

Preserving Privacy on the Searchable Internet

Ruxia Ma

School of Information, Renmin
University of China, Beijing, China
The Capital Normal University, Beijing,
China
maruxia@ruc.edu.cn

Xiaofeng Meng

School of Information, Renmin
University of China, Beijing, China
xfmeng@ruc.edu.cn

Zhongyuan Wang

Microsoft Research Asia, Beijing,
China
zhywangchina@163.com

ABSTRACT

The Web is the largest repository of information. Personal information is usually scattered on various pages of different websites. Search engines have made it easier to find personal information. An attacker may collect a user's scattered information together via search engines, and infer some privacy information. We call this kind of privacy attack *Privacy Inference Attack via Search Engines*.

In this paper, we propose a user-side automatic detection service for detecting the privacy leakage before publishing personal information. In the user-side service, we construct a User Information Correlation (UICA) graph to model the association between user information returned by search engines. We map the privacy inferring attack into a decision problem of searching a privacy inferring path with the maximal probability in the UICA graph. We propose a Privacy Leakage Detection Probability (PLD-Probability) algorithm to find the privacy inferring path. Extensive experiments indicate that the algorithm is reasonable and effective.

Categories and Subject Descriptors

K.4.1 [Computers and Society]: Public Policy Issues – Privacy.

General Terms

Security, Human Factors, Algorithms

Keywords

Privacy, Inference, Search Engines

1. INTRODUCTION

Nowadays, the Web has been the largest repository of information. With the rise of Web2.0, more and more people can publish information on the Web. An amount of personal information rises sharply on the Web, e.g. employees' identity data in business enterprises' Web pages, and students' archives (including grade, record and identity, etc.) in schools' databases, etc. When a user wants to publish some privacy information, he maybe hides directly identifying information. But it is far from sufficient to protect his privacy. The myriad personal information on the Web is published in one way or another. And it can be associated with the privacy information to infer the user's identity, so that his privacy is unconsciously leaked. The personal information is usually distributed on different websites. How to get it? Information has never been easier to find. Search engines allow easily access to all kinds of information on the Web. Therefore,

according to background information and the hidden associations between scattered information an adversary collect the decentralized information of a specific person together via search engines. Then he can easily infer users' privacy or identity from fragments of information collected by search engines. As a result users' privacy information is leaked. Obviously, such information collection may infer Web users' personal privacy and bother the personal lives. We refer this kind of privacy attack as *Privacy Inferring Attack via Search Engines*. Unfortunately it is usually ignored by Web users.

1.1 Motivation Example

In order to make sense of the privacy inferring attack via search engines, we give a concrete example. In this example, Trudy is an adversary and Alice is a victim. Alice's personal information is distributed over the Web, but Trudy can obtain both privacy-sensitive information and identity information related to Alice with search engines.

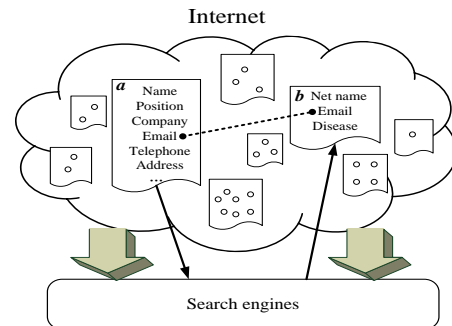


Figure 1. Motivation Example

In Figure 1, each data item represents a message of Alice, e.g. Name, Email and Address. Each Web page contains several data items. For example, page *a* can be regarded as an identity Web page which comes from the website of Alice's company, and page *b* can be regarded as a privacy-sensitive Web page which comes from a forum about diabetes where Alice asked for help. Trudy has Alice's telephone number which is called background information. She can launch a search with the telephone number as a search keyword. In returned results, there are many pages containing the telephone number. Obviously page *a* will be returned. Trudy obtains these related web pages and examines all data items. She just holds data items related to Alice and uses some of them as new keywords for next searches. When Trudy uses Alice's email as a new keyword to search, he can discover new pages such as page *b*. Finally, she can get Alice's disease information which is privacy information for Alice. It looks like there are link bridges among these web pages. Even if Trudy just has little information about Alice at first, the privacy may be collected and mined successfully via search engines.

People usually put too much confidence in the privacy preserving of websites where they publish personal information. But some websites provide their page hyperlink for search engines to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

iiWAS2011, 5–7 December, 2011, Ho Chi Minh City, Vietnam.
Copyright 2011 ACM 978-1-4503-0784-0/11/12...\$10.00.

improve website traffic. Even if some protective mechanisms (e.g. Web users must access these sites through a username and a password) have been established for privacy information publishing, they block information communication and sharing for fresh users. We focus on how neither harm the exchange and sharing of information, while protecting the privacy of users.

1.2 Summary of Contributions

The problem of privacy inferring attack via search engines is a new challenging problem that is easily ignored. In this paper, we take the initiative to tackle the problem. Our contributions including: firstly, we show a new kind of privacy threat to all Web users, and we propose a model which describe the principles and process of privacy mining attack via search engines; secondly, we construct a user information correlation graph to model the association between user information returned by search engines; thirdly, we propose a PLD-Probability algorithm to find the privacy inferring path in user information correlation graph; finally, in order to battle this privacy attack, we propose a user-side automatic detection service based on the construction method of the user information correlation graph and the PLD-Probability algorithm for detecting the privacy inference before publishing personal information.

