseminate encrypted files containing stolen credit card details, contacts and other contraband information. Due to the large and complex organization of Facebook users, successfully identifying suspects as Facebook users is a big challenge for law enforcement. Till date and to the best knowledge of the authors, there is no established method in practice which effectively deals with this challenge. Hence, the first step towards identifying the suspect group members is to estimate the size of the group without using log information from server.

The research problem of this paper is to estimate the efforts required by the law enforcement for investigating crimes and incidents coordinated via Facebook groups without requesting information from the service provider. Moreover, the results of this paper contribute to narrow down the investigative scope to the most relevant users. This research is meaningful for law enforcement personnel who reside in different countries where back-end log information from Facebook cannot be easily obtained.

In this paper, we translate forensic investigation on Facebook to the vertex cover problems on graphs. That is, we propose that investigations should be carried out on a number of suspects who form a minimal cover of vertices in a graph representing their connectivity. According to the graph theory, a vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set. Our method is equivalent to finding vertex cover of minimum size.

Definition I.1. \( G = (V,E) \) is an undirected graph with vertex set \( V \), and edge set \( E \). Each vertex \( v \in V \) has a weight \( w(v) > 0 \). \( S \subseteq V \) is a vertex cover of \( G \) if and only if all edges \( e \in E, e \) is incident to a vertex in \( S \) [5].

However, the vertex cover problem is known as a NP-complete problem. A problem called Non-deterministic Polynomial (NP) if its solution can be measured in polynomial time and if a problem is NP and all other NP problems are polynomial time reducible, then the problem is NP-complete. According to Weight and Hartmann [6], a discontinuous transition in solvability and typical-case complexity occurs when the size of the cover set reduces to a critical value. And “this transition is characterized by means of exact numerical simulations as well as by analytical replica calculations” [6]. Therefore, our problem of identifying a Facebook group becomes the derivation of the critical value — the group members can be isolated once we manage to know the critical value of this group.

To address our research problem, we tackle the following two issues:

- How to estimate the number of related Facebook users based on a probability distribution?
- How to derive the critical value of a Facebook group when a giant component forms?

Section II includes the background work on set theory, UML and random graph models which lead to the development of Facebook friendship relation. Section III lists our UML diagrams to describe how Facebook users communicate to each other. Section IV applies a random graph model to derive an indicative estimation.
of particular Facebook groups. Section V contains our experiments and results. Finally, Section VI concludes this paper.

II. BACKGROUND WORK

A. Relations for Facebook Friendship

Facebook friendship is a relationship between elements of sets. Relationships between elements of sets are represented using the structure of a relation [7]. A relationship between elements of two sets is expressed using ordered pairs or binary relations. Binary relation from $A$ to $B$ is a set of $R$ of ordered pairs where the first element of each ordered pair comes from set $A$ and the second element comes from set $B$. We use $ARB$ to denote that $A$ is a Facebook friend of $B$.

To describe Facebook friendship, we need to fulfill certain relational properties [7]:

- An element is always related to itself or a relation $R$ on a set $A$ is called reflexive if $aRa$ for every element $a \in A$.
- If an element $a$ is related to an element $b$, and $b$ is in turn related to an element $c$, then $a$ is also related to $c$; that is, a relation $R$ on a set $A$ is called transitive if whenever $aRb$ and $bRc$, then $aRc$, for all $a, b, c \in A$.
- If $aRb$, then $b$ is related by a symmetric relation $R$ to $a$ or on a set $A$ if $bRa$ whenever $aRb$ for all $(a, b) \in A$.
- If every element in $a$ also is in $b$ and every element in $b$ is in $a$, then $a$ and $b$ must be equal or an antisymmetric relation $R$ on a set $A$ such that for all $(a, b) \in A$, if $aRb$ and $bRa$, then $a = b$.

In our context, there is no reflexive relation in Facebook friendship. Conversely, transitivity holds in certain situations but not always. Facebook friendship seldom falls to the category of antisymmetric relation due to the fact that it is rare for two friends to have identical friends. For a transitive relation, $ARB$ and $BRC$, then $ARC$. It is likely for $A$ to be friends with $C$ if they share a common friend $B$. But it is not always possible for $A$ and $C$ to be friends. In reality, friendship is often intransitive. Even though friendship is not necessarily transitive, it can be viewed as a symmetric relation. In a symmetric relation, if $A$ is related by $R$ to $B$, then $B$ is related by $R$ to $A$, which can be true for Facebook friendship. Therefore, we can model the Facebook friendship by using a set of users and a binary relation of friendship which is irreflexive and symmetric.

