Online Social Networks and Media 6 (2018) 58–68

Contents lists available at ScienceDirect

Online Social Networks and Media

journal homepage: www.elsevier.com/locate/osnem

OSSINT - Open Source Social Network Intelligence
An efficient and effective way to uncover "private" information in OSN profiles

Giuseppe Cascavilla[a,*], Filipe Beato[b], Andrea Burattin[c], Mauro Conti[d], Luigi Vincenzo Mancini[a]

[a] Sapienza, University of Rome, Italy
[b] ESAT/COSIC – KU Leuven and iMinds, Belgium
[c] DTU Compute, Software Engineering, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark
[d] University of Padua, Italy

A R T I C L E I N F O

Article history:
Received 11 January 2018
Revised 27 March 2018
Accepted 16 April 2018
Available online 21 May 2018

A B S T R A C T

Online Social Networks (OSNs), such as Facebook, provide users with tools to share information along with a set of privacy controls preferences to regulate the spread of information. Current privacy controls are efficient to protect content data. However, the complexity of tuning them undermine their efficiency when shielding contextual information (such as the social network structure) that many users believe being kept private.

In this paper, we demonstrate the extent of the problem of information leakage in Facebook. In particular, we show the possibility of inferring, from the network "surrounding" a victim user, some information that the victim set as hidden. We developed a system, named OSSINT (**O**pen **S**ource **S**ocial Network **INT**elligence), on top of our previous tool SocialSpy, that can infer hidden information of a victim profile and retrieve private information from public one. OSSINT retrieves the friendship network of a victim and shows how it is possible to infer additional private information (e.g., personal user preferences and hobbies). Our proposed system OSSINT goes extra mile about the network topology information, i.e., predicting new friendships using the victim's friends of friends network (2-hop of distance from the victim profile), and hence possibly deduce private information of the full Facebook network. OSSINT correctly improved the previous results of SocialSpy predicting an average of 11 new friendships with peaks of 20 new friends. Moreover, OSSINT, for the considered victim profiles demonstrated how it is possible to infer real-life information such as current city, hometown, university, supposed being private.

1. Introduction

Online Social Networks (OSNs) are popular web applications that allow users to build connections, establish relationships, and exchange information over the Internet. At the same time, OSNs hold treasure troves of data that are "insufficiently" protected by default privacy preferences, by generally applying access control rules to content or users. Moreover, privacy preferences are by default hard to use and do not correctly reflect the intentions of users [1,2], which may lead to leakage of information to a broader audience. The privacy issues on OSNs has been a topic of interest within the research community demonstrated by several studies [3–8]. Even though the offered privacy controls are somehow effective to protect the data shared, they remain ineffective when safeguarding contextual information (such as the social network structure).

To analyze the leakage of information in OSNs, such as Facebook, we propose the use of Open Source INTelligence (OSINT) techniques to extract and infer information from publicly available data sources [9,10]. In particular, we aim at extracting publicly available data from Facebook and infer information that is set by users to private through privacy settings rules. To demonstrate the issues mentioned above, we set up two main targets:

- **Q1**: Would it be possible (and if so, to which extent) to reconstruct personal and supposed hidden friends list?
- **Q2**: Is it possible (and if so, to which extent) to infer personal private information like work, education, hometown, current city of a victim user from his social network (friends and friends of friends)?

* Corresponding author.
E-mail address: cascavilla@di.uniroma1.it (G. Cascavilla).

https://doi.org/10.1016/j.osnem.2018.04.003
2468-6964/© 2018 Elsevier B.V. All rights reserved.

To respond to our research questions, we built a system, named OSSINT (Open Source Social Network Intelligence), on top of our previously developed SocialSpy [11]. SocialSpy exploits the Mutual Content Page (MCP) [12] and rebuilds the friends list of a victim user, set and hence thought as private. SocialSpy compared to OSSINT has three main limitations: it is not able to go an extra mile to find friends at two-hop of distance, it is not able to infer real-life information, it is not able to depict the personal network of a victim user. OSSINT receives as input the list of friends, also called *Friends Found* list (the SocialSpy output), retrieved by SocialSpy from a victim user. Through the MCP, OSSINT retrieves the common friends between the owner of the friends list (victim user) and all the IDs from its *Friends Found* list. OSSINT improves the previous results of SocialSpy by predicting multiple-hop friendships (link prediction), such as 2-hop connections (friends-of-friends). Besides, OSSINT manages to reconstruct the friendship graph of a victim user along with the importance weight of friends. Hence, it is possible to use the surrounding network (2-hop network composed of friends and friends-of-friends) of a victim profile to infer extra personal information that is supposedly hidden by the privacy preferences. Finally, we underline that our system, OSSINT, does not exploit any Facebook system flaw like those in [13,14] to retrieve victim's information.

Contribution. The contribution of this work is manifold. First, we demonstrate that Open Source Intelligence (OSINT), applied to OSNs, allows retrieving a significant amount of information that users consider, set, and believe is kept private to any prying eyes or third parties. Second, we are capable of rebuilding the friendship graph of a victim user. Hence, we leverage the friendship graph to evaluate the weight of each friendship based on the number of shared friends. Moreover, we extend our information retrieval to the multiple-hop connections. Finally, we exhibit the possibility of rebuilding other private attributes from the 2-hop network, such as personal information. On all our testing cases, OSSINT, correctly improved the results of SocialSpy finding a more extended set of friends by link prediction applied to the 2-hop users. With an average of 11 new friendships and peaks of 20 new friends found from the 2-hop network, OSSINT demonstrated the feasibility and the correctness of our assumption. Moreover, for all the victim profiles, OSSINT showed how it is possible to infer real-life information supposed being private.

Organization. The remaining part of the paper is organized as follows. In Section 2 we give a formalization of Facebook and our system OSSINT. In Section 3 we give an overview of OSSINT, showing the interaction between *SocialSpy* and OSSINT, our proposed system. In Section 4 we give all the technical details of OSSINT, how it works, what are the tools and the techniques used, an example of the output. In Section 5 we present our experimental settings and discuss the results. Finally, Section 6 draws some conclusions, limitations and future works. In Section 7 we review state of the art.

2. System model

In this section, we formalize our "system model", i.e., the environment where our tool could be immersed, to retrieve the information. We assume Facebook as a representative implementation of such environment.

Facebook is composed of different entities. All these entities, together, give the possibility to the final user to perform different actions into such environment. The entities we consider are: *pages*, *users*, *groups* and *pictures*. *Users* are allowed to perform some actions: become "friend" of another user, "like" a *page* (and revoke the "like"), "join" a *group* (and leave the group), and "like" or "comment" pictures (and revoke the "like" or delete the "comment"). Instead, *pages*, *groups* and *pictures* are "passive" entities (i.e., they are managed by *users*). The set of pages a user likes can be interpreted as the *tastes* of that user. Usually *pages* enable public figures (such as companies, organizations, or celebrities) to create a presence on Facebook [15]. *Groups* on Facebook are "places" where people can share and discuss their common interests and express their opinion on common causes, issues or activities to organize [15]. A group is not always public. Tuning group's privacy rules, it is possible to set it as public (accessible and searchable to all users in Facebook), private (accessible only if invited; searchable to all users in Facebook) or hidden (accessible only if invited; not searchable to anybody in Facebook). *Pictures* are usually uploaded by users. The *Pictures* environment can be divided into three main categories. The personal pictures: uploaded directly by users. Pictures where a user is tagged and usually uploaded by other users (generally friends). Cover pictures are the larger photo at the top of your profile, above the profile picture and it is always public. All these different types of pictures, the ease of sharing and the need of feeling good by receiving likes [16] make challenging to keep the privacy under control. Furthermore, Facebook gives to its users the possibility to make the profile much more detailed filling fields regarding personal hometown rather than the attended university or the current city.