2. PRIVACY INFERENCE ATTACK VIA SEARCH ENGINES

In this section, we will show how an adversary can exploit the Web with some public user information and search engines to infer the private information of users.

2.1 Definitions

Firstly, it is necessary to identify the characteristics and categories of personal information on the Web. There is a wide variety of personal information about Web users on the Web. Formally, we define the following three categories of personal information according to their privacy-sensitive degree:

DEFINITION 1 (Personal Information).

Identity Information (I): *One person's open social identity.* E.g. SSN, ID number, full name, Credit card numbers, birthday, etc. This kind of information can be used to confirm a person's identity.

Privacy-Sensitive Information (S): *Privacy-sensitive topic or personal privacy,* e.g. salary, diseases, drunkenness, gambling etc. The boundaries and content of what is privacy information differ among culture and individuals, but share basic common themes. In order to reflect personalize privacy, users can specify privacy information.

Other Information (O): *Some inessential information.* E.g. interest, education, marriage state. It can't help identify people's identity, and doesn't refer to privacy-sensitive information.

Universal Set (U): *The set containing all the information of a specific person.* $U = I \cup S \cup O$.

I information and *S* information related to the same person seldom appear on the same page. If they are collected together and obtained by the attacker, this person's privacy is leaked. Thus we make the following definition:

DEFINITION 2 (Privacy Inferring Attack via Search Engines).

The attacker collects the information about the victim on the Web via search engines until he obtains both identity information (I) and privacy-sensitive information(S) related to the victim.

DEFINITION 3 (Inferring Paths). *The process of collecting the victim's I information and S information together according to relationships among them.*

The Goal of Privacy Inferring Attack: *what is a new privacy attack that collects the scattered I and S information about the victim together via search engines.*

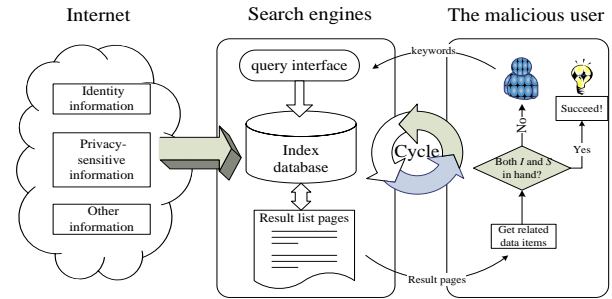


Figure 2. The Process of Privacy inferring via Search Engines

2.2 Description of the Attack

Figure 2 shows the process of an attacker collecting the victim's *I* and *S* information via search engines.

Before launching the privacy inferring, the attacker usually has some background information related to the victim. It may be *I*, or *S*, or *O*. The attacker uses background information as seeds to infer the victim's privacy via search engine. Without loss of generality, we suppose the attacker has known some *I* information of the victim at first. He launches queries with these keywords. Then he gets result pages returned by search engines. From these web pages, the attacker can find new and valuable information about the victim. The new information may be *I*, *S*, *O*, or the mixture of them.

(1) If the new information contains *S*, the attacker has obtained *I* and *S*. The victim's *I* information is linked up with his *S* information and the privacy attack successes.

(2) If the new information contains only *I* and *O*, it is put into the set of obtained information. Then, the attacker chooses keywords from the obtained information and goes on the next query. When new pages are returned, he checks whether they contains the victim's *S* information.

The attacker repeats this process to collect the scattered information of this victim on the Web until he finds the person's *S* information.

3. PRIVACY LEAK DETECTION SERVICE

In order to battle Privacy Inferring Attack via Search Engines, we try to provide a user-side automatic detection service for detecting the privacy inference before publishing personal information. We refer this service as a **Privacy Leakage Detection Service** which can detect whether personal privacy has the risk of leakage by information inferring attack via search engines. When a user publishes new information to the Web, the service will collect the user's decentralized information together and try to mine out his *I* and *S* information, then find a privacy inferring path with the maximal connectivity probability which denotes the possibility of privacy leakage. The user can find whether his privacy will leak and decide which information he should replace.

3.1 Overview

The hypostasis of Privacy Inferring Attack via Search Engines is linkage relationship mining on the Web to infer users' privacy. Users usually do not remember all the previously published information on the Web, so this problem is often ignored and difficult to prevent.

In this paper, the **Privacy Leakage Detection Service** first collects user information scattered on the Web, and records the user information and linkage relationships between them to form a *user information correlation graph (UICA)*. Then it simplifies the graph to reduce the size of graph. But the user information obtained maybe doesn't belong to the current user because of various impact factors. So we consider the impact factors space, and assign a reasonable probability for each vertex to represent possibility the vertex belongs to the user. Finally, the service searches a connected privacy inferring path with the maximal probability between identity vertex and privacy-sensitive vertex based on a PLD-probability algorithm.

The user-side service actually simulates the process of privacy inferring attack, so it needs to be provided by a trusted third party. When users want to post information on the Web, they can use the service to check privacy leakage.

Because of different culture and different privacy-sensitive, people have different privacy definitions. E.g. some users think that their phone numbers are privacy information. However, other users may want to open their phone numbers on the Web for more convenient communication with online friends. Thus, in the privacy leakage detection services, it is not fixed that which kind of information (*I*, *S* or *O*) a specific data item belongs to. Users can configure it according to their different requirements. So our approach is universal and personalized.