Friendship is a symmetric relation but the way in which the attitudes are shaped by the community is asymmetric [8]. The reason is simple — different people have different access to information and processes. Similarly, the concept of a social group is to build up a community that is based upon an interest, same view, likeness or dislikeness or some kind of association. In this paper, we define a “Facebook group” as a set of friends closed under the asymmetric relation of friendship.

B. Random Graph Model

To avoid unnecessary transitivity, we use random graphs to model Facebook group. Though graphs without probability can be used to model social networks, traditional graphs fail to reflect the dynamics of social relationship between users; and it is improper to infer a social relationship between Alice and Bob based on the fact that both Alice and Bob know Cathy. The latter intransitivity problem is addressed in random graphs by probability.

The term probability distribution depicts a statistical function that describes all the possible values that a random variable takes within a given range. To measure how well Facebook users are connected to each other according to an advanced graph theory model, we apply the existing theory of random graphs. Random graphs are a broad concept because in real world social networks, vertices and edges are dynamic and change overtime. However, we try to generalize the rules influencing the dynamics of the vertices and edges using random graphs as a prototype.

Concept of random graph is similar to an ordinary graph. The minimal model consists of $n$ nodes or vertices joined by links or edges but the pairs of vertices are chosen randomly. Every possible edge between two vertices is represented with independent probability $p$, and the absent with probability is given by $1 − p$ [9].

Bollobas [10] defines a random graph as a graph generated by a probability distribution — randomly adding edges between a given set of $n$ vertices. A random graph is denoted as $G(n, p)$ where every possible edge between the $n$ vertices occurs independently with probability $p$. In a random graph, if $n$ is big enough and $p$ is equal to neither 0 nor 1, then given any $n + m$ vertices $a_1, \ldots, a_n, b_1, \ldots, b_m \in V$, there is a vertex $c \in V$ that is adjacent to each of $a_1, \ldots, a_n$ and is not adjacent to any of $b_1, \ldots, b_m$.

Let a random graph $G(n, p)$ present a social network such as Facebook by assigning each vertex to a user profile and by using each edge as the relation between two user profiles regarding a specific object related to a crime. The above property can be of interest for digital forensic investigators. The fact that some vertices are adjacent to a subset but not adjacent to other subset indicates that the law enforcement may isolate a suspect from many user profiles.

Furthermore, the closeness of users can be represented if random graphs are used to model Facebook. That is, the police may investigate the user closeness in a social group. This closeness can be measured by the ratio of the number of trios of users; each of whom is connected to both of the others and the number of a user connected to two others, which is $C = 3 \times \# \text{triangles} / \# \text{connected triples}$ and $0 < C < 1$. Given a $G(n, p)$, the distribution function $p$ is equal to clustering coefficient $C = p$, according to [9], which is the average probability that two neighbors of a given vertex are mutual neighbors of one another.

The average degree of a vertex is denoted by $z$ and the
degree of a vertex is the number of edges connected to a particular vertex. In a social network, $z$ means the number of friends a person has. To describe the activeness of the vertices in a random graph $G(n, p)$, derives the average degree of a vertex as [10]:

$$z = \frac{\text{Perm}(n, 2) \times p}{n} = (n - 1) \times p \simeq n \times p,$$

where the last approximate equality holds for large $n$. For small values of $z$, when there are few edges in the graph, most vertices are disconnected from one another. Regarding a cyber crime within a group with little connectivity, the police can easily isolate the suspects because users have a fairly simple relationship with each other resulting in less communication messages.

Components in a graph represents a social group in Facebook. In a graph, a component is a subset of vertices in the graph each of which is reachable from the others by some path. With increasing $z$, a giant component forms. The giant component is the largest component whose size scales linearly with the size of the whole graph when $z$ is above a critical value, according to [10], that is, the size of the giant component scales linearly with the size of the graph. In Facebook social groups, the critical threshold corresponds to a minimum amount of communication effort by individual vertices, above which a global property exists with a high probability; when the effort is below the threshold, the desired global property exists with a low probability. Such global property includes user reachability with probabilistic friend-introduction, social group connectivity, group coordination and so on.