More formally, the portion of Facebook that we are going to use in the rest of this paper can be formalized as the tuple: $Facebook = (\mathbb{U}, \mathbb{P}, \mathbb{G}, \mathbb{I}, \mathbb{C}, \mathbb{S})$. Specifically, in this notation, we have that:

- $\mathbb{U}$ is the set of users. A user $u \in \mathbb{U}$ represents a person. Each person can "like" a page p, join a group g, leave comments into a page, request friendships to other users (accept friendship from other users), upload pictures into his profile pages.
- $\mathbb{P}$ is the set of pages. A page $p \in \mathbb{P}$ is something related to the tastes of a user, i.e., what a user might like.
- $\mathbb{G} = (G', n)$ is the multiset that represents groups. $G' \subseteq \mathbb{U}$ and $n : G' \to \mathbb{N}_{\geq 1}$ is the multiplicity function indicating the number of groups with the same set of users (please note that the same set of users may appear several times). $\mathbb{G}$ represents all the groups on Facebook (please note the same set of users may appear several times). A group, from the Facebook point of view, is a place where a user can promote, share and discuss relevant topics.
- $\mathbb{I}$ is the set of pictures. Every picture $i \in \mathbb{I}$ can receive one or more "likes" and one or more "comments" from a user $u \in \mathbb{U}$. Therefore, it is possible to consider a picture as the pair $i = (U_i^l, U_i^c)$. Where $U_i^l \subseteq \mathcal{P}(\mathbb{U})$ is the set of users that liked i, and $U_i^c \subseteq \mathcal{P}(\mathbb{U})$ is the set of users that commented on i.
- $\mathbb{C}$ is the set of cities. An hometown $h \in \mathbb{C}$ represents the city where a user u was born. Differently, $cc \in \mathbb{C}$ represents the current city where a user u currently lives.
- $\mathbb{S}$ is the set of schools. The education field $e \in \mathbb{S}$ represents the attended university of the user u. Differently $hs \in \mathbb{S}$ represents the high schools attended by the user u.

Within our model, a user u is defined as the tuple

$$u = (Personal, U, P, G, I, C, S),$$

where: *Personal* is the set of "personal" information (such as the name, the family name, the age), $U \subseteq \mathbb{U}$ is the set of friends of u; $L_1, \ldots, L_m \in \mathbb{U}$ are corresponding friends list of $u_1, \ldots, u_m$; $P \subseteq \mathbb{P}$ is set of pages u likes; $G \subseteq \mathbb{G}$ is the set of groups u belongs to; and $I \subseteq \mathbb{I}$ is the set of personal pictures (pictures that u uploaded into the social network); $C \subseteq \mathbb{C}$ is the set of cities where u was born and where is currently living; $S \subseteq \mathbb{S}$ is the set of schools where the user u attended the university classes and the high schools lectures. Given a user u we can extract each component using a pro-

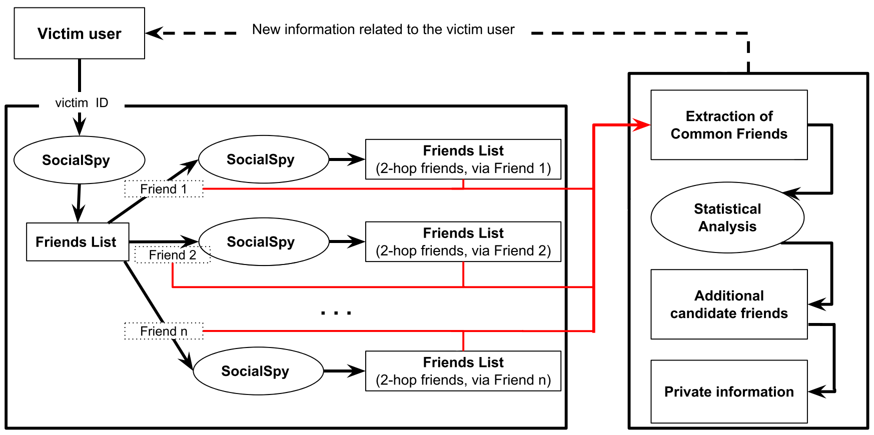

Victim user
New information related to the victim user
victim ID
SocialSpy
Friends List
Friend 1
Friend 2
Friend n
SocialSpy
SocialSpy
SocialSpy
Friends List (2-hop friends, via Friend 1)
Friends List (2-hop friends, via Friend 2)
Friends List (2-hop friends, via Friend n)
Extraction of Common Friends
Statistical Analysis
Additional candidate friends
Private information

Fig. 1. Flow chart and interaction of *SocialSpy* and OSSINT.

jection operator π. For example, $\pi_C(u) = cc$ the current city of our victim user, rather than $\pi_E(u) = e$ for the attended university of our victim and, lastly, $\pi_H(u) = h$ for the hometown of our victim. Due to all these interacting entities, and their complex set of privacy settings, we truly believe that there is a real possibility of a leak of information out of Facebook.

3. OSSINT: retrieving private information

This work aims to show how the friends list can be used as a master key to retrieve private information of a victim thought and hence set as private. Fig. 1 gives a graphical representation of our approach and the involved entities.

The system is composed of two main parts. The first and central part improves the result of our previous tool SocialSpy [11]. Then, differently, from the previous tool, OSSINT focuses its analysis at 2-hop of distance. To achieve this, we run SocialSpy on the victim user ID, and all the friends returned out from the first run. Therefore, if SocialSpy finds n friends for a victim, we will rerun it against all of them, resulting in a total of $n + 1$ runs of SocialSpy.

Within a big pool of 2-hop friends (i.e., friends of friends of our victim), OSSINT finds new connection and new friends of our victim user. Moreover, exploiting all the found links, the system can retrieve information regarding the current city, the hometown and the education of a victim profile. The retrieval of personal information constitutes the second part of our system. As mentioned in our previous work [11], we decided to have as a target the friends list of a victim user because we believe this is the main features to use as starting point to retrieve other personal information. With OSSINT we can finally demonstrate the validity of our assumptions and show through our testings the feasibility of our theory.

To retrieve the friends list, we based OSSINT on the result of our previous work in [11]. SocialSpy exploits different strategies to fetch this information. However, as we showed in our previous work, the most effective way is to exploit the victim's pictures (I, according to our system model). These (or a subset of them), many times, are left publicly available and the strategy we proposed exploits the likes and the comments that each public picture received. In particular, given a picture $i \in I$, belonging to the victim v, we can retrieve all the users that liked or commented the picture. Each of them, using Mutual Content Page, is checked for his friendship with v. Algorithm 1 illustrates the steps of SocialSpy to retrieve the friends list of a victim user fetching the information from publicly available pictures.

We used the same strategy in OSSINT, and we decided to apply it not only at 1-hop of distance from our victim profile but also at 2-hop (friends-of-friends) of distance. The idea is to improve the pool of IDs on which find new possible friends. Moreover, the computational complexity of Algorithm 1 is $O(n)$ where n is the number of images.

Algorithm 1: SocialSpy (using "Strategy 4" (*S4*), Likes and Comments).

```
Data: Victim user $v$
Result: Set of friends of $v$
1 $I \leftarrow$ set of public images of $v$
2 $CandidateFriends \leftarrow \emptyset$
3 foreach $i \in I$ do
      /* Add candidate friends set all users that liked or
         commented the image                                  */
4     $CandidateFriends \leftarrow CandidateFriends \cup U_i^l \cup U_i^c$
5 $FriendsFound \leftarrow \emptyset$
6 foreach $c \in CandidateFriends$ do
      /* Check friendship with Mutual Content Page             */
7     if $AreFriends(c, v)$ then
8         $FriendsFound \leftarrow FriendsFound \cup \{c\}$
9 return $FriendsFound$
```

Algorithm 2 shows how OSSINT extracts 2-hop IDs using the Mutual Content Page (MCP).

