Detecting Facebook Synthetic Accounts

Gil Raytan and Ron Galay
Under the supervision and guidance of
Amichai Shulman
# Table of Contents

<table>
<thead>
<tr>
<th>Subject</th>
<th>Number of pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Motivation</td>
<td>3</td>
</tr>
<tr>
<td>Projects Layout</td>
<td>4</td>
</tr>
<tr>
<td>Tools of Trade</td>
<td>5-6</td>
</tr>
<tr>
<td>Methodologies</td>
<td></td>
</tr>
<tr>
<td>• Fan Page Mining</td>
<td>7-9</td>
</tr>
<tr>
<td>• Names Distribution Mining</td>
<td>10</td>
</tr>
<tr>
<td>• Classification Heuristics</td>
<td>11-12</td>
</tr>
<tr>
<td>Code Explanation</td>
<td></td>
</tr>
<tr>
<td>• Information Mining</td>
<td>13-14</td>
</tr>
<tr>
<td>• Classification Heuristics</td>
<td>15</td>
</tr>
<tr>
<td>Results Analysis and Summery</td>
<td>16</td>
</tr>
<tr>
<td>Appendix A - Future Work</td>
<td>17</td>
</tr>
<tr>
<td>Appendix B – Code Snippets</td>
<td>18</td>
</tr>
<tr>
<td>References</td>
<td>19</td>
</tr>
</tbody>
</table>
As our title suggests, our project focus was to find synthetic Facebook accounts. Why are we interested in synthetic accounts in the first place?

Well, as it seems, not only the myths behind Facebook creation have a darker side, but also Facebook's possible usage, as will be demonstrated below.

Facebook can be used as a platform for spreading malware, either by exploiting real security holes (such as persistent cross-site scripting vulnerabilities being published from time to time), such that an attacker can create synthetic accounts for the sole purpose of spreading his malware via those vulnerabilities; or by using the masses as a weapon, for example: creating a malware under the pretenses of a Facebook application, and using many synthetic accounts that "like" the application to lure legitimate users for using it.

Facebook, is currently the biggest social network in the world, which means it has an enormous influence in promoting products or brands, in making an issue trendy.

In this project we are willing to find out, how a mass of users can be a key by its owner to promote its purposes. We are willing to reveal whether there is a use of synthetic account as a mass of fake users, for advancing the targets of a firm, an organization or a person standing behind them.

Those examples demonstrate the central motivation of our project—synthetic accounts in control of an attacker create a "volume" of people, the sheer amount of them is sufficient to cause damage, or being used not for their original targets.
Project layout

Our work process was divided into few sections:
In the first section we were exploring the net and relevant sources for understanding the issue of synthetic users. We were trying to reveal whether there is a big use of synthetic users over Facebook, who is controlling them and what purposes they are using for.

After having in mind some ideas and knowledge about synthetic users, we were exploring the Facebook API. We were reading about methods to access the information stored on Facebook.

The next section was taking our knowledge further to the operative stages. We were trying to use our knowledge and develop some methodologies which will help us discover synthetic account over Facebook. Next we used the methodologies we developed to formulate some methods for finding synthetic users.

The next section was coding our methods: it was divided into two sections. First we wanted to gather a big storage of suspected users. After having a big mass of synthetic accounts we were trying to classify each of them and determine whether its a synthetic one or not. The methodologies we used, and the methods which formulate them will be explained below.

The last section was inspecting the results we received, analyze them and make some conclusions about our research.
Tools of Trade

Facebook Graph API:

The Graph API presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph (for example users, photos etc) and the connections between them (for example friend relationships, shared content, and photo tags).

Every object in the social graph has a unique ID. One can access the properties of an object by requesting https://graph.facebook.com/ID.

The Graph API as such allows you to easily access all public information about an object.

The Graph API uses OAuth 2.0 for authorization. For fetching information one must obtain an access token for a specific user, and then perform authorized requests on behalf of that user by including the access token.

In our project we used an access token that doesn't expire via this site: http://fbrell.com/auth/offline-access-token.

We created a dedicated account for this project - Roy Levy

OAUTH ACCESS_TOKEN = "184484190795|387d40acb600b4421925c827-100001834274618|Yyl5-SYA_CozI5TjZ73SOV69dIs".