C. UML Diagrams

Unified Modeling Language (UML) diagrams enable developers and clients to view a software system design from a different perspective and in varying degrees of abstraction. In our scenario, we will use Class Diagrams, Sequence Diagrams and Data Flow Diagrams to demonstrate Facebook friendship. In particular, we focus on the scope of determining a Facebook group.

A use case diagram represents the interaction between users and the information system visually and communication at a higher level. The user becomes the actor with his/her role and the diagrams shows how he/she interacts with different tasks. Whenever an actor involves with an interaction described a use case, an association exists [11]. When a Facebook user receives a friend request from another user, he or she may accept or ignore the friend request. The extension use case contains one or few behavior segments that explain the additional behavior of the base use case. Every segment can be added into the base use case at a different point and it is called the ‘extension point’.

Sequence diagram demonstrates the objects that participate in a use case and the messages that pass between them over time for one use case [11]. Sequence diagrams are effective where we need to understand the real-time specifications and complicated use cases as it displays the time-based ordering of the activity among objects. Thus, we would depict the objects and messages for adding a known friend using a sequence diagram.

A class diagram describes the static structure of the system and gives us a detailed view of a single use case and the relationships among the classes [11]. The core of a class diagram is the class which stores and manages information — the attributes are the properties of the class where we capture information, and operations are actions or functions that a class can perform. In a Facebook User Profile, users are required to provide their personal information including email address, password, gender, date of birth and other mandatory items. After the profile is created, users have the right to declare some information public or private. In the next section, we represent the basic attributes and operations of a typical Facebook user and this information is revealed to the public Facebook community.

Data Flow Diagrams (DFD) are a commonly used process modeling technique to depict the business processes and the data that passes through them. DFD focuses on the processes or activities that are being performed. According to [11], the elements of a DFD are defined as follows:

1) Process is a function or an activity that is executed for a particular business reason.
2) A data flow is a data element or a collection of information.
3) A data store is a collection of data that is stored.
4) An external entity is a person, organization or system that is external to the system but interacts with the system.

In reality, a single DFD is often insufficient to include all the processes, when a composed set of DFDs is of use. The composition of DFDs works as follows — the first DFD provides a summary of the overall system, and the next level of DFDs provide a detailed description of each and every process of the overall system. A level of diagrams is categorized as Context Diagram, Level 0 Diagram, Level 1 Diagram, and Level 2 Diagram. Context Diagram shows the entire system in context as one process and the data flows to and from external entities. The next level of DFD diagrams is called Level 0 Diagram. It shows all the processes at the first level of numbering, data stores, and external entities and data flows. A Level 0 DFD has the purpose of visualizing all the major high-level processes of the system and how they are related with each other. After the information of Level 0 DFD is balanced, we can proceed the same process to derive Level 1 and Level 2 Diagrams.

All the above described UML diagrams will be used to understand the relationships on Facebook including search for a particular user, add user and describe how Facebook friendship is formed using a system design method.

III. System Analysis for Facebook Friendship

We begin with the UML notation of different user scenarios beginning with the Class Diagram of a typical User.
Profile of Facebook and Use Case Diagram, Sequence Diagram, Data Flow Diagram for adding Facebook users.

As shown in Fig. 1, a typical Facebook User Profile has the basic attributes including name, home town, gender, current town, networks, friends, mutual friends, profile picture etc. Name is a mandatory field for every user; and the profile picture is usually visible to other users.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>+name</td>
<td>+searchUser()</td>
</tr>
<tr>
<td>+home town</td>
<td>+addFriend()</td>
</tr>
<tr>
<td>+gender</td>
<td>+sendMessage()</td>
</tr>
<tr>
<td>+current town</td>
<td>+poke()</td>
</tr>
<tr>
<td>+networks</td>
<td>+shareProfile()</td>
</tr>
<tr>
<td>+mutual friends</td>
<td>+report/block()</td>
</tr>
<tr>
<td>-date of birth</td>
<td></td>
</tr>
<tr>
<td>-address</td>
<td></td>
</tr>
<tr>
<td>-relationship status</td>
<td></td>
</tr>
<tr>
<td>-wall</td>
<td></td>
</tr>
<tr>
<td>-photos</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. Class Diagram for Facebook User Profile