Algorithm 2 takes v as input, and for each ID from the friends list of v (line 2) applies *SocialSpy* (line 4) to retrieve the friends list of all the IDs from the friends list of v. Then, it extracts all the common friends between u_i and u_j. u_i belongs to the friends list of v meanwhile u_j belongs to the friends list of u_i (line 5). The computational complexity of Algorithm 2 is $O(n^m)$ where n is the number of friends of v, and m is the maximum number of 2-hop friends. Please note that 2-hop friends are not necessarily friends of our victim user, as reported in the example in Fig. 2.

Fig. 2 shows the result after the execution of *SocialSpy* on victim v and on *1-hop_ID-x*. In order to enhance the results of *SocialSpy*, and then the possibilities to find new friendships at 2-hop, we apply Algorithm 2 on *1-hop_ID-x* and *2-hop_ID-y*. The result of the execution of Algorithm 2 is depicted Fig. 3.

As above mentioned, Fig. 3 shows the output after the execution of *SocialSpy* and after the execution of Algorithm 2 on a victim ID. OSSINT produces then: *(i)* a set of friends of v; *(ii)* a set of friends for the friends of v; *(iii)* a map data structure where the key of the map is composed by 1-hop friends of v and 2-hop IDs friend of the 1-hop ID. Each key is associated with the list of mu-

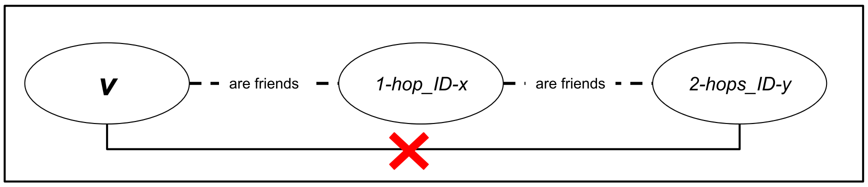


Fig. 2. Example of 2-hop friends after the execution of *SocialSpy*. *2-hop_ID-y* not friend of the victim *v*.

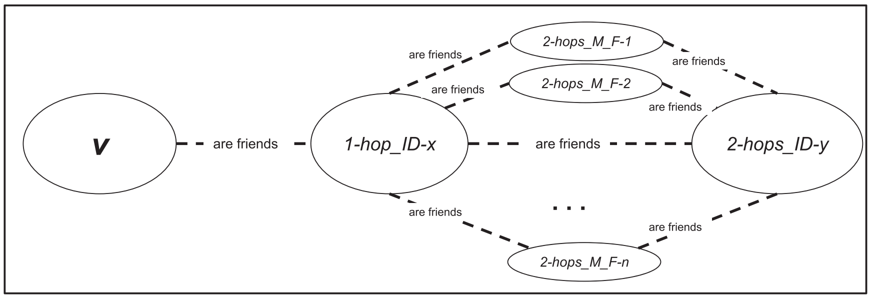


Fig. 3. Example of 2-hop candidate friends after the execution of Alg. 2 on *v*.

Algorithm 2: Set of potentials friends at 2-hop of distance from *v*.

```
Data: Victim user $v$
Result: The map of each tuple $(u_i, u_j)$ and its common friends.
1 $List\_dicts = []$
2 foreach $u_i \in socialspy(v)$ do
3     $dict\_temp = \{\}$
4     foreach $u_j \in socialspy(u_i)$ do
          /* M_F is the function in charge to extract
             mutual friends using the MCP                  */
5         $dict\_temp[(u_i, u_j)] = M\_F(u_i, u_j)$
6     $List\_dicts.append(dict\_temp)$
7 return $List\_dicts$
```

Algorithm 3: Production of the friendship graph *G* of *v*.

```
Data: Victim user $v$, $List\_dicts$ output of Algorithm 2
Result: $G$ Friendship graph of $v$ at 2-hop
1  $V = [v]$
2  $E = []$
3  foreach $u_i \in socialspy(v)$ do
4      $V.append(u_i)$
5      $E.append((v, u_i))$
6      foreach $u_j \in socialspy(u_i)|\{v\}$ do
7          if $(u_j \notin V)$ then
8              $V.append(u_j)$
9              $E.append((u_i, u_j))$
10             for $u_k \in M\_F(u_i, u_j)$ do
11                 if $(u_k \notin V)$ then
12                     $V.append(u_k)$
13                     $E.append(u_i, u_k)$
14                     $E.append(u_k, u_j)$
15 return $G = (V, E)$
```

tual friends. All the edges from Figs. 2 and 3 are dashed because it is now task of Algorithm 3 to connect all of them. Once all the data are connected, Algorithm 3 will produce the friendship graph.

Algoritm 3 is used to generate the friendship graph related to our victim user. It takes as input all the outputs generated from the previous algorithms. Starting with the set of friends of *v*, it connects all the IDs (line 3 to line 5). Once the 1-hop friends are connected, it is now the turn to add the 2-hop IDs. Lines 6, 8 and 9 take care of this step, meanwhile line 7 removes the duplicates. Once the graph *G* results connected at 2-hop, the algorithm uses the common friends to insert new edges among the already connected IDs (line 10 to line 14). The output of Algorithm 3 is the friendship graph of the victim *v* at 2-hop. The computational complexity of Algorithm 3 is $O(n^{m*n})$ where *n* is the number of friends of *v*, and *m* is the maximum number of 2-hop friends. Algorithm 4 shows the steps to extract information and compute the rate of each info from each feature (education, hometown and current_city) retrieved from the friends of *v*. To compute the rate of each element from each feature, Algorithm 4 counts how many time the same information appears into the feature taken into account over the number of friends of *v*. Where $\pi_E(u_i)$, $\pi_H(u_i)$, $\pi_{CC}(u_i)$ are the projections of Education, Hometown and Current_City.

Algorithm 4 computes the percentages of each feature education, hometown, current_city extracted from the 1-hop users. Line 6 verifies that the information is not already into *edu*{}, if not, it adds the new feature into *edu*{}. If the feature is present, Algorithm 4 increments the counter for the same feature. The same action is also used for the other features, hometown and current_city. The computational complexity of Algorithm 4 is $O(n)$ where *n* is the number of friends of *v*.

Algorithm 5 shows the steps to score the likelihood of the 2-hop IDs. Moreover in Algorithm 5 we use the function showed in Algorithm 4. Indeed this algorithm first extracts the information regarding education, hometown and current_city of the 2-hop users, then score the likelihood based on the result given by Algorithm 4 applied on *socialspy*(*v*) users. To have more reliable likelihood score, Algorithm 5 takes into account also the number of edges (common friends) among u_j (2-hop user) and *v*.

