PROJECTE FINAL DE MASTER

EVALUATION OF THE PRIVACY RISK FOR ONLINE SEARCH AND SOCIAL TAGGING SYSTEMS

***Estudis**: Máster en Ingeniería Telemática*

***Autor**:* Ana Fernanda Rodríguez Hoyos

***Director**/a:* Jordi Forné Muñoz, PhD

Co-Director: Javier Parra Arnau, PhD (c)

***Any**:*2013

Index

Index of figures

Agradecimientos

El presente proyecto de titulación va dedicado a la memoria de mi querido hermano *Héctor Alejandro Rodríguez Hoyos*, gracias ñaño por ser el regalo más valioso de nuestra familia. Tu recuerdo y ejemplo de vida siempre permanecerá presente en nuestros corazones.

Quiero expresar un profundo agradecimiento a Dios por su inigualable amor y sus constantes bendiciones en mi vida. Gracias Dios por bendecirme con mi mayor tesoro, mi familia, siempre estaré agradecida con mis padres Inés Hoyos y Héctor Rodríguez; su amor, esfuerzo y dedicación en mi formación académica y personal han sido fundamentales en la culminación de cada logro profesional.

Un agradecimiento especial a mi amado esposo José Antonio Estrada, gracias mi vida por ser mi compañero y mi ayuda idónea en esta etapa de mi vida. Su amor, comprensión y paciencia fueron cruciales para hacer realidad este sueño.

Agradezco las enseñanzas y la valiosa guía brindada por mi director y co-director de tesis, el profesor Jordi Forné y Javier Parra por su dirección y apoyo en la elaboración de este proyecto.

Para finalizar, quiero agradecer el apoyo de la Senescyt, Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador, de la que recibí el financiamiento, con beca completa, para realizar mis estudios de maestría en la Universidad Politécnica de Catalunya.

Resum del Projecte

La manca de privacitat és actualment un greu problema de seguretat per als usuaris d'Internet, ja que els serveis d'informació personalitzats recol · lecten tal quantitat d'informació que semblaria que aquests ens coneixen millor del que ens coneixem nosaltres mateixos. Cada pas que donem a la web pot ser detectat per les companyies que proveeixen els serveis més populars com a motors de cerca i xarxes socials. Les consultes de cerca, les etiquetes, els clics, els missatges de correu electrònic, els twits poden ser utilitzats per obtenir un perfil precís de la nostra activitat. Encara que l'obtenció de perfils en base a aquesta informació és una font gegantina de risc de privacitat, només unes poques eines ofereixen un mecaniso de protecció que prengui en compte aquestes dades d'usuari.

TrackMeNot és una d'aquestes eines, que implementa pertorbació de les consultes de cerca d'usuari, mitjançant la generació de consultes falses, per tal de ofuscar el perfil d'usuari. No obstant això, així com succeeix amb altres aplicacions, no hi ha certesa de l'eficiència d'aquesta eina en la protecció de la privacitat.

En aquest treball, inicialment contribuïm amb el treball realitzat en [8], on es mesura la privacitat de l'usuari, interpretada i mostrada per l'usuari en el navegador web Firefox mitjançant una extensió anomenada PrivMeter. La privacitat en aquesta proposta es calcula basant-perfil generat a partir de les consultes que realitza l'usuari en els principals motors de cerca. Però, en ser les etiquetes socials peces importants d'informació d'usuari, desenvolupem alguns mòduls que consideren aquestes etiquetes en el procés de mesurament de privacitat.

A més, avaluem també el mecanisme d'ofuscació ofert per TrackMeNot en el seu afany de protegir la privadesa dels usuaris. Mesurem l'eficiència de TrackMeNot en la millora de la privacitat de l'usuari. Trobem que, davant d'atacs d'identificació, aquesta eina millora de manera important la privacitat de l'usuari. No obstant això, davant d'atacs més sofisticats, com els de classificació, el mecanisme d'ofuscació no és prou reeixit. La forma de generar les consultes, en aquest cas, és crucial per ofuscar el perfil d'usuari de manera eficient davant d'atacs de classificació.

Finalment, s'integra TrackMeNot amb l'eina PrivMeter de manera que les mètriques de privacitat corresponents tant al perfil real com aparent de l'usuari es mostrin a la interfície.

Keywords: extensió de navegador, obtenció de perfils, mètriques de privacitat, etiquetatge social, privacitat d'usuari, ofuscació de consultes.