These attributes are shown as public attributes with the "+" sign in the class diagram. By default, most of these attributes are visible to others and we need to change the visibility using the privacy options. The fields name, gender, current town, networks, friends and mutual friends are visible to everyone; but date of birth, address, relationship status, wall and photos are kept as private attributes. Hence, these attributes are controlled by the privacy options so that the user’s friends could view them. For the operation section, we have given only the operations a user can initiate before confirming a friendship with another user. Search user, add friend, send message, poke, share profile and report/block are such basic operations. Depending on the level of privacy these operations could be private (-) or protected (#) because some users only allow their friends to perform several operations and they could be hidden to the public users. Class diagrams are efficient in visualising this factor because it is possible to show the visibility of the attribute of the diagram. However, to demonstrate a friendship using a class diagram seems confusing because any user on Facebook would model the same class with the same attributes and operations given to the each Facebook Profile class.

We follow the process in [11] to draw a use case diagram for the “Add User” feature of Facebook. Use cases are connected to the actors in the following manner:
1) User A searches for user B on Facebook.
2) If user B exists on Facebook, then A’s friend request is sent to B.
3) Upon receipt of the request, B can choose to accept or to decline.
4) No matter what B’s choice is, a confirmation notification is sent back to user A.

Since these actions may be repeated, we denote them with the asterisk symbol "*". Fig. 2 shows the process of adding a known friend where there are two main actors user A and user B. Moreover, Search, Send Friend Request, Accept Friend Request or Ignore Friend Request are the main actions associated with adding a Facebook friend. As for an instance, any user can search for a user or send a friend request as many times as he or she wishes.

We draw a sequence diagram for this process in Fig. 3. It consists of two actors A and B and the Profile object. The dotted line denotes the lifeline of the actors over time. We draw the execution occurrence in the thin rectangles to indicate when the classes send and receive messages. When user A wishes to communicate with a non-associated friend B, A sends the Request Friend message; upon receipt of this message, B initiates a Search User message and sends it to the object Profile which holds the public information of A. Upon B’s approval, B sends the Acceptance message to A; otherwise, a default message Ignorance is generated.

Then we draw the context diagram for Facebook depicted as number 0 in Fig. 4. This diagram is drawn on the basis of the use of Facebook: Firstly, user A types
the user B’s name in the Facebook search bar, and the system finds the possible profiles which match with the typed name; then user A could find the correct profile from the result list; if successful, user A sends a friend request to B; User B could either accept the friend request or ignore it; Depending on that decision a friendship is granted or refused, then Facebook notifies the results to user A. According to our findings by using Facebook, we have four use cases — Search for User Information, Send Friend Request, Accept Friend Request and Ignorance. Note that the database is internal to the Facebook server system and hence is outside of the context diagram.

Fig. 4. Context Diagram for adding a friend on Facebook

Because an action of Add Friend requires the correct location of user profile, we need to further decompose the context diagram to level 0 data diagram. Concretely, we enumerate Search Profiles as a level 1 diagram by replacing the context diagram’s single process with a new process numbered as 2 for Add Friend and adding a data store D1 in Fig. 5. We preserve all of the inputs and outputs associated with the context diagram in the level 0 data diagram so that the level 0 DFD is well balanced. We introduce a data store where Facebook stores user data, which is not accessible by general public. We number Add Friend as 2 which means it relies on the results from the process 1 (Search Profiles). When process 1 finishes searching for the profiles, it sends User Profile Data to process 2 and returns the specified User Profile to user A. Upon user’s approval, the system sends a friend request to the process 2. Now process 2 starts to negotiate with user B for friendship establishment. As described in the context diagram, a Friend Request is therefore sent to B.