Personal information is scattered in various Web pages with large scale and structural inconsistencies. So we need to extract personal information from Web pages returned by search engines. It's difficult to find an automatic way to accurately extract all kinds of data items from Web pages. According to the existing work about Web data extraction, we adopt two extraction techniques for different types of personal information: (1) for format data items, using the regular expressions; (2) for non-format data items, using weighted mutual exclusion bootstrapping for entity extraction and labeling [1]. The first method can extract phone number, email address, SSN, date, etc. The second method can extract name, sex, nationality, companies, organizations, position, location, diseases, etc. Then we use these extracted data items to build a user information correlation graph for the user who orders our service.

3.2 User Information Correlation Graph (UICA graph)

This service is customized for each user. It uses automatic algorithms based on search engines to collect information for every user and build a UICA graph $G_{detection}$.

DEFINITION 4 (Vertex). Each vertex represents a user data item extracted from the Web page. It's a 7-tuple notation:

$$v_i = (ID, Attribute, Value, Probability, Category, PageID, Depth)$$

Where *ID* is the sequence number of data items; *Attribute* represents the attribute name of a user data item and is added when the data item is extracted, such as name, email, etc.; *Value* is a attribute value of a user data item, such as Alice, Alice@gmail.com, etc.; *Probability* represents the possibility of

this data item belong to the user; *Category* is the category (*I*, *S* or *O*) of the data item; *PageID* is the sequence number of the Web page in which the user data item is.

DEFINITION 5 (Edge). According to the different relationships between user data items, $G_{detection}$ contains two kinds of edges: internal edges and external edges.

Internal edge: Link between user data items in the same Web page. We can use E_{in} to denote the set of internal edges:

$$E_{in} = \{(v_i, v_j) \mid v_i.PageID = v_j.PageID\}$$

External edge: Link between data items with the same attribute and value in the different Web pages. We use E_{out} to indicate the set of external edges:

$$E_{out} = \{(v_i, v_j) \mid v_i.Attribute = v_j.Attribute \wedge v_i.Value = v_j.Value \wedge v_i.PageID \neq v_j.PageID\}$$

In next section, we talk about how to build a UICA graph.

3.3 UICA Graph Construction

Based on the definition of the UICA graph $G_{detection}$, we provide an algorithm for the graph construction, which we call GraphConstructor (see Algorithm 1 for the pseudocode).

The GraphConstructor algorithm takes as input some user information which will be published by the user and outputs a UICA graph G of this user. The algorithm is parameterized by a value D that controls the depth of iterative search, and by a value M that controls the number of returned Web pages. The two values can be specified by users or is set through experience statistics.

The UICA graph construction approach is iterative and in each step it will maintain the vertex set V and the edges set E . In each iterative it sequentially picks a vertex v_i from the set V and uses the Value property value of the node v_i as a search keyword to launch a query. Then it takes the top M Web pages containing the search keyword and extracts user data items from the selected Web pages with the information extraction methods mentioned above. Here every user data item will generates a new vertex. For the user data item on each selected Web page which has the same attribute and value with the node v_i , the algorithm creates an external edge to connect it with the node v_i . For each pair of user data items on each web page, it generates an internal edge between them. Then it inserts these new vertices into the vertex set V and sets the Depth attribute for each new vertex to the sum of $v_i.Depth$ and 1. Those new user data items will be used as search keywords to mind more user information in the process of following iterations.

Algorithm 1 The GraphConstructor algorithm.

Input: initial set $V = \{v_1, v_2, \dots, v_k\}$, where $v_i.Probability = 1$ and $v_i.Depth = 0$;

Output: V, E // Vertices and edges of $G_{detection}$;

1. $E \leftarrow \emptyset$;
 2. **For** ($i = 1$; $i \leq k$ && $v_i.Depth < D$; $i++$) **Do**
 3. $keyword \leftarrow v_i.Value$;
 4. launch a query to search engines using $keyword$;
 5. get the top M returned result pages: $\{W_1, \dots, W_M\}$
 6. **For** ($j = 1$; $j \leq M$; $j++$) **Do**
 7. **If** W_j has been selected before **Then**
 8. From vertex set V get the node v_l which is on the
-

```

web page  $W_i$  and have the same attribute and value
with  $v_i$ ;
9.   Generate an external edge between  $v_i$  and  $v_{j_i}$ ;
10.  Continue;
11.  End If
12.  extract data items to form  $v_{j_1}, v_{j_2}, v_{j_3}, \dots, v_{j_m}$  from  $W_j$ ;
13.  insert an external edge  $e_{i-j_l}$  into  $E$  which denote
vertex  $v_i$  can find  $v_{j_l}$  which is the keyword appearing in the
Web page  $W_j$ ;
14.  For ( $w = 1; w \leq m; w++$ ) Do
15.    For ( $l=w+1; l \leq m; l++$ ) Do
16.      generate all  $e_{j_w-j_l}$  to denote vertex  $v_{j_w}$  and  $v_{j_l}$  are
in the same Web page;
17.      Insert the internal edge  $e_{j_w-j_l}$  into  $E$ ;
18.    End For
19.  End For
20.  insert  $v_{j_1}, v_{j_2}, \dots, v_{j_m}$  into  $V$ ;
21.   $k+=m$ ;
22.  End For
23. End For
24. Return  $V, E$ 

```

3.4 UICA Graph Simplification

According to the process of building a UICA graph $G_{detection}$, it has duplicate data items which contain the same attributes and values. Therefore, we need to simplify the UICA graph and reduce the scale of $G_{detection}$. We propose a graph simplification algorithm called GraphSimplification, the pseudocode of which is shown in Algorithm 2.