Resumen del Proyecto

La falta de privacidad es actualmente un grave problema de seguridad para los usuarios de Internet, ya que los servicios de información personalizados recolectan tal cantidad de información que parecería que éstos nos conocen mejor de lo que nos conocemos nosotros mismos. Cada paso que damos en la Web puede ser detectado por las compañías que proveen los servicios más populares como motores de búsqueda y redes sociales. Las consultas de búsqueda, las etiquetas, los clics, los mensajes de correo electrónico, los twits pueden ser utilizados para obtener un perfil preciso de nuestra actividad. Aunque la obtención de perfiles en base a esta información es una fuente gigantesca de riesgo de privacidad, solamente unas pocas herramientas ofrecen un mecaniso de protección que tome en cuenta estos datos de usuario.

TrackMeNot es una de esas herramientas, que implementa perturbación de las consultas de búsqueda de usuario, mediante la generación de consultas falsas, con el fin de ofuscar el perfil de usuario. Sin embargo, así como sucede con otras aplicaciones, no hay certeza de la eficiencia de esta herramienta en la protección de la privacidad.

En este trabajo, inicialmente contribuimos con el trabajo realizado en [8], donde se mide la privacidad del usuario, interpretada y mostrada para el usuario en el navegador web Firefox mediante una extensión denominada *PrivMeter*. La privacidad en esa propuesta se calcula en base a perfil generado a partir de las consultas que realiza el usuario en los principales motores de búsqueda. Pero, al ser las etiquetas sociales piezas importantes de información de usuario, desarrollamos algunos módulos que consideran estas etiquetas en el proceso de medición de privacidad.

Además, evaluamos también el mecanismo de ofuscación ofrecido por TrackMeNot en su afán de proteger la privacidad del usuario. Medimos la eficiencia de TrackMeNot en el mejoramiento de la privacidad del usuario. Encontramos que, frente a ataques de identificación, esa herramienta mejora de manera importante la privacidad del usuario. Sin embargo, frente a ataques más sofisticados, como los de clasificación, el mecanismo de ofuscación no es suficientemente exitoso. La forma de generar las consultas, en este caso, es crucial para ofuscar el perfil de usuario de manera eficiente frente a ataques de clasificación.

Finalmente, se integra TrackMeNot con la herramienta *PrivMeter* de manera que las métricas de privacidad correspondientes tanto al perfil real como aparente del usuario se muestren en la interfaz.

Keywords: extensión de navegador, obtención de perfiles, métricas de privacidad, etiquetado social, privacidad de usuario, ofuscación de consultas.

Abstract

The lack of privacy is nowadays a serious security issue for users on the Internet, since personalized information systems are in fact collecting an incredible amount of information, to the point where it appears they know us better than ourselves. Every single step we take on the Web can be detected by the same companies that are providing the most popular services, such as search engines and social networks. Search queries, tags, clicks, e-mails, tweets can be used to get an accurate profile of our activity. Even when the profiling of this information is a huge privacy risk, only a few tools propose a protection mechanism taking this into consideration. TrackMeNot is one of these tools, which implements perturbation of a user's queries, by generating fake ones, to obfuscate the user's profile , although, similarly to other tools, it is not immediate to evaluate to what extent it is protecting users privacy.

We have firstly contributed to the work made in [8], where the user's privacy is measured, interpreted and showed for the user in the Firefox web browser. Privacy is computed based on the search queries of the user. But, since social tags are an important component of users' data, we have developed some modules to take these into consideration in the process of measuring privacy, and not only the search queries.

Additionally, we also evaluated the obfuscation mechanism offered by TrackMeNot to protect user's privacy. By implementing justified privacy metrics we measured the efficiency of TrackMeNot to enhance the user's privacy. We found that for identifying attacks TrackMeNot importantly improved the user's privacy. However, for more sophisticated attacks, such as classification attacks, the obfuscation mechanism was not successful enough. The way of generating fake queries is crucial to efficiently obfuscate the user's profile against classification attacks.

Finally, TrackMeNot was integrated with PrivMeter in such a way that the privacy metrics are showed both for the real and obfuscated user's profiles.

Keywords: browser extension, profiling, privacy metrics, social tagging, user's privacy, query obfuscation.