Furthermore, the level 0 DFD has a multi-staged process Search Profiles. For this process is important for normal users to find any unknown friends, we decompose this process further into three children processes — processes 1.1 (Search using User Information), process 1.2 (View Search Results) and process 1.3 (View Profile). These children processes are very similar to each other as shown in Fig. 6. That is, when user A types the B’s name, process 1.1 searches for the name using User Information and outputs the Profile Matches from the user database D1; the results are subsequently forwarded to process 1.2 from which user can view; the matched profile information is available to process 1.3 for display purposes.

Fig. 5. Level 0 DFD for Adding a Friend on Facebook

Fig. 6 shows the level 1 DFD for adding a friend on Facebook. Different to level 0 DFD, we add two new data flows as Results and Profile Information which are embedded within process 1 of the level 0 DFD. Since the origin and the destination of these data flows are only indicated in level 0, we need to refer to both DFDs.

In Section III, we analyze the friendship on Facebook using several UML diagrams. We understand that the Facebook friendship is only established by mutual agreement and authorization. Fig. 5 and 6 tell us that the search results are crucial for establishing a Facebook friendship. Dynamic Facebook friendship can be modeled by using graphs. To separate a group of Facebook users associated with a suspect, we propose to use a random graph which only requires local knowledge inside the group instead of the global knowledge which is difficult to acquire and analyze.

IV. MODELING FACEBOOK WITH RANDOM GRAPHS

The probability \( p_k \) that a vertex in a random graph has degree exactly of \( k \) is a Poisson distribution, according to [9]:

\[
p_k = \frac{e^{-z} z^k}{k!}.
\]

The Poisson distribution is peaked about the mean \( z \), and has a large-\( k \) tail that decays rapidly as \( 1/k! \). Let \( p_k \) be
the probability that a randomly chosen vertex has degree $k$, and $q_k$ be the probability that a vertex is the suspect given that it has degree $k$. Then the probability $p_k q_k$ is the probability of having degree $k$ and being the suspect [12], so that

$$F_0(x) = \sum_{k=0}^{\infty} p_k q_k x^k$$

(IV.1)

is the probability generating function for this distribution. For the trivial case when $x = 1$, $F_0(1) = q$ where $q$ is the overall fraction of suspects on Facebook.

In a graph $G(n, p)$, if we follow a randomly chosen edge, the vertex we reach has degree distribution proportional to $k \times p_k$ because a randomly chosen edge is more likely to lead to a vertex of higher degree. [12] derives the probability of such a vertex as

$$F_1(x) = \frac{\sum_k k p_k q_k x^{k-1}}{\sum_k k p_k} = \frac{F_0'(x)}{z},$$

(IV.2)

where $z$ is the average vertex degree defined in formula II.1.

Let $H_1(x)$ be the generating function for the probability that one end of a randomly chosen edge on the graph leads to a social group of a given number of suspect vertices. The social group may contain zero vertices if the vertex at the end of the edge is innocent, which happens with the probability $1 - F_1(x)$, or the edge may lead to a guilty suspect vertex with $k$ other edges leading out of it, distributed according to $F_1(x)$. [9] defines $H_1(x)$ the generating function that one end of a randomly chosen edge leads to a given number of suspects to satisfy a self-consistent condition:

$$H_1(x) = 1 - F_1(x) + x F_1[H_1(x)]$$

(IV.3)

The probability distribution for the size of the group to which a randomly chosen vertex belongs is generated by $H_0(x)$, where

$$H_0(x) = 1 - F_0(x) + x F_0[H_1(x)]$$

(IV.4)

From equations IV.1, IV.2, IV.3 and IV.4, we can determine several quantities of interest such as mean group size, position of the suspicion threshold, and giant component size.

The sizes of the social groups correspond to the size of a cyber crime among groups of suspects. If it is below the transition phase, the number of suspects is small, and there is no global behavior. Newman et al. [9] define two generating functions for vertex degree in a random graphs as $G_0(x) = \sum_k p_k x^k$ and $G_1(x) = G_0(x)/z$. Then, the mean group size is derived as

$$\langle s \rangle = H_0'(1) = q + q G_0'(1) H_1'(1) = q \left[ \frac{q G_0'(1)}{1 - q G_1'(1)} \right],$$

which diverges when $1 - q G_1'(1) = 0$. This condition determines the critical value when a global crime occurs, the point at which a giant component of connected vertices starts to form. The critical suspicion probability is

$$q_c = \frac{1}{G_1'(1)}.$$
The connection between other nodes looks like a tree in graph theory. A tree is an undirected graph in which any two vertices are connected by exactly one simple path as shown in Fig. 7. Since the nodes are opened, the root, that is the main suspect, can be isolated from the group.