Algorithm 5 takes a 2-hop user (line 6) and using the function $proc\_info(u)$ extracts the value of the feature. On line 8 the feature retrieved from the 1-hop results is education, on line 10 the

Algorithm 4: Algorithm to extract and rate information from the IDs from the friends list of v.

```
Data: Victim user $v$
Result: Information and the related rate
$edu = \{\}$
$hometown = \{\}$
$cur\_city = \{\}$
$M = |socialspy(v)|$
foreach $u_i \in socialspy(v)$ do
    if $(\pi_E(u_i) \notin edu)$ then
        $edu[\pi_E(u_i)] = 1/M$
    else
        $edu[\pi_E(u_i)] = edu[\pi_E(u_i)] + 1/M$
    if $(\pi_H(u_i) \notin hometown)$ then
        $hometown[\pi_H(u_i)] = 1/M$
    else
        $hometown[\pi_H(u_i)] = hometown[\pi_H(u_i)] + 1/M$
    if $(\pi_{CC}(u_i) \notin cur\_city)$ then
        $current\_city[\pi_{CC}(u_i)] = 1/M$
    else
        $current\_city[\pi_{CC}(u_i)] =$
        $current\_city[u_i.current\_city] + 1/M$
return $(edu, hometown, curr\_city)$
```

Algorithm 5: Algorithm to score the likelihood of IDs at 2-hop.

```
Data: Victim user $v$
Result: Score of likelihood of 2-hop users associated to $v$
/* See Algorithm 4 */
$edu, hometown, cur\_city = proc\_info(v)$
$dict\_scores = \{\}$
$score = 0$
$max = 0$
foreach $u_i \in socialspy(v)$ do
    foreach $u_j \in (socialspy(u_i)|socialspy(v))$ do
        if $u_j \notin dict\_scores$ then
            if $(\pi_E(u_j) \in edu)$ then
                $score = score + edu[\pi_E(u_j)]$
                if $(\pi_H(u_j) \in hometown)$ then
                    $score = score + hometown[\pi_H(u_j)]$
                    if $(\pi_{CC}(u_j) \in cur\_city)$ then
                        $score = score + cur\_city[\pi_{CC}(u_j)]$
                        $score = (score)/3$
                        $N = |socialspy(u_j) \cap socailspy(v)|$
                        if $max < N$ then
                            $max = N$
                            $dict\_scores[u_j] = (score, N)$
foreach $u \in dict\_scores$ do
    $dict\_scores[u] =$
    $(dict\_scores[u][0] + (dict\_scores[u][1]/max))/2$
return $dict\_scores$
```

feature is hometown and on line 12 the feature retrieved is current_city. The related score from the 1-hop results is added into the variable *score*. Line 14 compute the average of all the scores previously extracted from the 1-hop users data. Line 15 extracts the number of edges (or common friends) from u_j (2-hop user) and v. Lines 16, 17 and 18 are in charge to find the highest number of shared edges. Based on that all the other values are normalized using the highest one. Lines 19 and 20 determine the final score based on the extracted information and on the number of shared edges.

Once Algorithm 5 returns the *dict_score* we compare each score of each ID with a threshold defined looking at the optimum value among our data. If the score of the ID at 2-hop is above our threshold, we mark the user ID as “FRIEND” if below the ID is marked as “NOT FRIEND”. The computational complexity of Algorithm 5 is $O(n^m)$ where n is the number of friends of v, and m are the 2-hop friends.

4. OSSINT: implementation

OSSINT fills the gap of SocialSpy between the retrieved friends list and the possibility to have a graphical representation of the whole network of a victim user at 2-hop. The OSSINT system receives as input the list of friends from the IDs found after the execution of SocialSpy. The tool, then, iterates on each ID (from the friends list of v) and their list of friends to extract all the common IDs. After the execution of OSSINT on each ID (from v friends list) and their personal friends list, it connects all the IDs at 1-hop with the IDs at 2-hop. The common friends are then used both as edges and also to find new connections among 1-hop and 2-hop IDs.

In particular, the tools used in our system OSSINT are:

(i) Mutual Content Page (MCP), i.e., a page that displays that content two users have in common;
(ii) Selenium Web Driver to browse the Facebook MCP pages;
(iii) Graphviz python library to generate a `.dot` file that represents our output graph;
(iv) Gephi platform for the visualization and the manipulation of the graph.

After the iteration of OSSINT on each 1-hop ID and the related friends list, we have multiple outputs (`.json`) files. In the `.json` output are listed all the common friends among each 1-hop ID and the IDs from the related friends list. OSSINT can then connect v with its 1-hop friends list and all the 1-hop IDs with the IDs at 2-hop using the common friends as edges.

4.1. Graph notation

Table 1 shows the notation used in the next paragraphs. Each colour corresponds to a different type of user. The main actors are the victim, the friends of the victim, and the friends of friends of the victim. Moreover, among them, we highlight the common friends and the most relevant and less relevant 2-hop IDs.

4.2. 1-hop strategy

We are now going to analyze in detail our approach. The first step of OSSINT is to run SocialSpy on a victim user ID. The SocialSpy tool receives as input a victim ID and returns the friends list of the victim.

From the retrieved list of friends, we depict the social graph of our victim ID. Indeed, in case of 1-hop IDs (friends of our victim profile), OSSINT depicts merely the social graph of our victim. In Fig. 4 an example of graph composed by those friends found at 1-hop. The yellow, dark green and blue nodes are our testing nodes and together with the purple nodes represent the 1-hop friends.

Table 1
Notation of the colours used in the graphs.

Table of colours	
🟩	Victim user
🟦 🟩 🟨	Testing IDs and friends of victim user
🟪	Other friends of victim user
🟫	Friends of 🟦 (friends of friends)
🟥	Friends of 🟩 (friends of friends)
🟪	Friends of 🟨 (friends of friends)
🟧	Most relevant 2-hop IDs used to predict new friendships of the victim
⬛	2-hop IDs with only one common friend with the victim (not relevant IDs)
🟦	Common friends between victim user and its testing friends 🟦 🟩 🟨

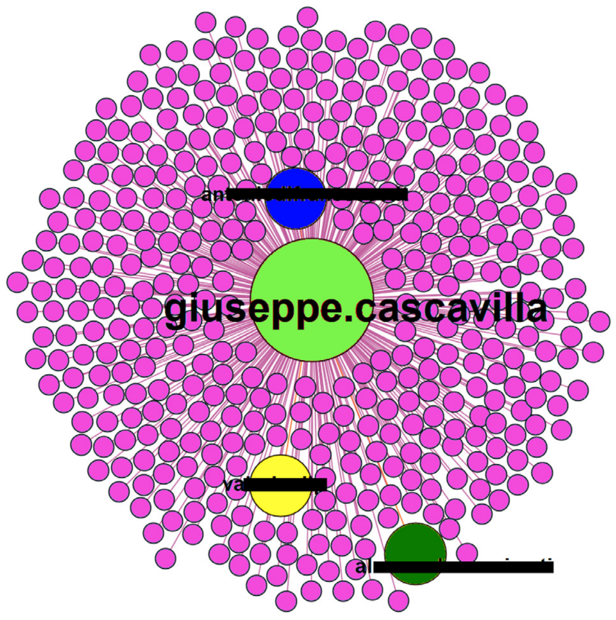


Fig. 4. 1-hop friendship graph after running *SocialSpy*. Green node: victim ID. 1-hop friend IDs: Yellow, Dark Green, Blue and purple nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The green node in the centre of our graph is our main victim profile. All the small nodes (purple nodes) are other friends of our victim ID. Yellow, dark green and blue nodes are our testing IDs and friends of our victim.

Once we have the friends list of our victim user, we run SocialSpy on each user ID from the retrieved list. After this second round of SocialSpy we have, per each friend of our victim, a friend found list and the related network graph.

We can then depict the social graph of our retrieved IDs using OSSINT. In Fig. 5 an example using the testing IDs from Fig. 4.

The three graphs in Fig. 5 depict three different friends network for the three testing IDs from the victim friendship graph (Fig. 4). The dark green, blue and yellow nodes are the "new" victim IDs from the *giuseppe.cascavilla* friends found list. These three victims are friends of our user *giuseppe.cascavilla*, and on them, we run our SocialSpy tool. The node violet, brown and red represent the friend IDs of our victims (2-hop friends) respectively. Moreover the three graphs in Fig 5 give us two different information. Graphs in Fig 5(a) and (b) reaffirm the friendship between our victim user *giuseppe.cascavilla* and his friends found IDs. The Fig. 5(c) shows that our SocialSpy tool "failed" in finding our victim ID *giuseppe.cascavilla*. However since the friendship in a Facebook social network is an undirected edge, we can assert that exists the friendship between *giuseppe.cascavilla* and al*** simply using the data from *giuseppe.cascavilla*.