For input the UICA graph $G(V, E)$, the algorithm first finds all the external edges and push them into a stack *OuterEdgeStack* (line 2-6). Then in line 7-22, we repeatedly pick the top external edge e_k from the stack until the stack is empty. For the edge e_k , we keeps all the internal edges connecting with two endpoints of e_k and deletes other internal edges between vertices on the same web page with the each endpoints of e_k . Then we delete the edge e_k and merge the two endpoints of e_k into a new vertex v_k , update the edges connected with two endpoints of e_k . So we can reduce the redundant vertices and edges. Finally, the algorithm returns the simplified graph $G(V^s, E^s)$. In next session, we will find a privacy inferring path form a simplified user correlation graph.

Algorithm 2 The GraphSimplification algorithm.

Input: the original UICA graph

Output: new graph G^s after simplification

```

1.  OuterEdgeStack  $\leftarrow \emptyset$ ;
2.  For ( $i=1; i \leq |E|; i++$ ) Do
3.    If ( $v_i^1.Attribute = v_i^2.Attribute \ \&\& \ v_i^1.Value = v_i^2.Value$ 
       $\ \&\& \ v_i^1.PageID \neq v_i^2.PageID$ ) Then //  $e_i = (v_i^1, v_i^2)$ 
4.      push  $e_i$  into OuterEdgeStack;
5.    End If
6.  End For
7.  While (OuterEdgeStack  $\neq \emptyset$ ) Do
8.     $e_k = \text{Pop}(\text{OuterEdgeStack})$ ;
9.    get vertices  $v_k^1, v_k^2$  connected by  $e_k$ ;
10.  get vertices  $v_1, v_2, \dots, v_m$  whose PageIDs equal
 $v_k^1.PageID$  // These vertices have internal edges with  $v_k^1$ ;
11.  keep edges between  $v_1, v_2, \dots, v_m$  and  $v_k^1$ , delete other

```

```

edges;
12.  get vertices  $v_1', v_2', \dots, v_l'$  whose PageIDs equal
 $v_k^2.PageID$ ;
13.  keep edges between  $v_1', v_2', \dots, v_l'$  and  $v_k^2$ , delete other
edges;
14.  delete  $e_k$  from  $E$ ;
15.  delete  $v_k^1$  and  $v_k^2$  from  $V$ ;
16.  generate a new vertex  $v_k$  which is the merge vertex of
 $v_k^1$  and  $v_k^2$ ;
17.   $v_k.Attribute = v_k^1.Attribute$ ;
18.   $v_k.Value = v_k^1.Value$ ;
19.   $v_k.PageID = null$ ;
20.  insert  $v_k$  into  $V$ 
21.  update edges connected  $v_k^1$  and  $v_k^2$  to  $v_k$ 
22. End While
23. Generate new sequence numbers as ID for vertices
24. update edges in  $E$ 
25. Return  $G^s$ 

```

3.5 PLD-Probability Algorithm

In the simplified UICA graph G^s , vertices are different from each other. Each vertex is attached with a probability value which denotes the possibility that the data item vertex belonging to current user. The graph G^s is a general graph with weighted vertices.

Without loss of generality, I vertex representing identity user information is defined as the starting vertex, and S vertex denoting privacy-sensitive information is defined as the target vertex accordingly, vice versa. According to the Algorithm 1, vertices are iteratively constructed step by step from initial data items. Therefore, probabilities of vertices are dependent between each other.

Actually, many factors influence the vertex's probability. For example, the proportion of vertices with the same attribute value, the distance between data items and keyword item on a Web page, and so on. The probability of vertices directly affects the probability of the privacy inferring path between I vertex and S vertex. In order to find the privacy inferring path, we need to compute the probability of each vertex.

We define a special variable φ to describe the implicit dependency relationships among probability values of vertices. The values of φ form the impact factors space: *IFSpace*. Suppose values of φ comes from a finite set of discrete numbers $\{1, 2, \dots, n\}$. Variable φ and its values are abstract and simplified representations of impact factors of vertices' probabilities. For example, the variable φ with a value of 1 can represent the proportion of vertices with the same attribute value. The probability of each value of φ is the prior probability of computing vertices' probabilities. Each vertex v_i has a probability value denoting the probability of belonging to current user, therefore the privacy inferring path between I vertex and S vertex has a connectivity probability value. Therefore, the privacy inferring problem is defined as: whether there is such a privacy inferring path in the current user information graph, the connectivity probability value of which is a threshold T at least? This problem is a NP-complete problem. We will focus on the problem on searching the privacy inferring path with the maximum probability in the UICA graph by using φ variable as the prior probability.

The essential reason of the privacy inferring path searching problem becoming NP-Complete problem is that the special variable φ has unknown number of values, and the probability $P(\varphi=j)$ of each value of φ is not known. Therefore, it is impossible to find a perfect solution theoretically. There are a variety of factors impacting vertices' probability. Suppose that φ has k possible values, then $\sum_{j=1}^k P(\varphi=j) = 1$. The probability $P(\varphi=j)$ represents the weight of the j -th factor's impact on vertices' probabilities. According to Formula of Total Probability, the probability of the vertex v_i is equal to $\sum_{j=1}^k P(v_i|\varphi=j) \cdot P(\varphi=j)$.