V. EXPERIMENTS AND ANALYSIS

This section illustrates our experiments to search Facebook users without any system-level privilege. We use Wireshark and WebScarab because Facebook disallows ordinary users to search blocked users.

To understand Facebook user groups, we need to find the group boundaries which are formed by the unknown and/or blocked users. The flow chart for the steps of performing this task is shown in the Fig. 8.

When user A blocks B, B cannot view A’s profile; thus, user A’s profile becomes invisible to its blocked users — A will not appear in friend lists or search results. Facebook regards blocking a user as confidential and the blocked user is not notified. Nevertheless, using third-party applications may circumvent Facebook’s blockage. We use a network security tool WebScarab to test whether Facebook blocking scheme works properly as an effective boundary to delimit different user groups.

Firstly, we capture the network packets using Wireshark for the following scenarios, save the packets in binary dumps and name them accordingly:
1) Search a user on Facebook and store all packets in the file Search-user-A-packet-no01.pkt
2) Add a friend on Facebook and store all packets in the file Add-friend-B-packet-no02.pkt
3) Remove a friend on Facebook and store all packets in the file Remove-friend-C-packet-no03.pkt
4) Search a blocked user on Facebook and store all packets in the file Searchblocked-user-D-packet-no05.pkt
5) Send a message to a friend on Facebook and store all packets in the file SendMessage-E-F-packet-no06.pkt

We examine the TCP sequence and acknowledgement numbers in Wireshark’s packet analyzer. We capture HTTP requests to Facebook server. Then we open raw contents of the TCP stream. From the captured filters we can trace the IP addresses corresponding to the request. For instance, when we search a specific user on Facebook, we can identify critical information such as DNS information, geographical location and registration of the captured IP address using the facility of IP2Location1. For instance, IP address 96.17.159.66 appeared in one of captured packets. The geographical location of this IP address is registered in the United States shown in Fig. 9.

For each scenario we documented the packet details such as Protocol, Source IP, Destination IP and Content Type. Table. I represents the gathered information.

1IP2Location is a geo IP solution to help you to identify visitor’s geographical location such as country, region, city, latitude, longitude, ZIP code, time zone, connection speed, ISP and domain name, ID country code, area code, weather station code and name, and mobile carrier information using a proprietary IP address lookup database and technology.
**TABLE I**

<table>
<thead>
<tr>
<th>Session</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Content Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10.136.93.44</td>
<td>69.63.181.11</td>
<td>Cookie, search.php</td>
</tr>
<tr>
<td>B</td>
<td>120.19.137.81</td>
<td>69.63.189.34</td>
<td>POST, Connect.php</td>
</tr>
<tr>
<td>C</td>
<td>10.136.93.44</td>
<td>66.220.146.18</td>
<td>GET, profile.php</td>
</tr>
<tr>
<td>D</td>
<td>120.18.136.164</td>
<td>66.220.138.32</td>
<td>GET, search.php</td>
</tr>
<tr>
<td>E</td>
<td>120.19.137.81</td>
<td>69.63.181.16</td>
<td>GET, search.php</td>
</tr>
<tr>
<td>E-F</td>
<td>120.21.238.11</td>
<td>66.220.149.32</td>
<td>GET, composer.php</td>
</tr>
</tbody>
</table>

**CAPTURED PACKETS CONTAINING INFORMATION OF FACEBOOK SERVERS**

E-F). We have highlighted the HTTP packet which has the source IP address and the destination IP address which is the Facebook server. Depending on the scenario, Facebook uses different servers. From this experiment, there are certain results to be discussed. Facebook lists the IP known to the public as following: 66.220.144.0 – 66.220.169.255, 69.63.176.0 – 69.63.191.255, and 69.171.224.0 – 69.171.255.255. During the experiment we manage to identify a few of them, for example, 66.220.145.41, 66.220.146.18, 66.220.146.54, 66.220.147.33, 66.220.149.25, 66.220.149.32, 66.220.151.78, 66.220.158.32, 69.63.180.47, 69.63.181.11, 69.63.181.12, 69.63.181.16, 69.63.189.16, 69.63.189.34, and 69.63.189.39.