Python SDK:

Working on this project we wanted to gather different data on users, such as: individual user information, user wall post, user Fan pages likes and gather information from fan pages news feed as well. For that matter we used the Facebook Graph API.

We connected to the Facebook Graph API via python SDK.

The python SDK is a client library which is designed to support the Facebook Graph API and the official Facebook JavaScript SDK, which is the canonical way to implement Facebook authentication.
**FQL:**

Facebook Query Language (FQL) is a query language that allows querying Facebook user data by using a SQL-style interface and without using the API (usually more efficiently).

```
SELECT [fields] FROM [table] WHERE [conditions] [ORDER BY {field}] [LIMIT {[offset,] row count}].
```


The returned data format can be selected as either XML or JSON.

In this project we used JSON as our return data format.

**MySql Server:**

MySql is a relational database management system that runs as a server providing multi-user access to a number of databases.

We used MySql server to gather our data in Sql tables, and run queries for our classification section of the project.
Fan pages:

The motivation:

After researching the issue of Facebook mal-use we noticed that the vast majority of foul play is directed to advertisement and general spamming.

In order to use Facebook as a marketing tool, the marketer needs to gather a vast dataset of real Facebook accounts (meaning accounts that are used by real people), which he can address.

Although the end result is directed for real users, synthetic users play a big role.

There is a need for creating a mass amount of users in order to use them for specific purposes by their creator: (By having a big base of users, one can expose other real users to content he wants to share)

- A company or organization can "become trendy" by having many users who like their fan pages, comment on their posts or like their videos. For instance, many likes or comments for a specific post of a fan page may expose it to more users by retaining it on the top of the list in the fans news feed.

- Moreover, people use Facebook to promote their own external sites.

  For example, some mal-user can obtain a big fan base, and then increase the traffic on his site by sharing videos or other content on the fan page. That method could be really helpful for increasing a site income – by increasing the number of users exposed to it. Synthetic users can be a great way of virally spreading links.

Some examples:

- "Hi, I want an auto like script which you give a list of fb pages and it can like the posts that fans have written in that page... I can trade Facebook guaranteed fans in you page if you like” written by mfiili, black hat forums, Facebook section. http://www.blackhatworld.com/blackhat-seo/facebook/253577-i-want-pages-posts-auto-liker.html

- "Hi guys, I am searching for program for Facebook mass add friend and wall posts, the only working tool that I know is Facebook friend adder pro...can somebody offer less expensive solutions..." written by matrixbgglan, black hat forums, Facebook section. http://www.blackhatworld.com/blackhat-seo/facebook/287841-best-marketing-solution-for-facebook-mass-friends-add.html
Our method:

Using this motivation, we would like to acquire a base of potential synthetic users. Fan pages can be hijacked for spamming, either by posting comments on the wall or a given post, or by posting pictures that contain a spamming description. We planned to examine a fan page as a spammer would, and retrieve any allowed information.

We have collected a database of suspected synthetic users, and then run some "number crunching" algorithm to ultimately decide on their classification. Using the heuristics we will describe below, we tried to detect synthetic users.

Heuristics:

• Go over the 15 most popular fan pages on Facebook (written on fanpages.txt)
  ◦ For each fan page, gather all the relevant information: wall posts, comments on wall posts, likes on wall posts. Comments and likes on the videos and photos posted on the fan page. Each piece of data will be stored on a text file named main_db.txt with the id and name of the published and the data of the post (id, text etc).
  The examined user will stay in our repository of suspected users for further inspections.

• For each user retrieved by the fetching in the first stage, try to access any possible information. Retrieve the user private information: name, gender, location, picture etc. Retrieve the user wall posts, his own statuses and his friends posts on his wall. Try to reach his friends list, and to retrieve the list of pages he likes.

• Build SQL tables to store all the information we retrieved, and then run some queries for the classification section.
**Storage files schema:**

we keep the information we retrieved in the following order:

**Main database:** Includes all the information gathered from the fan pages we explored.

The table consists of the fields: FanPage name, user id, user name, obj_type (type of post is one of the following: Wall post, Like, Comment), obj_text (if exists), obj_id (does not exist for Like).

The table represents all the suspected users we explored after retrieving their posts data from the fan pages.