1. Introduction

1.1 Motivation

There is a digital revolution in the world, which is specially deepened by the re-imagination of content generation on the Internet. Users' experiences that involve information are now susceptible to be reproduced on the Web. This phenomenon is possible thanks to innovation, not only related to the way of conceiving digital devices, but also related to the way of generating the information flow. Clearly, the massive usage of mobile devices combined with the emergence of collaborative applications, have initiated an (even more) exponential growth of the Internet. User-generated content is now massive; every minute Google receives over 2 million of search queries, users add 3125 new photos on Flickr, Twitter users send over 100 thousand tweets and Facebook users share over 600 thousand pieces of content [4]. That means that the information flow from the user is much greater than before.

But collaboration is not the only experience that encourages users to be active agents on the Internet. Personalized information systems, that offer content tailored to the users' interests, are being integrated to all of the most popular Internet services, in order to offer a customized experience.

The combination of these new tools generates a more interactive and cooperative experience on the Internet. Being humans social by nature, it explains the success of the most popular social networks that implement this combination of strategies.

Search engines and social networks are, undoubtedly, the most used services on Internet today. These systems have become the gates to basic services as blogs, news sites, etc. It is common to find users accessing government websites or news pages only by clicking on a link found as a reference in a tweet or in a Facebook post, but rarely by directly writing the corresponding URL in the web browser.

Among the main search engines, Google is the dominant service. It receives millions of search queries everyday with regard to user interests. Based on these queries the user is then redirected to web pages with related content. Social applications coexist in a more heterogeneous environment, but not less popular. Social networks such as Facebook or

Twitter, and others oriented to tagging like Delicious or Flickr, receive millions of content contributions from users.

With such amount of information flowing from users during the last years, data privacy has become an important concern, mainly because of personalization. We are seeing the effects of personalization in a daily basis; when Facebook recommends us friends and products, when Twitter suggests us to follow some politician and when Google corrects our search queries to tailor our requests to what we "really" want. Users are being continuously profiled on the Internet and the raw material used for this profiling process is every query, tag, and click sent on the Web.

Search engines, social networks and, in general, every web service is collecting search queries, tags, clicks and more metadata that, if they were publicly revealed, it may cause a great harm. It already happened that companies like AOL [5] and Netflix [6] revealed these logs for research purposes. These releases were immediately followed by technical analyses demonstrating that even when the datasets were anonymized, privacy of users could be compromised.

The fact is that the amount of user information (traces) delivered to the Web is so great and of such quality that linking it to the corresponding owners could be relatively easy [7]. Internet services like search engines, social networks and social tagging systems collect information (tags and search queries) that could directly represent user interests.

With such technology and information about users, privacy is hardly guaranteed, especially when the creators of these services are not thinking of stopping, since profiling, they say, is the key of the quality they are offering.

The biggest problem is that, nowadays, this appears to be just the tip of the iceberg since many governments apparently have direct access to private information of the citizens stored by the main service providers [1]. The collected material could be: search history, the content of e-mails, file transfers and even live chats, which is basically all the information that ordinary users interchange on the Internet.

Although privacy has been considered a fundamental human right since a long time ago, these privacy issues have really attracted enough attention in society only during the last few years. Thus PETs (Privacy-Enhancing Technologies) have appeared as technical means for protecting user's

privacy. The problem is, though, that the concept of privacy is related to a multidimensional notion. For that reason, although there are many tools and theoretical proposals to protect user's privacy, none offers a measurable strategy. Thus, it is difficult to know if such mechanisms are truly reducing the risks that users face on Internet, given that they do not delineate what kind of privacy is being protected.

Measuring the efficiency of privacy protection technologies is, hence, critical for the user to be able to compare them. Provided that these approaches commonly affect the utility of the systems where those are applied, an idea of how much privacy is gained with a given technology would help to evaluate its suitability for a particular user environment. So measuring the privacy of a user before and after applying a PET will be useful to calculate such gain of privacy.

Measuring privacy requires the construction of a precise user model to represent his privacy level, in the same way as user profiles are built by personalized systems. This means to collect a lot of user information.

Then, measuring privacy seems to be the first step towards implementing effective means of protection. But, having a wide vision of users profiles, similarly to how those are seen by adversaries, is essential to model the environment where privacy is measured. Finally, measuring privacy supports the evaluation of already deployed privacy protection tools to know if such heuristic mechanisms provide the expected benefits.