From the derived IP addresses of Facebook there are some addresses which seem to be the most common active IP addresses like 69.63.181.11, 69.63.181.16, 69.63.180.47, 69.220.158.32. Just as the way we visit Facebook home page by entering http://www.facebook.com, when we directly visit the above IP addresses, we could get the Facebook home page. But when we enter http://66.220.145.41 and http://69.63.180.47 we received an error message indicating the address is forbidden to visit. Unlike the other Facebook IP addresses, these two addresses do not provide directory browsing for some reason. Another finding is that the IP address 10.136.93.44 is a private IP address or a virtual private network because certainly Facebook use a virtual private network (VPN) behind the users. The data traveling over the VPN is not generally visible. It could be a hidden server of either Facebook or source IP.

To use WebScarab, we modify the proxy settings of our Firefox browser by including WebScarab as a web proxy. We then create a few Facebook profiles in order to find the blocked users. Firstly, we browse user A’s profile (“Mazz Denizz” in Fig. 10) and block user B (“Din Ruwi” in Fig. 11). After user A blocks the user B, and B should not be able to find user A via Facebook.

We set user C (“Dazz Denizz” in Fig. 12) as a random user who we need to impersonate. We attempt to find whether user B could find user A by using a packet-editing tool like WebScarab.

Now we open the web browser and login to Facebook as user B. After the login session we change the proxy settings and then launch WebScarab. Then we begin to intercept the requests and modify packets before they are sent to a Facebook server. Afterwards, we choose the “Manual Edit” tab and click on both the “Intercept requests” and “Intercept responses”. The request method we would be considering is the GET method. Now go back to Facebook profile and type the person’s name you want to find which is, in this case, User A. When we hit the search button the first thing to pop-up is the Edit Request window (Fig. 13) in WebScarab and now we get the chance to edit the requests manually.

While running WebScarab, we type the name of user C in the search box and the requests and the responses are being intercepted. When the Edit Request window pop-ups we need to replace the name of user C into user A in the text field. Once we are done with replacing the names in the Edit Response window, click “Accept changes” and when we go back to the Facebook page we could see that the user C’s name has changed into user A, which is the blocked user as shown in Fig. 14.

As expected, Facebook disallows users adding the user A. After we click on the “Add as Friend” the displayed name changes back to the original name as shown in Fig. 15. Hence, editing and intercepting requests do not
bypass Facebook’s security mechanism protecting the blocked users.

Hence, we have proved that Facebook implements certain security methods to protect user’s privacy. That is, a user should not be visible by his/her blocked friends. Since, user B could not retrieve any result for the user A, he or she could search for a random user like C and change its value according to the values of A to see if there is any improvement in searching for a blocked user.

VI. CONCLUSIONS

In summary, we provide a method to estimate the efforts required by the law enforcement to investigate crimes and incidents coordinated via Facebook without requesting centralized information from the service provider. We tackle the problem by firstly plotting and examining the UML diagrams to describe the Facebook relationship and the process of establishing a Facebook friendship in order to understand the connectivity in a Facebook group. We derive a probability model to estimate a group size on Facebook. This group size can serve as an indicator of the efforts spent by the law enforcement to investigate group crimes. Our experiments and analysis reveal that block function on Facebook does not allow network security tools to bypass its security.

This study contributes to the theory of random graphs as an initiative to derive an indicative estimation of particular Facebook groups. It is meaningful for law enforcement agencies who reside in different countries where back-end log information cannot be easily obtained from Facebook.

ACKNOWLEDGMENT

The authors would like to thank Dr. Shaun Bangay for proofreading this paper. The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper.

REFERENCES


Dinithi Pallegedara is a current PhD candidate of the School of Information and Business Analytics at the Faculty of Business and Law in Deakin University, Australia. She is a researcher in the areas of Information Security, Social Media, Knowledge and Information Leakage, Corporate governance and Organizational Behavior.
Lei Pan received his PhD degree in computer forensics from Deakin University, Melbourne, Australia, in 2008. He is currently a lecturer at Deakin University. His research interests include security and computer forensics.