**Users tables:** The tables includes all the relevant information about a user which is accessible and which was valuable for the research.

We divided it into two tables, users_post_table and users_like_table which keeps the information we retrieved about the suspected users.
Names Distribution:
The Motivation:

We followed this simple premise: USA is the bases of Internet technology. Looking at Facebook as a test-case, we predict a large amount of Facebook users to originate from the USA. Taking into consideration the findings regarding the mal-usage of Facebook, most likely synthetic accounts will be used as an aid for spamming, thus those accounts should mimic regular accounts as much as possible, i.e. they will have a “normal” name and a picture, at least. The “normal” name of a user is a pivotal idea in our attack. A “normal” name is a name that is not rare. Moreover as we have seen, automatic account creators use popular real names for their fake accounts. That’s why we thought we could gather a large base of synthetic users by searching for popular names.

Under this assumption, we decided to examine the distribution of names in the USA population and to check if it fits the distribution among Facebook users. For this research we used data from http://www.namestatistics.com/
If the distribution among Facebook users and among the population in the USA is not correlated, we examined and retrieve information about this users.
So in theory we have a way to map the “black Facebook” - those accounts that canceled the option to be search-able.

Unfortunately this method failed. We could not retrieve information about users who are not in the directory by searching for their names. Otherwise, we could get search them by their Ids. Moreover the distribution in USA and among Facebook users was almost the same, therefore we decided to fetch names that are popular in the Facebook directory.

Our method:

we used the FQL to retrieve information about users by their name (users which are in the directory). We used the Facebook directory of popular names and retrieve information about users named with one of the 10 most popular names over Facebook (the Facebook directory and US population statistics were almost similarly). After fetching these ids, we used the user information harvest (as we did in the previous method) to retrieve users information.
This way we achieved another big storage of suspected synthetic users.
**Classification heuristic:**

This is the algorithm we intended to use to classify whether a user is synthetic or not based on the information we gather in the last stages of the heuristic. (later we will explain why some of the steps did not work).

The algorithm uses the storage we gathered in the previous methods.

After researching the issue of synthetic account we wrote this heuristic to detect a fake user.

**The algorithm is consist of the following steps:**

- checking 10 different measurement to determine if the user is synthetic or not.

**The measurements are:**

1. Inspecting the percentage and number of male and female friends in the user friends list. If the list is consists of more than 90% friends of the same gender or more than 2500 friends then the suspected user failed this test.

2. Inspecting the number of friends which are kept in the suspected users too. If a user has more than 20 suspected friends then the test fails.

3. Exploring the user wall if available to check which kind of traffic he has. We are looking for suspicious actions, for example doing the same action again and again, too much operations, or almost not at all.

4. Inspecting suspected users pictures. The test fails if two users has the same picture.

5. Observing the user name. If it is ranked top 3 in the popular distribution names as written in the document attached then the test fails. We rely on the assumptions that many synthetic users has a popular American name as our research shows.

6. Identical posts: we exploring all the posts of the suspected user and check if there are at least two identical posts. If exists, then the user fails the test.

7. number of posts: we are exploring all the posts by a user. If he posts more than 25 posts among the posts we scanned, then he fails the test.

8. common spam words: exploring common words spammers use, as indicates in the common spam words (a file we wrote after researching for common words spammers usually use). If he usually use words from the list, the test fails.
9. Inspecting again the posts on a certain fan page an hour later, and checking whether the post was deleted or not. If the post was deleted then the suspected user fails the test.

10. Inspecting how many pages the user likes, if the user like tremendous number of pages, he fails the test.

A suspected user is considered as a synthetic user if he fails the tests 3, 4, 6, 8, 9 or fails at least 2 of the tests: 1, 2, 5, 7, 10. After conducting a comprehensive research and checking a large sum of users manually we figured out that a user is synthetic if he fails the test specified. If the user fails 1, 2, 5, 7, 10, he is synthetic in most cases and therefore we decided to consider him synthetic if he fails at least two to ensure we won’t “convict” regular users.
As we mentioned in our opening description of the project, we first “harvest” the data and then perform heuristics classifications. The method we chose of obtaining the data consists of two stages: (we here by describe in general lines our algorithm for mining, later on we will elaborate and describe our full algorithm)

1. First, we save the mined data in a regular file.
2. Later, we traverse the regular stored text and save it in a MySql dedicated server.