1.2 Objectives

Part of this work includes a complement of the privacy measuring tool presented in [8] where *PrivMeter* is developed as a Firefox extension to measure, interpret and show the user's privacy. *PrivMeter* shows the privacy risk level of the user essentially by processing search queries that are sent to the most popular search engines. In that sense, we want to initially contribute to this tool by incorporating mechanisms to measure the user's privacy in social tagging environments where tags are now considered as inputs to the measuring process.

Also, by leveraging the same metrics justified in [9] and used to measure privacy in [8], we want to evaluate the benefits of a widely known privacy protection tool called TrackMeNot. We also propose integrating TrackMeNot with *PrivMeter* so that the evaluation of the privacy protection tool can be done continuously in the user's interface.

The proposal, initially, involves adapting browser extensions in order to capture (in the user side) the user's information going to social tagging services. Since this information is clearly identifying user interests and preferences, it could be used by attackers to build a profile. In the same way, this information can be leveraged in the user side, to obtain a user profile whose level of privacy can be measured.

Last, but not least, an evaluation of TrackMeNot benefits is proposed. Being a popular Firefox extension to protect the user privacy, it is fundamental to assess if its strategies are having the wished effects regarding privacy.

1.3 Organization

This work is organized in the following way. Available privacy enhancing technologies are introduced and classified in Chapter 2. Chapter 3 summarizes the mathematical criterion used to measure the privacy of user profiles. Chapter 4 details the contributions on measuring user privacy in a Firefox web browser by using social tagging data to build the user profile. The evaluation of TrackMeNot mechanisms to protect privacy is made in Chapter 5. Conclusions and future work can be found in Chapter 6.

2. Privacy-Enhancing Technologies

PETs are essentially technical means for protecting user's privacy [10]. But, privacy is a very wide concept that involves many approaches, from the solely characteristics of the communication traffic to the content of transmitted messages.

An interesting contribution is made in [11] by analyzing the privacy protection levels a PET could offer, depending on the amount of user information that personalized search engines have collected. Based on these ideas, privacy-enhancing technologies may also be classified in basic anti-tracking technologies, cryptography-based methods, TTP-based approaches, collaborative mechanisms and data-perturbative techniques. This classification contemplates that combining all of these technologies will result in having the highest level of privacy.

2.1 Basic anti-tracking technologies

Tracking is a mechanism through which an entity is able to identify another one in the process of data communication. Tracking modules are fundamental components of personalized services since they enable the providers to match identities with their corresponding preferences.

Tracking could be implemented at different levels of communication. An ISP can identify a user message from the source IP address (network level), whereas a personalized service provider may also use a cookie (application level).

There are some methods to prevent attackers to track a user, most of them hiding or even blocking the parameters employed to identify the user.

If a user connects to Internet by means of any kind of proxy or masquerading NAT, they have by default some degree of privacy protection since he is hidden in a group with other users behind the intermediate device. The attacker outside on Internet would see the same source address in messages originated from all the users in the group.

Blocking the usage of HTTP cookies in the browser is another alternative to prevent tracking, and then hindering the profiling process. The problem

of doing this is that the user shall not receive personalized services and, in some cases, other web services will be blocked too.

Completely blocking tracking reduces the utility of the personalized services to the minimum so it does not seem to be realistic to apply these methods, although the user would reach the maximum level of privacy.

2.2 Private information retrieval (PIR)

PIR is a technology that enables the user to retrieve information from a database in such a way that the database provider is not able to know the content retrieved [12].

A naive way to provide this functionality could be by allowing users to download the complete database so that they can look up the information by themselves. This solution is by no means practical due to the huge amount of information it requires to be transferred. Since the database provider does not know neither the queries nor the corresponding answers, users cannot be profiled by the provider so personalized services are unfeasible.

PIR mechanisms relying on cryptographic techniques use homomorphic encryption schemes and P2P communication protocols [13]. These technologies were commonly studied for environments of recommendation systems. These mechanisms consist on not revealing user profiles in an individual basis, but generating an aggregated profile from the information of a group of users. Attackers, thus, cannot see individual profiles but users can still have a personalized recommendation by locally computing it.

2.3 TTP-based mechanisms

Using a third trusted party (TTP) as an intermediary between the user and the personalized information system is another option to provide privacy protection.

The user sends all his personal information to the TTP and the TTP forwards the data on behalf of the user. Hence, the external system is not aware of any user ID because messages appear to be originated on the TTP.

Not identifying the user is, again, a restriction for personalization services. A solution to this problem could be to provide pseudonyms to the external system. These pseudo IDs are matched to the real IDs only in the TTP, which is the only one knowing the real identity of the user. However, the collusion of the TTP and the external system would threaten user privacy.