At this stage, our SocialSpy tool retrieved the lists of friends from our victim ID *giuseppe.cascavilla*. SocialSpy, then, retrieved the lists of friends from the 1-hop IDs (Fig. 5) composed of our testing victims va***, an***, al*** friends of *giuseppe.cascavilla*. We have, then, a bunch of list of friends files from all our target IDs. At this point our SocialSpy tool becomes limited. Actually, with SocialSpy is not possible to link together all the friends list. It is, hence, not possible to build the friendship graph of the victim *giuseppe.cascavilla*. Moreover, is not possible to infer private information from the victim ID, *giuseppe.cascavilla*, because of its privacy settings. Lastly, from our studies, we can assert that all these lists of friends can be linked together using shared friendships as edges and in the next paragraphs we show how we do that.

4.3. 2-hop strategy: proposed system

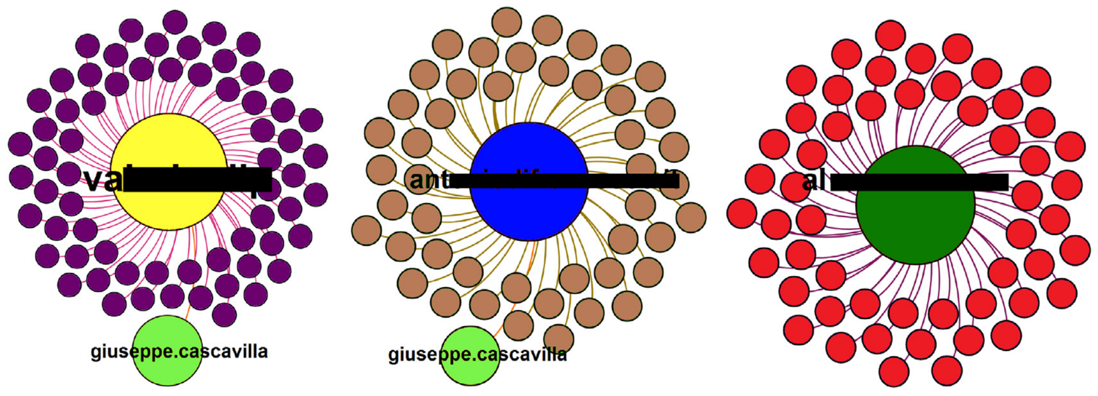
With the lists of friends retrieved by *SocialSpy*, we are now able to use OSSINT to build the friends network of our victim profile.

The main steps of OSSINT can be summarized as follow:

1. extracts all the common friends between each 1-hop ID and the IDs from friends list retrieved by SocialSpy,
2. produces the text version of the final friendships graph where all the IDs are connected based on the friendships,
3. generates a visual representation of the final friendships graph of the victim user at 2-hop of distance
4. infers information regarding school, hometown and current city of a victim user exploiting the publicly available information of the 1-hop IDs,
5. produces the list of possible friends of the victim user found at 2-hop of distance (link prediction among victim user and 2-hop IDs) and based on the information retrieved from the 1-hop IDs.

Using the MCP and the Selenium Web Driver, OSSINT can retrieve all the mutual links (also known as friendship) among the 1-hop IDs and the friends from their friends list.

OSSINT produces a `.dot` file right after finishes collecting all the `.json` files (containing the common friends) among all the 1-hop IDs and the IDs from the 1-hop's friends list. This `.dot` file is the text version of our final graph. Giving to *Gephi* our generated `.dot` file we can have a visualization of our network at 2-hop of distance.



(a) Friendship graph of victim ID **va*****. (b) Friendship graph of victim ID **an*****. (c) Friendship graph of victim ID **al*****.

Fig. 5. Graphs of friendship of three IDs that share the friendship with our victim user.

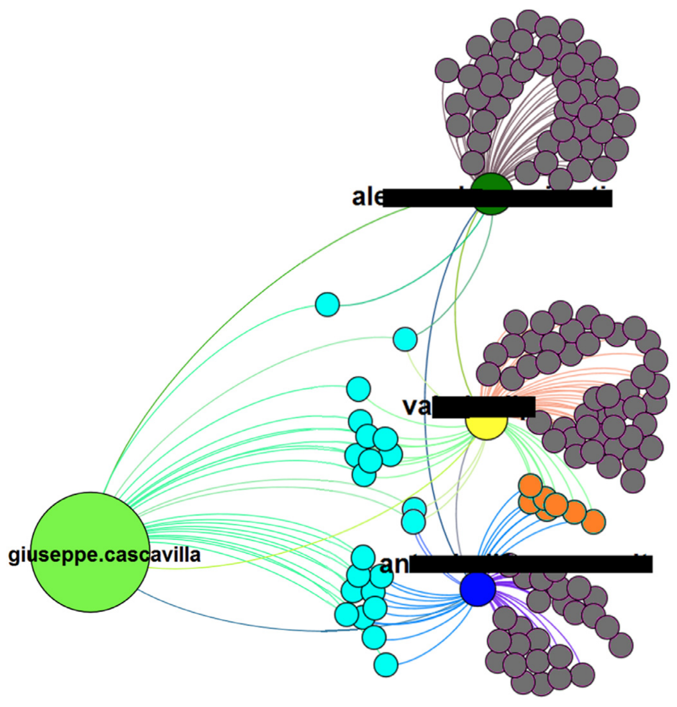


Fig. 6. 2-hop IDs graph after running *SocialSpy* in conjuction with OSSINT.

In Fig. 6 an example using the data from Figs. 4 and 5.

For readability purpose, from Fig. 6 we removed all the purple nodes, friends of our victim ID *giuseppe.cascavilla* (Fig. 4). However, it has to be taken into account that, for each purple node, OSSINT makes the same steps as for the dark green, blue and yellow nodes (our testing nodes). However, we now focus only on our three nodes: dark green, blue and yellow and consider them as the only friends of *giuseppe.cascavilla*. From Fig. 6 we then identify our victim ID (light green node) surrounded by its friends (dark green node, blue node and yellow node). The dark green, blue and yellow nodes are the 1-hop IDs to which we applied SocialSpy to retrieve the 2-hop IDs. The 2-hop IDs are identified by the grey nodes and the orange nodes. The common friends of the victim ID and its friends are the light blue nodes. From our studies, we can assert that the light blue nodes give us the information regarding who are the closest friends of our victim. Indeed, we strongly believe that more are the common friends among two users, stronger is the friendship among the involved users in real life. We will now focus on the 2-hop IDs to find new friends of *giuseppe.cascavilla* (orange nodes and the grey nodes).

4.4. 2-hop strategy: information extraction

The network in Fig. 6 gives us some useful information regarding our victim ID.

1. we have the list of friends of our victim ID *giuseppe.cascavilla*;
2. we can infer who are the closest friends of *giuseppe.cascavilla* based on the number of common IDs between the victim and its friends (higher is the number of common friends, higher is the possibility the two IDs know each other very well in real life);
3. we have the list of users at 2-hop of distance;
4. based on the number of shared friends between the victim and each of the 2-hop IDs, we can remove the “useless” 2-hop users (from Fig. 6 the grey nodes sharing only one edge with only one victim’s friend), keeping the useful ones (from Fig. 6 the orange nodes).

From the list of friends we retrieved and, using the OSINT technique, we extract the information regarding education, hometown and current city, publicly available, from the IDs that share the friendship with *giuseppe.cascavilla* (in our case from dark green, blue and yellow nodes). OSSINT produces an `.xml` file with all the considered information from the friend IDs.

OSSINT processes all the date from the `.xml` files and produces an `.xlsx` file containing one table per each field education, hometown and current_city. Each table contains all the rates regarding the information retrieved from the 1-hop list of IDs. Through this first step, we can infer the personal information of our victim user. Indeed we truly believe that the personal information of the IDs from the friends list of our victim reflect the private information of our victim ID in real life.