Our heuristics of classification are various sql queries performed on our dedicated server.

For the data mining we used the facebook python SDK (https://github.com/facebook/python-sdk) and at times constructed manual FQL queries. Our sql queries were written in pyhton (we used this library as our sql driver - http://sourceforge.net/projects/mysql-python/).

We designed our code with scalability and ease of testability in mind1. As mentioned in the Methodologies section we had two algorithms in mind. One of them being “Fan Page Mining” and the other “Names Distribution Mining”. In the following paragraph we will describe the “Fan Page Mining” algorithm and the “Names Distribution Mining” algorithm.

All of our source code is supplied, and relevant snippets can be found at Appendix B
Information Mining

Fan Page Mining:

In the “Fan Page Mining” we decided to centralize the listing of the fanpages we traverse, by creating a special file containing them (we named the file fanpages). For testing reasons we enabled the option to comment out a fanpage, i.e. we skip the mining of that page – this is achieved by typing # at the beginning of the line.

Our code for handling the actual mining is split between two files spammers_search and data_retrieval, each harvests a fanpage and a user details respectively.

When spammers_search is activated it creates our textual databases: main_db, users_post_table, and users_like_table.

To create our actual sql tables we activated the db_creation script, and afterwards, for our final classification we run the queries script.

We also had a utility file that centralized the use of common data and behavior, originally named util.

spammers_search:

We shall now explain with more detail the workings of spammers_search script.

In get_info we collect the pages feed, photos and videos. For each queried element we obtain through the graph api the comments and likes made on it (this is done by the use of query_elements. In spammers_search we store the likes and comments in main_db. For each record we store we call the data_retrieval script to collect information on each user that made the comment or the like.

data_retrieval:

The main purpose of the script is to obtain information on a specific facebook user. We try to obtain the users wall and his likes. We save this information in the users_post_table and users_like_table respectively.

FQL – fetch:

used to retrieve the ids of users with popular names. Generated a file named ALGO2. Then we used the data_retrival algorithm to fetch information about the users
Comparison with the theoretic algorithm:
At this point, one might ask why did we choose to harvest only those types of data? Well, there are a few factors here at play. One of them (and the most important) is that we are bounded by the facebook API – we actually traversed almost all of the relevant information regarding our attack vector, the missing data that we didn't consider is the users profile pictures. Here comes to play our second factor – usability. In our experimentations (with the profile pictures) we mined roughly 4mb of data to 100 users. Since we had in mind a scalable algorithm we sadly had to give up that bit of information.

Classification Heuristics

These heuristics are used to classify a given set of users. In other words, given any set of users (mined via our first algorithm or our second one) we are able to find our suspected spammers.

queries:

Our theoretical classification differs from our concrete implementation due to the fact that facebook api doesn't allow for us to retrieve the friend list or any other kind of friend information. Other information wasn't queried due to performance reasons, as mentioned above. What we did manage to achieve are those criteria:

1. The amount of users from our main_db that are classified as spammers (we searched the text of each post and comment for spamm strings)
2. The amount of users from our post_db that are classified as spammers.
3. Users with vast amount of likes.
4. Amount of identical posts or comments from our main_db
5. Amount of identical posts from our post_db.
Results analysis and summary

Our goal was to offer a programmatic way for detecting synthetic users. We found that we are able to locate a certain type of synthetic users (and to the best of our knowledge, the most common type of synthetic users) – spammers.

In the table below we will show some interesting results according to our classification criteria:
( The rows correspond to the criteria and the columns to the results)
We accumulated 11779 unique users and were able to access the profile of 782 of them.