An extent of this approach could consider the emission of digital credentials for the users so that the TTP is not compelled to be always online when user transactions are sent to the external system.

Another drawback of these technologies is the fact that, provided that personalization is possible, profiling activities would finally allow the system to identify the user. The queries or tags in the information sent by the user could be enough to break the user privacy.

These mechanisms consider user privacy in an environment of user content analysis. But simple parameters regarding network traffic (e.g. direction) could be analyzed by an attacker to reveal personal details of the user, such as its location.

Mixes [14] appeared a long time ago, as systems that receive a message and then forward it in such a way that the input event cannot be linked to the output event. An attacker is, therefore, unable to follow a message path by "listening" forwarding events in the routers. To accomplish such objective, a mix device changes the appearance and the flow of messages as they goes to the destination.

Mixes employ a technique that generally consists on keeping received messages in the mix to then randomly forward them only when a predefined number of messages have arrived. Other variations are also feasible by introducing, for example, thresholds for the time that a message is kept in the mix, or for the number of messages released.

In order to overcome the problem of a single point of failure, a network of mixes could be implemented so that an adversary has to compromise the whole path to have a successful attack.

The disadvantage of using mixes is, clearly, the delay introduced during the process, which could dramatically reduce the utility of the whole system.

A last technology based on TTP is onion routing, which consists on the user sending a message that is encrypted several times. As the message

arrives to each onion routing intermediate node, it gets progressively decrypted, so the nodes do not know what is been transmitted. Provided that there is no mixing processes, delay is not introduced and the system is again vulnerable to timing attacks.

2.4 User collaboration

This approach considers user collaboration to enhance privacy. In [15] Crowds is proposed as a classical example of this mechanism and works as follows. When sending a message, a user forwards it to the destination with probability *p*. With probability 1-*p* the message is forwarded to another user, not necessarily in the path towards the destination.

Anonymity, with Crowds, is provided at the level of intermediate nodes and final recipient, but at the expense of traffic overhead and delay generated by the randomized decisions to reach the destination.

Another approach of privacy protection relying on user collaboration is described in [16] for personalized Web search environments. It poses a P2P protocol and follows the Crowds methodology but by grouping users with similar interests based on social networks.

Also in the field of web search, [17] proposes user collaboration between users to protect their privacy by exchanging a portion of user queries before submitting them. The idea is to obfuscate their interest profiles against external systems.

A representative proposal of user collaboration is made on [18] for Location Based Services. When submitting a request to an LBS, users send queries including the location data to which those queries refer. The objective is to cloak the specific location of the user by not sending the exact location data. Firstly, some neighboring peers are grouped in order to collaborate with each other. When a user wants to send a query, he randomly selects one of the members of the group to which the query is forwarded to reach the destination. The location attached to the query includes the coordinates of the whole region where the group members are located.

Distribution, however, is another technique that tries to overcome the risks of storing private information in a single repository. The idea is to have users that exchange private data in a P2P network.

2.5 Data perturbation

It is another technique to obstruct the attacker when trying to obtain a precise user profile. Sending bogus data together with real user data is an example of data perturbation.

Data perturbation mechanisms are commonly located in the user side so that the user does not need to trust any external system. Sending additional data poses another trade-off between the data utility and privacy level. Data utility gets reduced when fake content is introduced to reduce the risk of user profiling.

Query forgery is a very illustrative approach of data perturbation where forged queries are generated on behalf of the user. On receiving both real and bogus queries, the search engine (here the adversary) would not be able to obtain an accurate user profile since it is obfuscated. The trade-off inherent to these methods between privacy and additional traffic is analyzed in [19] where optimization of query forgery is studied.

A widely known implementation of query forgery is TrackMeNot [20] which is a web browser extension generating fake queries and sending them to different search engines on behalf of the user. This application relies on RSS content to obtain keywords that will be later used to generate fake queries. There are also some interesting options to modify the way in which the queries are sent, so that the user behavior can be accurately mimicked.

GooPIR [21] is another proposal at application level that offers privacy protection by doing query obfuscation. When a user query is sent, GooPIR obtains keywords with the same frequency of use that the user query has. These keywords are attached to the query to obfuscate it. However, it is difficult to obfuscate very sensitive queries related to health condition or political affinity. In the same way as TrackMeNot, GooPIR is vulnerable to semantic analysis attacks over aggregated queries that could reveal the real interests of the users.