In order to provide reasonable and feasible privacy leakage detection service for Web users, we propose a reduced algorithm: PLD-Probability Algorithm. The core of the algorithm is how to compute the probabilities of vertices. We choose two significant factors in the *IFSpace*: (1) the entropy based on the value collection of attributes; (2) the average distance between new data items and query keywords on the same Web page. Namely, φ have two possible values: 1 and 2. We will introduce the two effect factors respectively.

(1) *The entropy based on the value collection of attributes*

In the process of collecting the user's personal information, it is critical that identifying which values of data items with the same attribute belong to the current user. The different values of data items with the same attribute constitute a collection, called the value set of the attribute. If the collection has high purity, most of the elements have the same value, and this value is the most possible candidate for the attribute. Therefore, we use the entropy to characterize the confusion degree of certain an attribute's value collection: $Entropy(S_A) = \sum_{i=1}^c -p_i \log_2 p_i$.

Where c is the number of distinct values for the attribute A and S_A is the value set of the attribute A . Here, p_i denotes the portion of the i -th category value in the collection S_A . For example, four kinds of values have been found for data items with "Email" attribute in information searching for current user. They are Alice@gmail.com, AliceL@yahoo.com.cn, Aileen@gmail.com and Alice-Lee@live.com. The amount of data items with the value "Alice@gmail.com" occupies 80% of all found data items with the Attribute "Email", and which is the first category value. Therefore, $p_1=80\%$. The more the same value appears in the attribute's values collection, the smaller the entropy value is. The maximum value of the entropy is $\log_2 c$. The factor's impact for the probability of the vertices with the i -th category value c_i can be computed by the formula:

$$ImpactFactor(E)_{C_i} = p_i \cdot \left(1 - \frac{Entropy(S_A)}{\log_2 c}\right).$$

(2) *The average distance between new data items and query keywords on the same page*

Information related to the same user usually is close together on the same Web page. Based on observations, the same user's data items usually locate in one Web table or DIV region on a Web page. Based on such characteristic, we define the following formula to compute the impact effect of the average distance between new items and query keywords on the same Web page:

$$ImpactFactor(D)_i = \begin{cases} 1, & \text{if } v_i \text{ in } WebTable \\ 1 - \frac{\sqrt{(x_i - x_{keyword})^2 + (y_i - y_{keyword})^2}}{\sqrt{x_{region}^2 + y_{region}^2}}, & \text{if } v_i \text{ in } DIV \end{cases}$$

In the formula, v_i is the i -th data item on the Web page. x_i and y_i denote the abscissas and ordinate of v_i on the page respectively. $x_{keyword}$ and $y_{keyword}$ denote the abscissas and ordinate of the keyword item on the same Web page. x_{region} and y_{region} denote the width and height of current region (Web table or DIV). The probability $ImpactFactor(D)_i$ of the data item v_i belonging to the current user is inversely proportional to the distance.

Next we use the statistics to represent the impact of the C_i type of value in an attribute's value collection:

$$ImpactFactor(D)_{C_i} = \begin{cases} \bigvee_{v_k.value=C_i} ImpactFactor(D)_k, & \text{if } v_k \text{ in } WebTable \\ avg_{v_k.value=C_i} ImpactFactor(D)_k, & \text{if } v_k \text{ in } DIV \end{cases}$$

If data items v_k with the C_i type of value and the keyword item appear in the same Web table for more than one and one times, the impact value is 1. If they appear in the same DIV, we use the average value to represent the impact effect.

Based on the two impact factors, a vertex probability formula is defined as follows:

$$P(v_i \in G^S) = (\omega_E * ImpactFactor(E)_{C_i} + \omega_D * ImpactFactor(D)_{C_i}) * \underset{v_{keyword} \Rightarrow v_i}{avg} P(v_{keyword} \in G^S)$$

Here, ω_E and ω_D represent the weights of the two significant factors respectively. As mentioned previously, they also can be expressed as $P(\varphi=1)$ and $P(\varphi=2)$. And the sum of ω_E and ω_D is 1. As shown in the formula, the probability of the vertex v_i belonging to the current user is affected by the probability of the keyword items. So we use the average of the probability of all keyword items to compute the probability of the vertex v_i .

After determining probability values of all vertices in the UICA graph, we use the greedy algorithm to search the privacy inferring path with the maximum probability, as shown in Algorithm 3.

The algorithm 3 maintains three arrays: d , *previous* and *indegree*. $d[v]$ denotes the maximum probability of the privacy inferring paths to vertex v . *previous*[v] represents the previous node of v in the privacy inferring path with maximum probability. And *indegree*[v] keeps the in-degree of v . First, we initialize the three arrays (line 1-5). Iteratively, we pick the vertex u from the vertex set Q if $d[u]$ is the maximal value in the array d and *indegree*[u]=0. The vertex u is removed from the set Q . For each edge (u,v) outgoing from u , we set u as the precursor of v and update $d[v]$ to $\arg \max\{d[v_i] \mid v_i \in Q\}$ if $d[v] < d[u] + P(u)$. Then the in-degree of v is decremented by one. We scan the set Q and stop once the set Q is empty. Finally, based on the two arrays *previous* and d , the privacy inferring path with maximum probability is constructed (line 20-25).

Algorithm 3 The algorithm GreedyPrivacyInferring.