<table>
<thead>
<tr>
<th></th>
<th>2012 users were suspected of being spammers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>989 users were suspected of being spammers.</td>
</tr>
<tr>
<td>3</td>
<td>We made a threshold of 150 likes. We found 9 such users, one of them has as many as 3654 pages he liked!</td>
</tr>
<tr>
<td>4</td>
<td>We found 1762 users who had the same comment repeat multiple times.</td>
</tr>
<tr>
<td>5</td>
<td>We found 757 users who had the same comment repeat multiple times.</td>
</tr>
</tbody>
</table>

False Positives:
We cannot provide a concrete value of false-positives since we cannot inspect each users inspected of being a spammer. We instead offer a programmatic way to be more convinced:

For each of our criteria we supplied the relevant users “framed” by it. One can cross-check between the various criteria and check if a suspected user is also suspected by other measures. This will raise the probability that the user is indeed a spammer.
Appendix A – Future Work

• **Inspecting user photos:**
  As we mentioned before, retrieving user photos takes an enormous memory space, and thus require very large computation power that we did not have.
  (100 users pictures weight 4 MB).
  We think that this kind of classification could be very useful and should be implemented and future work. Two users sharing duplicated photo are very likely to synthetic. This classification could be also combined with a search on Google photos (or any other engine) to check for photos which are not authentic.

• **Code Optimizations:**
  The code we wrote was meant to be working good and fast, however running it on such a large number of users, retrieving all their information and write it into files (and then into the database) took us tens of hours.
  Trying to write a concurrent code, which retrieve several pieces of information simultaneously could turn to be very useful.

• **Facebook social gaming Applications**
  The application on Facebook could be a source for finding synthetic users on Facebook as well. We read an article named “Ghost-busting Facebook: detecting and characterizing phantom profiles on social networks”. We think this could be very interesting inspection for future work.
Appendix B – Code snippets

We show some bits of code relevant to understanding of our implementation:

Snippet of code that is used to traverse the fan pages:

```python
obj_list = util.GRAPH.get_connections(fan_page, connection)
for obj in obj_list['data']:
    obj_id = obj['id']

    # getting all the comments for a given fan_page
    comments = util.GRAPH.get_connections(obj_id, "comments")
    # printing each comment in the format of the specified scheme
    print_comments(fan_page, comments)

    # getting all the likes for a given post
    likes = util.GRAPH.get_connections(obj_id, "likes")
    # printing each like in the format of the specified scheme
    print_likes(fan_page, likes)
```

Snippet of code that is used to traverse the user profile:

```python
# retrieving the user wall feed
wall_feed = util.GRAPH.get_connections(usr_id, "feed")
posts = []
for post_list in wall_feed['data']:
    text = ""
    try:
        text = post_list['message']
    except KeyError:
        pass
    posts = posts + [text]

    # retrieving the user likes
    likes_list = util.GRAPH.get_connections(usr_id, "likes")
    likes = []
    for like in likes_list['data']:
        likes = likes + [like['id']]
```

Some query example:

```python
def same_records(db_name, text):
    ... 
    exe_str = "SELECT user_id, COUNT({0}) AS txt FROM {1} GROUP BY(user_id) HAVING txt > 1".format(text, 'SECURITY.' + db_name)
    cursor.execute(exe_str)
    rows = cursor.fetchall()
    dic = {}
    for row in rows:
        dic[row[0]] = row[1]
    return dic
```
References

Articles and essays:

• “Securing Facebook Application with SecureSphere” by Nitzan Niv, Imperva.
• “Defacing Facebook: A security Case Study” by Adrienne Felt, university of Virginia.
• "Prying data out of social networks” by Joseph Bonneau, Jonathan Anderson and George Danezig.
• "Ghost-busting Facebook: Detecting and characterizing phantom profiles in online social gaming applications” by A.Nazir S.Raza C.Chua, B.Schipper, university of california – davis.

Facebook API:

• Facebook directory: http://www.facebook.com/directory
• FQL section: https://developers.facebook.com/docs/reference/fql/
• Graph API: http://developers.facebook.com/docs/reference/api/
• Facebook developers forum: http://forum.developers.facebook.net/

Relevant web sites:

• Getting an access token: http://benbiddington.wordpress.com/2010/04/23/facebook-graph-api-getting-access-tokens/
• Black hat forums, Facebook section (including):
  ◦ http://www.blackhatworld.com/blackhat-seo/facebook/247895-how-would-i-confirm-phone-number-fake-fb-account-without-buying-etc.html
• Facebook articles: http://www.allfacebook.com/, http://www.insidefacebook.com/
• Distribution of names in USA: http://www.namestatistics.com/
• MySql site: http://mysql.com/