Applied in recommendation systems, there are some proposals to protect users' privacy from the risks of sharing ratings with the recommender entity. The mechanism described in [22] poses an algorithm where users send perturbed ratings to the recommender so that it cannot realize the user preferences. In order for the rest of users to be able to use the genuine rating, they calculate it with the information received from user who sent

the rating. The problem of this application is that the recommender could still learn about user preferences from the user activity itself along the recommender site (searches, clicks, accessed web pages, etc.).

3. Privacy metrics[1]

This chapter describes the theoretical analysis used as basis for measuring privacy in this work and basically summarized the ideas proposed in [9].

When evaluating a PET, quantifying user privacy could be very helpful since it provides tools to compare privacy levels before and after a mechanism has been applied.

As precisely indicated in [9], and summarized in [8], defining the environment where privacy is measured is fundamental to delineate the scope of the privacy metrics. In security analysis this scope is strongly determined by the capabilities of the adversary. Modelling the user is, yet, necessary since the user privacy depends also on the user skills and the type of personal information capable of threaten his privacy.

3.1 User profile model

When doing a security analysis, we must consider the profile of the victim seen by the attacker, in order to envisage the parameters susceptible to be abused.

Users, in the scenario proposed for this project, send search queries or, more generally, keywords (queries or tags) to a service on the Web (search engines, social networks, etc.). These queries, but particularly the tags, represent concepts on which the user is interested. A few of these queries may easily reflect the user's concern about a particular health condition or his passionate wish to have a child. Service providers can automatically process these keywords to create a precise profile of the user which is then used (they say) only to enhance the personalization of the service.

A user profile is usually represented as a histogram of absolute frequencies, as showed in Fig. 1. The histogram is hierarchically structured as a series of categories that (tries to) represent the user interests from his collected keywords. The size of each bar is proportional to the popularity of each category in the user profile.

[1] This chapter and its figures is taken from [8] since both works were done jointly and the theoretical base is the same.

Privacy metrics analyzed in the next sections are based on this user model, also used by several personalized information services.

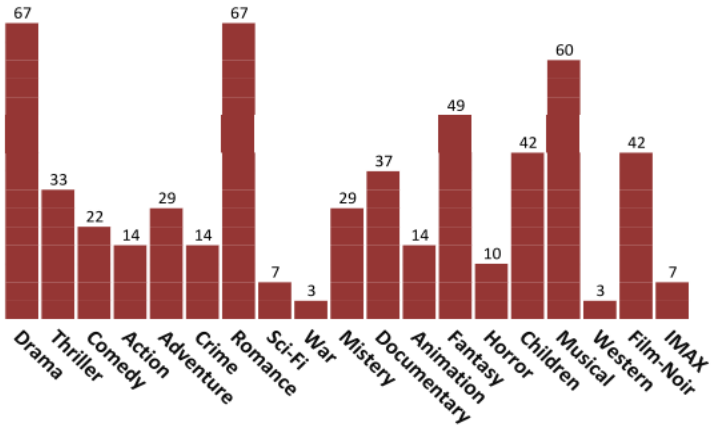


Fig. 1: Histogram modeling a user profile.

3.2 Adversary models

The adversary model defines the properties of the attacker. An attacker is considered as any entity capable of having access to user information in order to obtain his profile.

Defining and describing the adversary is very important since the privacy is quantified according to this entity. Depending on the adversary, a protection strategy could be more or less efficient. This context, when doing security analysis, is important because all threat scenarios cannot be covered, so depending on the hypothesis that are done about the environment (e.g. the adversary), the used metrics will be different.

Considering the histogram of interests as a way to represent the user profile, a user could spoil a profiling strategy by provoking changes in his histogram of interests (e.g. artificially modifying his normal browsing behavior). The attacker would not be able to obtain valuable information, since the user profile would have been perturbed. This new version of the profile, from the user perspective, is known as its **apparent profile.**

Essentially, two attacker objectives are considered, depending on his capabilities, which define the adversary model: *identification and classification*.

Identification. In this case, an attacker is attempting to identify a user in the sense of distinguishing this user from the rest of the population, by detecting deviations among the user's interests with respect to the average profile of the population.

Classification. An attacker tries to classify a user by comparing the user's profile with the representative profile of a group, in order to predict the group in which this user could be catalogued.