Input: graph G^S after simplification

Output: a privacy mining path with the maximal connectivity probability

1. **for** each vertex v in $V[G]$ // initialization

2. $d[v]=0$;
3. $previous[v] := \text{undefined}$;
4. set $indegree[v]$ to the in-degree of v in graph G^s ;
5. **End for**
6. $S := \text{empty set}$;
7. $Q := \text{set of all vertices}$;
8. **while** Q is not an empty set
9. $u := \arg \max \{d[v_i] \mid v_i \in Q \wedge indegree[v_i]=0\}$;
10. remove u from Q and $S := S \text{ union } u$;
11. **for each edge** (u,v) outgoing from u
12. if $d[v] < d[u] + P(u)$ then
13. $d[v] := d[u] + P(u)$;
14. $previous[v] := u$;
15. **End if**
16. $indegree[v]--$;
17. **End for**
18. **End while**
19. $Path := \text{empty sequence}$;
20. Find a privacy-sensitive vertex w with the maximum $d[w]$;
21. **while** defined w
22. insert w to the beginning of $Path$;
23. $w := previous[w]$;
24. **End while**
25. **return** $path$

4. EXPERIMENTS

In this section, we made three kinds of experiments to evaluate the seriousness of privacy leakage problem and the performance of our methods. The first one is a statistical experiment to verify the existence and seriousness of the privacy problem. The second is a rationality verification experiment. And the third one is an effectiveness evaluation experiment.

4.1 The Statistical Experiment

This kind of privacy threat proposed in this paper is different from traditional privacy threats. It is based on open and available information on the Web. And the malicious user can collect the victim-related information via search engines.

We conduct a statistical experiment to examine whether privacy inferring attack via search engines may happen in practice. In additional, we also can observe the importance of different linkage information (such as username, email, etc.) which is used as keywords to search for other user information.

In the statistical experiment, we generate a real data set. Firstly, we randomly choose three web forums which contain multiple users' privacy-sensitive information (S). They are web forums or communities for some sensitive diseases: cancercompass.com, diabetsdaily.com and aidscommnityservices.com. We randomly choose some users from each forum as experimental objects. Then we check the content of the information previously published by these users, and filter users who didn't referred to themselves suffering from sensitive diseases. We ultimately obtain a data set from these three forums: 1000 users and their S information.

Each user has a unique username in a forum, and some users open their email addresses. We use these data with other information on these forums (such as location, birthday and so on) as known background knowledge to find users' identity information. Through multiple searching steps, we find some users' identity information (I) on other websites such as company websites, social organization websites, etc.

Finally, identity information and privacy-sensitive information of 460 users could be gathered up successfully. Namely, the success rate of the privacy inferring attack is 46 percent. So privacy inferring attacks via search engines are a real issue for users' information publishing on the Web.

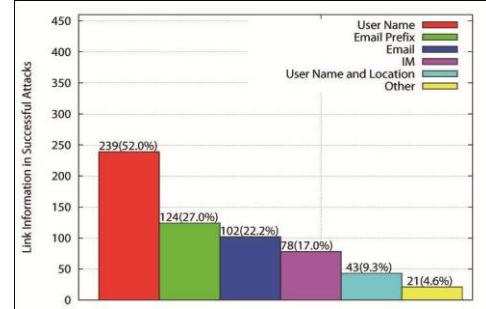


Figure 3. Statistics of Linkage Information Frequency

Figure 3 shows the importance of different user information in privacy inferring processes. User name, email address, email prefix, IM account and the combination of user name and location are the top five user information which are the most possible to play the role of linkage information in actual inferring processes. This experiment shows which kinds of user information easy to result in privacy leakage.

Figure 4 reports the cost of privacy inferring attack which is measured by the length of privacy inferring paths. For the successful attacks, the lengths of inferring paths almost tend to fellow a normal distribution. Most of the path lengths are in a range between 6 steps to 8 steps.

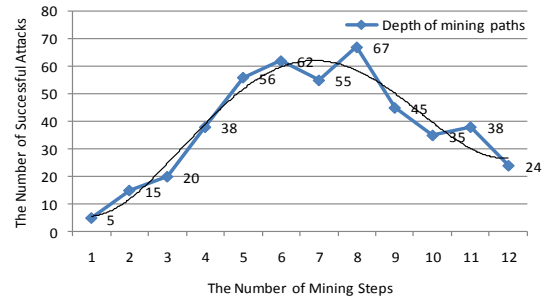


Figure 4. Statistics of inferring Step

Results of the three statistical experiments show that privacy inferring attacks are very serious and widespread.

4.2 Rationality Verification

In this section, we try to verify whether the vertex probability formula is rational and is consistent with common Web users' actual situation. Because there is no direct related work, we conduct the verification experiment by inviting multiple users to mark our formula's results.

We choose randomly 10 volunteers, who help to score the results of the vertex probability formula. Firstly, we collect their personal information on the Web using search engines. Then the probability of each data item is computed by the formula. As the data set, all data items and their probability values are returned to volunteers. They determine whether data items belong to themselves and the relational degree of corresponding probability values, and give scores between 0 and 10 describing the reasonable degree of each probability value as shown in Table 1.

As shown in Table 1(a), “# of data items” represents the number of items about current user found by search engines. “Initial data items” is known background information provided by users. “Scores distribution” represents user’s evaluation to the probability of data items belonging to him. And the entry “10(6)” denotes there are 6 items to be rated 10 by the current user. We make statistics of the distribution of users’ scores and average score of each user. From the statistical results, the average value of 8 users is 8.284, which represents the probability formula including two impact factors can preferably compute data items’ probability with higher rationality. As Table 1(b) shown, items with higher scores (8-10 points) account for the majority of all data items (112/137). It is also shown that most of users think our formula with higher rationality.

