Web User De-Identification in Personalization

Jiaqian Zheng, Jing Yao and Junyu Niu
Department of Computer Science and Engineering
Fudan University
220 Handan Road, Shanghai 200433, China
{jqzheng, 062021115, jyniu}@fudan.edu.cn

ABSTRACT
It is a kind of privacy infraction in personalized web service if the user profile submitted to one web site transferred to another site without user permission. That can cause the second web site easily re-identify to whom these personal data belong, no matter whether the transfer is under control or by hacking.

This paper presents a portable solution for users to bind their sensitive web data under the appointed domain. Such data, including query logs, user accounts, click stream etc, could be used to identify the sensitive information of the particular user. By our domain stretching de-identification method, if personal data leak from domain A to B, the web user could still not be identified even though he logins to sites under domain B using the same name and password. In the experiment implemented by javascript, we show the flexibility and efficiency of our de-identification approach.

Categories and Subject Descriptors
H.3.m [Information Storage and Retrieval]: Miscellaneous

General Terms: Algorithms, Security

Keywords: de-identification, domain, user profile

1. INTRODUCTION
Network security consists of two aspects generally. (a) How to choose safe methods and routes to deliver the data. The safety means the data should keep confidentiality, integrity and efficiency in the transmission. There have been lots of well-rounded encryption theories and algorithms to ensure the security in routes. (b) How to store and use the data properly at receiving ends. Data receiver usually stores the data in database and set the administrative permission carefully. Once data submitted to web sites, users are no longer able to control them from disclosure. So we have to rely on second parties to protect the data properly.

Sometimes we may not feel fully comfortable if our sensitive personal information stored in someone else's disks. In the scenarios such as personalized web search and new auto-recommendation system which require users to submit their profiles to supply better web services, we prefer these web sites processing personalized stage on the client side instead of doing so on the server side.

For example, as a promising way to improve search quality, personalized web search essentially plays the role to select the set of results that match each individual's unique information retrieval goal. The crucial stage of personalized web search is to re-ranking general query results by data representing current user bias or preference, such as user profiles, click stream and navigation histories. Since these data can mostly contain personal privacy, the ideal solution is to handle this stage locally.

However, re-ranking on the client side is bandwidth intensive because web corpus is on the server side, and it requires a large number of search results transmitted to the client before re-ranking. For this reason, most of personalized search services online like Google Personalized Search and Yahoo! My Web adopt the approach to re-ranking on the server side alternatively, by sending personal profiles and queries together to the search engine. The factor is that large percents of search engines prefer caching these profiles in persistence to save data transfer cost on network, which exposes latent privacy issues.

The privacy issues do not occur only in web search. It is the common risk behind all types of personalized web services such as news subscription systems, online shopping recommendations. People have to give up some privacy to gain economic benefits from personalization. Once users expose their personal data to one site, they will be out of control how the web site caches and delivers their profiles.

2. METHODOLOGY
In practice, however, privacy is not absolute. Personalization can be the good example itself where people give up some privacy to gain better services. It is unreasonable to ensure full security in aspect (b). Otherwise, user profiles transmitted to web site would be helpless to build personalization. The significant work targets at bridging the conflict needs of personalization and privacy protection.

Related work proposed by *Yabo et al.*, presents the method that users can control the degree of the personalized service by choosing how detailedly their personal data are exposed[1]. For the purpose of selectively exposing users' interests to the personalized service, the user profile is built within term-based hierarchical structure by applying term frequency clustering algorithms. Users can protect themselves by avoiding exposing too many details about their interests and web histories.

2.1 De-Identification Approach
Our work changes the view from user level to domain level, to answer how to restrictively expose users' identification under the certain domain. To bridge the conflict between personalization and privacy security, our approach grants users the ability of binding their personal identifiable data on the web sites under the specific domain. In other words, users are likely to feel comfortable if they can be only recognized in web sites permitted by themselves. If personal data happened to leak, users would

Copyright is held by the author/owner(s).
WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

tolerate that other sites were able to get the contents of their preferences but unable to re-identify the owners of these data.

Here, we assume that a user permits one web site to recognize him if the user submitted his profile to it. We divide the data in user profiles required in personalized web services into two parts. One is called the identifiable data such as user names, account IDs and identity card numbers. These data can uniquely identify which person one profile belongs to. But they are useless to build personalized services. Another part of data purely present user interests and preferences, which may not be so sensitive as identifiable data since web users will not be easily identified from them.

The basic idea of our method is to encrypt identifiable data with the domain salt which makes it impossible to map users with their profiles under different domain sites. The non-identifiable data remain unchanged so that personalized stage can handle them without decryption steps. Then the key point is to find a suitable domain stretching algorithm encrypting identifiable data properly before user profiles submitted to the web site.