3.3 Privacy metrics

As justified in [9], Shannon's entropy and Kullback-Leibler (KL) divergence are used as privacy measuring parameters. Interpretations of these parameters will depend on the hypothesis made, essentially with respect to the adversary model.

Another more general metric, not limited to privacy profiles, is the one proposed in [23]. In this work, the authors propose measuring privacy as the estimation error of an adversary, and interpret, by means of information and bayesian decision theory, other metrics of the state of the art as particular cases of hers.

In order to facilitate the comprehension, the main proposed definitions are exposed below, in order to justify the metrics used for measuring privacy. An interpretation of these metrics is also made to justify their usage as privacy level parameters.

Considering *H* as the Shannon's entropy and *D* as the KL divergence, the entropy *H(p)* of a discrete random variable X with probability distribution *p,* is a measure of its uncertainty, defined as

$$H(X) = -E\log_2 p(X) = -\sum_x p(x)\log_2 p(x).$$

KL divergence, also called, relative entropy *D* (p || q) between two probability distributions *p(x)* and *q(x)* over the same alphabet is defined as

$$D(p \,||\, q) = E_p \log_2 \frac{p(X)}{q(X)} = \sum_x p(x)\log_2 \frac{p(x)}{q(x)}.$$

The KL divergence is a measure of discrepancy between probability distributions, ensuring that $D(p||q) \geq 0$ with equality if, and only if $p=q$. Consequently, it is deduced that entropy *H(p)* reaches its maximum value at $H(u)=\log n$, being *n* the cardinality of the finite alphabet over which *D* (*p* || *u*) is calculated, for a uniform distribution *u*:

$$D(p \,||\, u) = \log n - H(p) \,.$$

Specifically, according to the analysis made in [9], we have that entropy maximization is a special case of divergence minimization, ideally reached when the distribution to be optimized is identical to the reference one.

Being q a user interest profile, t a perturbed (or modified) version of the

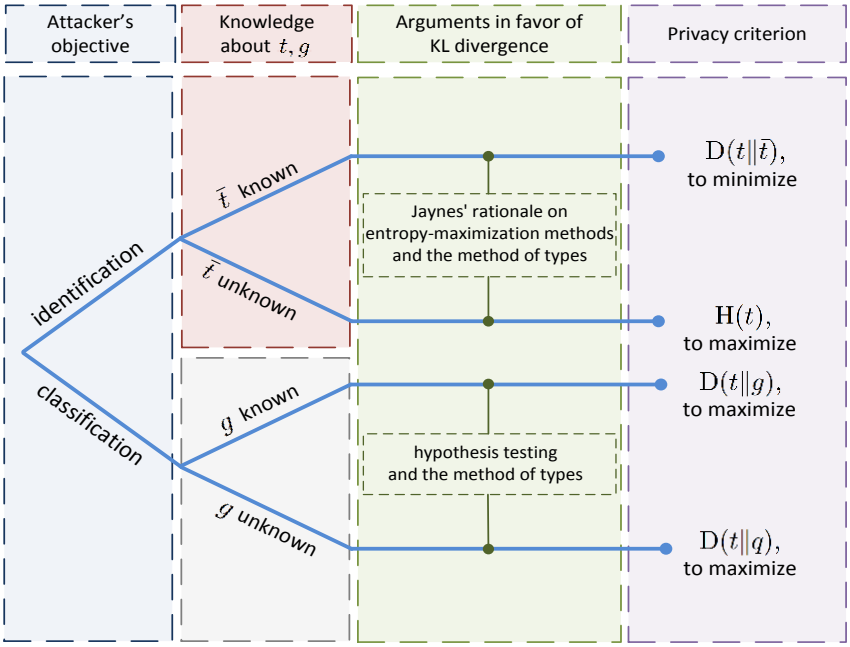


Fig. 2: Summary of the interpretations of Shannon's entropy and KL divergence as metrics of privacy, taken from [9].

user profile, and $\bar{t}$ the distribution of population profile, the interpretations of Shannon's entropy and KL divergence as privacy metrics are shown in Fig. 2. These ideas are deeply detailed, with respect to the attacker's objective, in the following sections.

3.3.1 Metrics against identification

If the goal of the attacker is to identify the user, in the sense mentioned above, Janes' rationale about entropy maximization methods allows the justification of both the divergence and the entropy as measures of privacy.