Table 1. Reasonable Experiments

(a) Statistics of all user ratings

User	# of data items	Initial data items	Avg. score	Scores distribution
1	9	phone number	7.875	10(6); 8(1); 2(1); 1(1)
2	13	email	9.231	10(11); 8(1); 2(1);
...
Total	137		8.284	

(b) Statistics of user ratings distribution

Score	10	9	8	7	6	5	4	3	2	1
Frequency	82	17	13	4	2	0	0	5	5	8

These experiments show that our method can help users to understand the degree of their privacy leakage on the Web, and remind user of taking preventive measures.

4.3 Effectiveness Evaluation

In order to verify the PLD-Probability algorithm, we design some simulation experiments.

Firstly, we specify an attribute collection of user data items which includes name, company, address, email, phone number, IM, ID number, username, disease, and so on. These data items are extracted from returned Web pages. Disease items are classified into S information. ID number and name are classified into I information. Other items belong to O information. We simulate to generate some web pages and 1-10 data items for each web page by a program. Then each item is assigned with a value chosen randomly from the value collection of its corresponding attribute. By this way, we can simulate all the data items on different pages for a user and generate the original UICA graph.

The entropy based on the value collection of attributes is a very important impact factor in the PLD-Probability algorithm. Next, we tests the influence of the values’ confusion degree for a specific attribute in the data item set generated by the simulation program on vertices’ probability values. The probability represents the possibility of this data item belong to the user. The variable δ is used to represent the proportion of data items with the same value. The value range of δ is $[0, 1]$. The smaller the value of δ , the larger the entropy value $Entropy(S_\lambda)$. We choose different δ values: $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Based on different entropy values, vertices’ probabilities are computed. The distribution of vertices’ probabilities is shown in Figure 5.

In Figure 5, the regions of different colors represent different probability intervals. E.g. the red region includes the vertices whose probabilities are in interval $[0.2, 0.4]$. When the value of δ is smaller ($\delta < 0.4$), vertices with probabilities in $[0.2, 0.6]$ account

for the majority of all vertices. This means when information is more confusing, it is difficult to determine whether a data item belongs to current user. With the increase of δ ’s value, vertices take on the trend of convergence to the interval $[0.8, 1]$. It is obvious that the purple region (including vertices with probabilities in interval $[0.6, 0.8]$) gradually shrinks until disappearance. The entropy of attributes’ value collections reduces with the increase of δ ’s value, and the probability distribution of vertices takes on the characteristic of convergence. It proves that our formula can effectively identify data items belonging to the current user, and filter noise vertices by assigning them with lower probabilities.

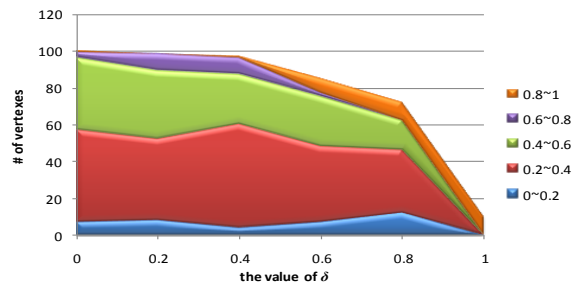


Figure 5. The Influence of Information Confusion on Vertex Probability Distribution

The variables ω_E and ω_D in the vertex probability formula denote the weights of the two factors respectively which have different impact on vertices’ probabilities. We take the *third experiment* to evaluate the weight distribution’s impact on vertices’ probabilities when the value of δ is fixed. We conduct six experiments with different δ values. The results are shown in Figure 6.

In the extreme case of $\omega_E=0, \omega_D=1$, namely there is only the average distance factor impacting vertices’ probabilities, there are too many vertices fall in probability intervals $[0.8, 1]$. That means some noise data items which actually don’t belong to current user are mistaken for too high probabilities and cause the amount of wrong information increases. While, in the extreme cases of $\omega_E=1, \omega_D=0$, there are too many vertices fall in probability intervals $[0, 0.2]$. It means some vertices which actually belong to the current user are mistaken for too low probabilities and misjudged not to be the user’s information. And this weight distribution plan will lead to lose too much useful information in the UICA graph. Based on observations on experiment results, when the weights of ω_E and ω_D are close to each other, the probability distribution of vertices is more reasonable. The vertices with probability in middle interval $[0.2, 0.8]$ account for the majority of all vertices and there are fewer vertices with extreme probabilities in $[0, 0.2]$ and $[0.8, 1]$. It is basically consistent with the actual situation. In summary, the overall solution proposed by this paper is reasonable and effective.

5. RELATED WORK

There are many privacy concerns on the social networks. Social network sites usually provide some mechanisms that can enable users to restrict access to friends’ list or circle of trust [6]. For secure personal web content sharing in such networks, [4, 5, 6, 7, 8] proposed many methods to protect users’ information from being accessed by unexpected visitors except friends’ list or circle of trust. Most of these methods are based on authentication mechanism and encryption. But the protection mechanism of these works is different from our methods. The goal is to prevent