2.2 Domain Stretching Method

Ross et al. proposed a common stretching algorithm that uses a domain name as a salt and *HMAC* as F [2]:

$$K_{long} = HMAC(K_{short}, dom)$$

where *dom* is the domain name. *HMAC* is originally used to stretch a weak key into strong one. We can input the K as identifiable data here to de-identify a user account. However *HMAC* does not match the dissimilar character:

$$if\ K_A \neq K_B \Rightarrow HMAC(K_A, dom) \neq HMAC(K_B, dom)$$

The character constrains that different users cannot be recognized as the same one after de-identification. But *HMAC* algorithm does not match the requirement since it builds on hash functions essentially.

Since identifiable data usually do not contain large bytes, we employ RSA as the basic encryption algorithm. Client generates the pre-defined RSA public key $PK=\{e,n\}$ and destroys the private key $SK=\{d,n\}$. The browser can store PK as local settings for the sake of next access. In the process of navigation, once the web site A requires $user_i$'s profile to continue, the encryption module on the client side intercepts the submitting event, replaces identifiable fields (e.g. username and password) in profile with the algorithm output below:

$$KEY_{de}(user_i) = RSA_{PK\{e,n\}}(\{identifiable_i\}, salt) + e$$

In the definition above, KEY_{de} represents de-identified user account and the *salt* inputs the domain string. Of course RSA matches the requirement of dissimilar character because it contains the code which can be decrypted into original key. The reason we appending e after the KEY_{de} string is to prevent the less possibility that different users under different PKs have the same RSA encrypted results by accident. The server side can authorize and validate users' login by purely comparing KEY_{de}s instead of by usernames and passwords, without any more changes at the back-end. Since the SK has been destroyed, even though PK releases to public, the information fields name and password that can identify the web user would not been decrypted. Attacker could not impersonate $user_i$ either only by $PK_{\{e,n\}}$ without plain account data. For this reason, $user_i$ can declare his PK, on the public blog etc., to obtain the same authorization to access a web site on different PC clients.

In the scenario of the profile being exposed to a third party, the user would not be identified because his identifiable data are encrypted in it. Although the user may login into the third party's web site within the same name and password, he would never be recognized and mapped into his original profile.

2.3 Cross-site Enhancement

In some scenarios, a big company may hold several web sites, and the company prefers sharing user accounts on different web sites. This enhancement can be simply implemented in our de-identification framework. In the definition of KEY_{de}, all web sites of one company can declare a same domain string as the specific *salt*, adapting for cross-site flexibility. However, attachers in domain B may declare a fake domain address to obtain KEY_{de} under domain A. Our method prevents this kind of attack by validating the declared domain string with its original URL. For example, site "*sales.dell.com*" and "*research.dell.com*" are permitted to declare their domains "*dell.com*" but not to "*ibm.com*". Finally, figure 1 briefly illustrates the proposed de-identification framework.

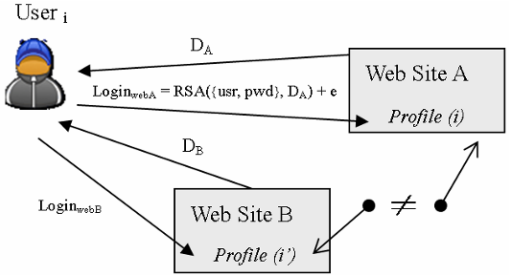

Figure 1. De-identification Stretching Method.

$User_i$ submits his profile to web sites A and B within identifiable data stretched by different domain salts: D_A and D_B, which ensure no site can match *profile(i)* and *profile(i')* as equal, even if the profile happened to be exposed spitefully.

2.4 Experiment

Based on the general RSA library[3], We implemented the de-identification domain stretching method purely in javascript on the client side to prove the feasibility and efficiency[4]. We encrypted user identifiable data with the key size of 1024 bits. Since the encryption algorithm runs on the client side, no extra burden increases on personalized web servers. Nearly all user profile submissions response in 0.5 second.

3. REFERENCES

[1] Yabo Xu, Benyu Zhang, Zheng Chen and Ke Wang, Privacy-Enhancing Personalized Web Search. International World Wide Web Conference, WWW 2007. May 8-12, 2007, Banff, Alberta, Canada.

[2] Blake Ross, Collin Jackson, Nicholas Miyake, Dan Boneh, and John C. Mitchell. Stronger Password Authentication Using Browser Extensions. Proeedings of the 14th Usenix Security Symposium, 2005

[3] Javascript RSA library: http://www.ohdave.com/rsa

[4] Demo: http://www.cs.fudan.edu.cn/wimlab/deidt