4.5. 2-hop strategy: friends extraction

After the steps described in Section 4.3, we now have a huge network, with a lot of IDs at 2-hop of distance and all of them can be possible friends of our victim user. To reduce the possibility of errors, because of the huge amount of IDs, we decided to remove from the network, at 2-hop of distance, all the IDs with

Table 2
Data example from 1-hop user IDs.

Current City	#	% (%)	Homewtown	#	% (%)	Education	#	% (%)
Padua	15	27	Padua	18	13	Padua	39	40
Bologna	5	9	Rome	15	11	Venice	8	10
Paris	2	4	Venice	3	3			
Madrid	1	2						

Table 3
Data example from 2-hop user IDs.

Source ID	Current City	Homewtown	Education	# Shared Edges
2-hop_ID-1	Padua	Padua	Padua	8
2-hop_ID-2	Brussels	Turin	Rome	3
2-hop_ID-3		Venice		10
2-hop_ID-4	Venice		Venice	2

Table 4
Example of scoring based on data from Tables 2 and 3.

Source ID	Information score	Edges score
2-hop_ID-1	0.266	0.8
2-hop_ID-2	0.016	0.3
2-hop_ID-3	0.010	1.0
2-hop_ID-4	0.033	0.2

Table 5
Confusion matrix.

	Not predicted	Predicted
Actually not friend	True negative(TN)	False positive (FP)
Actually friend	False negative (FN)	True positive (TP)

only one edge. Using as an example the graph in Fig. 6, we remove all the grey nodes that share only one friend with our victim ID *giuseppe.cascavilla*. Differently, we keep all the IDs with more than one common friend with *giuseppe.cascavilla* (Fig. 6 the orange nodes). The decision is based on the fact that the higher is the number of common friends shared between the 2-hop ID and the victim ID, the higher is the probability that the victim shares the friendship with the considered 2-hop ID. After removing the one-edge IDs, we apply the OSINT technique on the remaining nodes. We extract then the personal information publicly available from the IDs at 2-hop. As for the 1-hop IDs, also in this case OSSINT produces a `.xml` file with all the personal information regarding our 2-hop IDs. The processing phase of the `.xml` file produces the `.xlsx` file containing one table per each field education, hometown and current_city. However, differently from the previous file, we now use this information to score each user ID. The score is based on how many information fit between the 1-hop IDs statistics and the 2-hop information. In Table 2 a scoring example.

Table 2 shows the rates regarding the information from the user IDs at 1-hop (friends of our victim user *giuseppe.cascavilla*). These percentages will be part of our scoring mechanism. In Table 3 an example of retrieved data from 2-hop IDs.

Table 3 shows the information retrieved from the IDs at 2-hop of distance. The score is estimated based on how much personal information fit with the data from Table 2 and how many common friends are shared with our victim user *giuseppe.cascavilla*. All the data are normalized between 0 and 1. If we suppose that the highest number of shared edges is 10, we normalize the values in column # Shared Edges dividing by ten all the values. In Table 4 a scoring example.

Table 4 shows an example of scoring 2-hop IDs. Information Score is the average score based on the data from Table 2. Edges Score is a score based on the normalization of the value given by the number of shared friends between the 2-hop IDs and the victim user *giuseppe.cascavilla*.

In order to understand if a 2-hop ID can be highlighted as *"FRIEND"* or *"NOT FRIEND"* of our victim ID *giuseppe.cascavilla*, we compare Information Score and Edges Score values with our "Best Value". The "Best Value" scores are respectively the optimum value of Information Score and the optimum value of Edges Score from all our victim IDs from all our data. Thus "Best Value" are two fixed parameters, respectively Best Information Value and Best Edges Value, with which to compare the score of our 2-hop IDs. Lastly, if a 2-hop ID is scored with both the values equal or higher than our "Best Value", the considered ID is highlighted as *"FRIEND"*, otherwise marked as *"NOT FRIEND"*.

5. Evaluation

To evaluate our approach, we conducted several experiments on eight different real profiles of selected volunteers. For our tests, we decided to choose only Facebook profiles with a high level of privacy and with a really low disclosure of information from the personal profile. The choice of using Facebook profiles with high privacy concern as a target is since we truly believe that even with high accuracy in tuning the privacy options, the disclosure of information can happen through the surrounding network (friends and friend-of-friends). All the victim IDs are real profiles. To each of them, we asked to share with us their personal friends list. The personal friends list of our victim profiles has been used as ground truth for our experiments. Thus, to score the precision of our technique, we compared the results of our experiments from each victim ID with the real data from the personal friends list.

To perform our tests, we logged into Facebook using more than twenty different accounts. All of them are part of a network composed by only those accounts used for the experiments. We built a fake network with fake accounts to appear on Facebook as real profiles and avoid to be blocked. The attacker profiles we used does not share any information nor with the victim profile neither with the friends of the victim. As above mentioned, for our experiments we decided to use only users with a high privacy concern. All of them do not show anything about their personal information. The victim profiles have an amount of publicly available pictures between 1 and 13. Most of them are cover pictures. Regarding the friends list, differently from the other information, only one profile out of 8 have a public friends list. However, even if the friends list is publicly available, our system OSSINT does not perform any action on it. Indeed, our system OSSINT, re-builds a list of friends using publicly available information and not considering or retrieving data from a publicly available friends list. The friends list then, from the OSSINT point of view, is always regarded as private.

5.1. Experimental results on friends found at 2-hop

To show the feasibility and effectiveness of our system, we provide a more in-depth analysis that demonstrates that with OSSINT we can identify new friendships at 2-hop of distance. Table 5 depicts an example of the general confusion matrix we used for our outputs.

A confusion matrix [17] contains information about actual and predicted classifications done by a classification system. The

Table 6
Confusion matrix with average results of the experimentation phase.

	Not predicted	Predicted
Actually not friend	253	118
Actually friend	28	11

entries in the confusion matrix have the following meaning in the context of our study:

- **TN** is the number of correct predictions that the 2-hop_ID-# is not friend of v;
- **FP** is the number of incorrect predictions that the 2-hop_ID-# is friend of v;
- **FN** is the number of incorrect predictions that the 2-hop_ID-# is not friend of v;
- **TP** is the number of correct predictions that the 2-hop_ID-# is friend of v.

To evaluate **Q1**, we run OSSINT system on a set of eight users. Differently, to evaluate the quality of OSSINT we used concepts as *precision* and *recall*. The *precision* is the fraction of retrieved instances that are relevant while *recall* is the fraction of relevant instances that are retrieved. **f1** considers both the precision and the recall of the test to compute the score. It can be considered as a weighted average of the precision and recall. The value of **f1** is large when both precision and recall are good, and small when either of them is poor. In Formula 1, 2 and 3 the formulas we used.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$f1 = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{3}$$

The confusion matrix in Table 6 summarizes the results of our experiments.

Table 6 demonstrates how our approach can use the information from the 1-hop network to retrieve new friendships at 2-hop of distance. Our OSSINT system correctly marked 253 IDs as "NOT FRIEND", and on the other side, OSSINT has been able to predict an average of 11 new IDs and mark them as "FRIEND". From our studies, we can consider the value of "True Positive" as a good result that demonstrates how our assumptions are correct. Moreover, from the above results, we can assert that we correctly answered to **Q1**.

5.2. Experimental results on personal information

Using the data from Table 6 we can assert that we can find friends at 2-hop of distance. We want now to show the results regarding the personal information like *education*, *hometown*, *current city* of a victim user. The OSSINT system highlights, indeed, a leak of information using the surrounding network of a victim user. The data showed in Table 2 underline the possibility to retrieve personal information of a victim ID that is supposedly being private. Table 7 shows the percentages of the correctness of the user information in the first two positions.

Through our experiments, we can assert that the first and highest two values from the information rate of the 1-hop IDs (Table 2) correspond to the real (from real life) information of our victim ID. From the above results, we can assert that we correctly answered to **Q2** as well. Moreover, we demonstrate the correctness of our assumption showing how it is possible to use the surrounding network of a victim profile in order to retrieve information related to the real life of our victim ID.