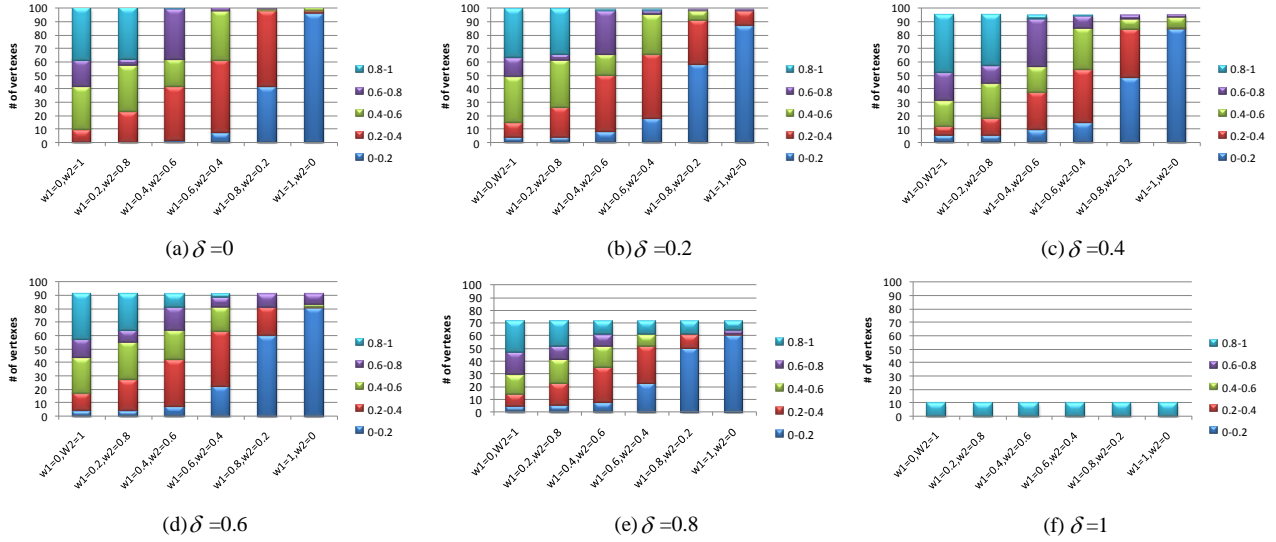


Figure 6. Weight ω_E and ω_D Distribution Experiment

attackers not in circle of trust from accessing or obtaining users' published information in social networks. However, we focus on all searchable information about users on the Web. Our work aims at protecting user's identity information and privacy-sensitive information from being collected together by malicious attacker. In addition, social network can be regarded as a part of Web, and the information just comes from the social network, while our work is to solve the problem of information inferring and collecting on the whole Web. The quantity and scope of information in our problem are greater.

There is also some related work [9, 10] on finding entities' relationship on the web and [11] proposed a set of techniques for finding terms that are correlated to one or more query terms. In this kind of research works, they find the relationships between two entities via search engines. In contrast with our problem, from the perspective of attackers, it collects all information about the victim together.

6. CONCLUSION REMARKS

In this paper, we introduce a new family of privacy attacks on the Web: privacy inferring attack via search engines. We present a privacy inferring model to describe the process and principles of personal privacy inferring attack via search engines. To solve this problem, we propose a user-side automatic detection service to detect the privacy inference before publishing personal information. In this user-side service, we construct a user information correlation graph and propose a Privacy Leakage Detection Probability (PLD-Probability) algorithm. Extensive experiments show that our methods are reasonable and effective.

7. ACKNOWLEDGMENTS

This research was partially supported by the grants from the Natural Science Foundation of China (No. 61070055, 91024032), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (No. 10XN1018), National Science and Technology Major Project of Key Electronic Devices, High-end General-purpose Chips and Fundamental Software Products (No. 2010ZX01042-002-003), the Graduate Science Research Funds of Renmin University of China (No. 11XNH121).

8. REFERENCES

- [1] T. McIntosh and J. R. Curran, "Weighted Mutual Exclusion Bootstrapping for Domain Independent Lexicon and Template Acquisition," in *Proceeding of the Australasian Language Technology Workshop*, Hobart, Australia, 2008.
- [2] L. Sweeney, "K-anonymity: A model for protecting privacy," *International Journal on uncertainty, Fuzziness and Knowledge-based System*, vol. 10, pp. 557–570, 2002.
- [3] P. Samarati and L. Sweeney, "Generalizing data to provide anonymity when disclosing information," in *PODS98*.
- [4] C. Dwyer and S. R. Hiltz, "Trust and privacy concern within social networking sites: A comparison of facebook and myspace," in *Proceedings of AMCIS 2007*, Colorado.
- [5] R. Feizy, "An evaluation of identity on online social networking: Myspace (poster)," in *ACM Hypertext and Hypermedia (HT)*, 2007.
- [6] M. Mannan and P. C. van Oorschot, "Privacy-enhanced sharing of personal content on the web," in *Proceeding of the 17th international conference on World Wide Web (WWW'08)*, Beijing, China, 2008, pp. 487–496.
- [7] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *AsiaCrypt*, 2000.
- [8] R. Gross and A. Acquisti, "Information revelation and privacy in online social networks," in *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2005.
- [9] G. Luo, C. Tang, and Y. li Tian, "Answering relationship queries on the web," in *Proceeding of the 16th international conference on World Wide Web (WWW'07)*, Banff, Canada, May 2007, pp. 561–570.
- [10] F. L. Sanda Harabagiu and A. Hickl, "Answering complex questions with random walk models," in *SIGIR'06*, 2006, pp. 220–227.
- [11] V. K. P. Tan and J. Srivastava, "Indirect association: Mining higher order dependencies in data," in *PKDD'00*, 2000, pp. 632–637.