Table 7
Correctness of retrieved information in position Top-One and Top-Two.

	Accuracy		
	Current City (%)	Hometown (%)	Education (%)
TOP 1	50.00	75.00	75.00
TOP 2	37.50	12.50	12.50

6. Conclusions

This work aims to present a proof-of-concept approach that demonstrates a significant privacy issue on Facebook. OSSINT is the second building block of our system that started with our previous tool SocialSpy [11]. As for the previous work, also here we exploited only tools publicly available to reveal information that the victim declared private. OSSINT improves the results of SocialSpy, finding new friendship connections at 2-hop of distance from a victim user. Moreover, our study reveals how the list of friends can be exploited to retrieve personal information, of a victim profile, like *education*, *hometown*, *current city*. We chose only real victim profiles from the Facebook social network. To better stress our system, we decided to select profiles with a little amount of publicly available information, at least one public picture and not more than thirteen public pictures. This is due to the fact that a smaller amount of public information entails a higher difficulty of retrieving private information of a victim user. OSSINT shows the feasibility of the new idea we expressed in [11]. The results of our experiments are now able to raise a real concern against Facebook. On the other hand, from our tests, we hope to create awareness on Facebook users.

6.1. Limitations

As mentioned above, our project aims to present a proof-of-concept and to create awareness on Facebook users. On the other hand, we are aware of the small number of victim users of our dataset. However, the small amount of victim profiles is since Facebook recognizes the pattern of actions. When the same actions are repeated more and more times, Facebook blocks the attacker for some days. To avoid to get blocked by Facebook, we introduced some tricks like:

- random delays;
- fake browsing actions;
- twenty different fake profiles.

Random delays are used between one action and another to not appear as a crawler. Moreover, the delays will make our activities as actions from a human profile. The fake browsing actions are used to vary the actions pattern. We, indeed, introduced actions like browsing random pages on Facebook. Fabricated activities let us appear as a human that is surfing on Facebook. Moreover, we used twenty different fake profiles to split the payload of our experiments. Furthermore, we used the fake profiles to appear on Facebook as different users doing different actions on the social network. Lastly, the exiguous number of victim profiles is due to the stricter rules we applied to select them. To be selected, a profile, need to have a really small amount of public information, no public information regarding hometown, city and university and not more than 13 public pictures. We chose to apply the mentioned above to select our victim profiles to stress our OSSINT tool better, and to prove that even on privacy-aware user profiles is possible to retrieve (supposed) private information. Moreover, we are aware of our low precision rate; however, these results come from a long process, and OSSINT demonstrated to be able to reduce the research of new friendships into an OSN environment such as

Facebook. Lastly, among our eight victim profiles, we have been able to have the results above discussed; however, it is to consider that we decided to test the worst case only where our victim users are really privacy aware having few public information available online. The results discussed in Section 5.1, then, can not be used to represent the overall quality of OSSINT. It is, then, not possible to take OSSINT for granted against a random victim profile since that there is a huge variety of profiles in terms of publicly available information and privacy settings.

6.2. Future work

As future work, we want to improve the dataset of victim IDs and rebuild their networks. The experiments, indeed, are still an ongoing task to have more data. However, we showed the feasibility of our proof-of-concept and how it is possible to retrieve (supposedly) hidden information from a random victim user. Moreover, we are working on our system, OSSINT, to introduce new fake actions (e.g., like and revoke the like of a Facebook page, upload pictures, sharing actions and so on). The aim of introducing new fake operations is to make our OSSINT as close as possible to real human interactions. This update will let us remove the random delays since that we truly believe that Facebook will not be able to recognize the difference between our system OSSINT and real human interactions. Lastly, we will improve the whole system introducing new information to exploit. Indeed we want to use the friends list of a victim user again to retrieve further and more information regarding our victim ID. Possible information to exploit can be public pictures where our target ID is tagged rather than comments left on a public friend's wall from our victim or comments where our ID has been tagged.

7. Related work

There are several studies regarding privacy in Online Social Networks (OSNs) in the literature. These works revealed the lack of privacy and security in OSNs and how simple it is, in some cases, to get private information about users. Unfortunately, many OSNs users are unaware of the security risks that exist in these types of communications.

Recent studies [18,19] showed how many OSNs users expose personal details about themselves, their friends, and their relationships, whether by posting photos or by directly providing information such as a home address and a phone number. Furthermore, according to [19], [20] and [21], the Facebook users accept friendship requests from people whom they do not know but with whom they merely have friends in common. By accepting these friendship requests, users unknowingly disclose their private information to strangers. The leakage of information is not only due to friendship requests from strangers but also to the difficulties in correctly tuning the privacy settings in OSN. Almost 13 million users said they had never set or did not know about Facebookâs privacy tools [1]. Moreover, 36% of users share all, or almost all, their wall posts with an audience wider than just their friends [1]. According to our studies and experiments, we hardly believe that users are completely aware of actual privacy settings that OSNs provide them. On the other hand, whenever users know that their profiles have some information leakages, they are often "too lazy" (or inexperienced) to properly modify the privacy options and make the profile private [2,22].

Several studies over the years tried to study what are the privacy and security risks originated from the use of OSNs. Interesting surveys, articles and journals are available online [3,23–26] and all of them try to explain what are the risks, what are the problems and how to try to get rid of threats. In particular, the survey of Fire et al. [25] is an interesting study about threats and solutions in OSNs. The survey is divided in *Threats* and *Solutions* part. Each category explains in details the possible attacks and solutions to mitigate the threats. Other solutions are proposed in [27–30]. Sundry studies from the literature aim at showing how it is possible to attack OSNs platforms and retrieve supposedly hidden information [31–33]. Counterposed some studies try to propose defences to protect OSNs from attacks. We classify all these studies as "Attacks" and "Solutions".

An attack example is in [11]. Burattin et al. demonstrate how it is possible to retrieve the friends list of a victim user on Facebook, using public information. Results have an average of 25% of hidden friends found and with peaks of 70%. The work in [11] shows that the lack of information on Facebook gives the possibility to an attacker to retrieve information supposedly hidden. Counterposed, the solution proposed by Fire et al. in [34] with Social Privacy Protector software (SPP). The SPP software consists of two main parts, namely, a Firefox add-on and a Facebook application. The two parts provide Facebook users with three different layers of protection. The first enables Facebook users to control their privacy. The second notifies what applications can arise privacy threats. The third layer analyzes a user's friends list. Other example of attack and solution are proposed in [35–37]. Lindamood et al. show how third parties (i.e. friends, groups, affiliations) can be the cause of a leak of personal information. Zheleva and Getoor in [36] propose, as a possible solution to mitigate the inference of private information, the removal of trait details and friendship links. The study in [37] addresses two different issues: *(a)* how third parties users launch an inference attack, *(b)* are there effective strategies to protect against such an attack.

Lastly, we have a less studied problem, the *social graph privacy*: preventing data aggregators from reconstructing large portions of the social graph, composed of users and their friendship links. The study in [38] examines the difficulty of computing graph statistics given a random sample of K edges from each node and found that many properties can be approximated. Our study shows how it is possible to reconstruct a friendship graph of a victim profile. Using the SocialSpy tool, first, we retrieve the friends list of a victim user. Then we re-run SocialSpy on the IDs from the friends list of our victim to have a 2-hop friends lists. Once we have the friends list, we use OSSINT to build a friendship graph of our victim ID. The system gives us the possibility to know who are the closest friends of our victim, to de-anonymize those users with a high level of privacy that were not retrievable by SocialSpy, and to profile our victim user using the information available from the 1-hop IDs. The main difference between our study and the others is that for our experiments we faced directly with the Facebook social network. We did not use dataset collection like the one from Stanford University [39]. All our data and experiments have been performed on real Facebook profiles. OSSINT, indeed, not only shows how to infer and extract real data from real profiles but also handles (and bypasses) all the security systems built by Facebook to prevent security threats.

Conflict of Interests

None

Acknowledgment

Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission under the agreement n. PCIG11-GA-2012-321980. This work has been partially supported by the TENACE PRIN Project 20103P34XC funded by the Italian MIUR, and by the Project "Tackling Mobile Malware with Innovative Machine Learning Techniques" funded by the University of Padua, and by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061).

References

[1] Consumer Reports Magazine, Facebook & your privacy, 2012, http://www.consumerreports.org/cro/magazine/2012/06/facebook-your-privacy.
[2] M. Madejski, M. Johnson, S.M. Bellovin, A study of privacy settings errors in an online social network, in: Proceedings of the IEEE PERCOM Workshops, 2012, pp. 340–345.
[3] R. Gross, A. Acquisti, Information revelation and privacy in online social networks, in: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, 2005, pp. 71–80.
[4] F. Beato, M. Kohlweiss, K. Wouters, Scramble! your social network data, in: S. Fischer-Hübner, N. Hopper (Eds.), Privacy Enhancing Technologies, PETS'11, Springer-Verlag, 2011, pp. 211–225.
[5] C. Tang, K. Ross, N. Saxena, R. Chen, What's in a name: A study of names, gender inference, and gender behavior in Facebook, in: Springer-Verlag, 2011, pp. 344–356.
[6] M. Conti, A. Hasani, B. Crispo, Virtual private social networks and a Facebook implementation, ACM Trans. Web (2013).
[7] M. Kosinski, D. Stillwell, T. Graepel, Private traits and attributes are predictable from digital records of human behavior, Proc. Natl. Acad. Sci. 110 (15) (2013) 5802–5805. http://europepmc.org/abstract/MED/23479631.
[8] G. Cascavilla, M. Conti, D.G. Schwartz, I. Yahav, Revealing censored information through comments and commenters in online social networks, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), ACM, 2015, pp. 675–680.
[9] osint.it, OSINT, one important kind of intelligence, 2015, http://www.osint.it/english/open-source-intelligence-osint.as.
[10] R.D. Steele, Open source intelligence, in: Handbook of Intelligence Studies, Routledge New York, 2007.
[11] A. Burattin, G. Cascavilla, M. Conti, SocialSpy: Browsing (Supposedly) Hidden Information in Online Social Networks, Springer International Publishing, 2015, pp. 83–99.
[12] J. Constine, Facebook Announces Friendship Pages That Show Friends' Mutual Content, 2010, http://www.insidefacebook.com/2010/10/28/friendship-pages-mutual-content.
[13] I. Abezgauz, Facebook vulnerability discloses friends lists defined as private, 2013, http://www.quotium.com/resources/facebook-vulnerability-discloses-friends-lists-defined-as-private.
[14] C. McGoogan, Facebook denies using your phone's location data to suggest new friends, 2016, http://www.telegraph.co.uk/technology/2016/06/28/facebook-is-using-your-phones-location-to-suggest-new-friends.
[15] N. Pineda, Facebook tips: What's the difference between a Facebook Page and Group?, 2010, https://www.facebook.com/blog/blog.php?post=324706977130.
[16] K.S. Burns, Social Media: A Reference Handbook (Contemporary World Issues), 2017.
[17] K. Ron, P. Foster, Glossary of terms, Mach. Learn. 30 (1998) 271–274.
[18] R. Acquisti, R. Gross, Imagined communities: awareness, information sharing, and privacy on the Facebook, in: Proceedings of the 6th Workshop on Privacy Enhancing Technologies, 2006.
[19] Y. Boshmaf, I. Muslukhov, K. Beznosov, M. Ripeanu, The socialbot network: when bots socialize for fame and money, in: ACSAC '11, ACM, 2011.
[20] F. Nagle, L. Singh, Can friends be trusted? Exploring privacy in online social networks, in: Proceedings of the ASONAM, 2009, pp. 312–315.
[21] L. Bilge, T. Strufe, D. Balzarotti, E. Kirda, All your contacts are belong to us: automated identity theft attacks on social networks, in: Proceedings of the 18th International Conference on World Wide Web, ACM,
[22] P.J. Wisniewski, B.P. Knijnenburg, H.R. Lipford, Making privacy personal: Profiling social network users to inform privacy education and nudging, Int. J. Human-Comput. Stud. (2016).
[23] D. Bernhard, L. Jennette, H.M.A. Ann-Kathrin, H. Brittany, Facebook and online privacy: attitudes, behaviors, and unintended consequences, J. Comput-Mediated Commun. (2009).
[24] R. Lee, K. Sara, Anonymity, privacy, and security online, 2013, http://www.pewinternet.org/2013/09/05/anonymity-privacy-and-security-online/.
[25] M. Fire, R. Goldschmidt, Y. Elovici, Online social networks: threats and solutions survey, arXiv:1303.3764 (2013).
[26] K. Xu, Y. Guo, L. Guo, Y. Fang, X. Li, My privacy my decision: Control of photo sharing on online social networks, IEEE Trans. Dependable Secure Comput. 14 (2) (2017) 199–210.
[27] H.R. Lipford, A. Besmer, J. Watson, in: Understanding privacy settings in Facebook with an audience view, 2008. http://www.usenix.org/events/upsec08/tech/full_papers/lipford/lipford.pdf
[28] L. Fang, K. LeFevre, Privacy wizards for social networking sites, in: Proceedings of the 19th International Conference on World Wide Web, ACM, 2010, pp. 351–360.
[29] M. Fire, D. Kagan, A. Elishar, Y. Elovici, Social privacy protector-protecting users' privacy in social networks, in: Proceedings of the Second International Conference on Social Eco-informatics, SOTICS, 2012.
[30] T. Paul, M. Stopczynski, D. Puscher, M. Volkamer, T. Strufe, C4PS: colors for privacy settings, in: Proceedings of the 21st International Conference Companion on World Wide Web, ACM, 2012.
[31] A. Mislove, B. Viswanath, K.P. Gummadi, P. Druschel, You are who you know: Inferring user profiles in online social networks, in: Proceedings of the Third ACM International Conference on Web Search and Data Mining, in: WSDM '10, 2010.
[32] J. Mahmud, J. Nichols, C. Drews, Home location identification of twitter users, ACM Trans. Intell. Syst. Technol. (2014) 47:1–47:21.
[33] Y. Gu, Y. Yao, W. Liu, J. Song, We know where you are: home location identification in location-based social networks, in: Proceedings of the 25th International Conference on Computer Communication and Networks (ICCCN), 2016, pp. 1–9.
[34] M. Fire, D. Kagan, A. Elyashar, Y. Elovici, Friend or foe? fake profile identification in online social networks, arXiv:1303.3751 (2013).
[35] J. Lindamood, R. Heatherly, M. Kantarcioglu, B. Thuraisingham, Inferring private information using social network data, in: Proceedings of the 18th International Conference on World Wide Web, in: WWW '09, ACM, 2009, pp. 1145–1146.
[36] E. Zheleva, L. Getoor, To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles, in: Proceedings of the 18th International Conference on World Wide Web, in: WWW '09, ACM, 2009, pp. 531–540.
[37] Z. Cai, Z. He, X. Guan, Y. Li, Collective data-sanitization for preventing sensitive information inference attacks in social networks, IEEE Trans. Dependable Secure Comput. PP (99) (2017) 1–1..
[38] J. Bonneau, J. Anderson, R. Anderson, F. Stajano, Eight friends are enough: Social graph approximation via public listings, 2009.
[39] J. Leskovec, Stanford large network dataset collection, http://snap.stanford.edu/data/egonets-Facebook.html.