The entropy of the user's apparent profile, that is, the profile observed by the attacker, is justified in [9] as a measure of the probability of this perturbed profile, in the sense of frequency of occurrence of such profile in the users' population. Considering this probability of the user's profile as a reasonable measure of his anonymity (or privacy), the authors in [9] also justify the entropy as a measure of privacy. In brief, the higher the entropy

of a profile, higher is its probability, and therefore greater is the number of users behaving according to this profile, which means that it is more private.

Furthermore, as it is observable in the first branch of Fig. 2, if the distribution of the average population's profile $\bar{t}$ is known, the divergence between the user's profile *t* and the average population's profile is a metric of privacy, so that, the lower is this divergence, more private can be considered the profile.

To conclude, choosing the best apparent profile, in order to minimize the KL divergence, improves the user anonymity. In simple words, a lower divergence corresponds to a higher frequency of occurrence of such profile, allowing the user to be unnoticed. When having a reference profile of the population, this is the same as maximizing the Shannon's entropy.

3.3.2 Metrics against classification

If the attacker's objective is to classify the user as a member of a particular group, divergence is used as a metric of privacy, according to the analysis done in [9], from hypothesis testing and the method of types. As shown in Fig. 2, in the second branch, if the profile of the group *g* is unknown from the user side, the option is to maximize the divergence between the real profile *q* and the observed (apparent) profile *t*, in order to avoid being classified according to the original profile.

It is necessary to note that in the classification problem, contrary to the identification problem, we are looking for KL divergence maximization, instead of its minimization. The intuition underlying to the cited analysis is that we wish to increase the distance between the user's apparent profile and the real profile, or the group representative profile on which we wish to avoid the categorization.

4. Measuring user privacy for social tagging services

Social tagging services encourage their users to generate personal information of very high quality. So, personalization systems could take advantage of such information to obtain more accurate profiles of their users, with the inherent risk that these practices represent to the user's privacy.

This chapter starts by explaining the impact that social tagging services may have over the user's privacy. Then, we present a summary of contribution made in [8] (where *PrivMeter* is proposed) to finally explain how this privacy measuring tool can be integrated to measure privacy based on the user activity on some social tagging systems

4.1 Social tagging services

Social tags [24] are free text labels that are applied or linked to items such as bookmarks, images, messages, videos, etc. Given that these labels are created by users to represent topics, the interpretation of tags could allow other users to predict the content they represent [34].

These metadata are, thus, very valuable from the standpoint of the quantity of information that they contain about the user. Personalized information systems can leverage these metadata to improve their recommendation services since they are able to more accurately profile their users. The fact is that even the users by themselves can exploit this information by exploring tags when they are publicly available.

This behavior has a direct consequence on privacy since more information about the user is provided to external services. The processes of user profiling are facilitated by the help offered by users themselves when they give a context to their interactions with the Web.

So, particular effort should be put on considering information originated from social tagging interactions as an important input of the privacy measuring process. Namely, if (potential) adversaries are exploiting social tagging, this information should also be considered when measuring the user privacy.

4.1.1 Popular social tagging services

There is currently a lot of services offering tagging mechanisms. Some of them offer tagging as the main component of its operating logic (Delicious, Flickr, BibSonomy, Evernote) and others as an interesting complement (Facebook, Twitter).

Social bookmarking services are a particular type of social tagging platforms, where the user basically adds a record that links a URL (a bookmark) to keywords or tags that, according to the user's perspective, represent the content located in the URL.

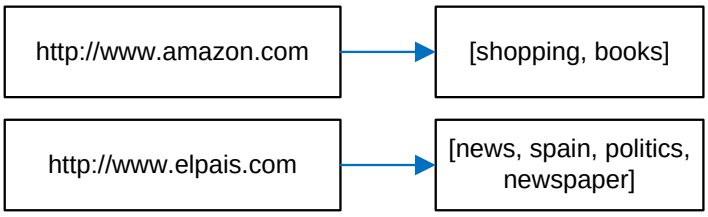


Fig. 3: Process of social bookmarking.

Social bookmarking emerged from the desire to explicitly share information among small groups or communities [24]. It has encouraged the generation of a very content-rich data structure. It is not a coincidence that plenty of research contributions have used datasets crawled from these services to take advantage of the semantic content of their records. The social tagging services that this project works with are the following:

Delicious [25] is a social bookmarking management service. It allows users to add bookmarks that can be categorized or labeled by means of tags. From a user interface functionality point of view, adding bookmarks to these platforms is done in a similar way as to create them into the browser. The advantage is that Delicious provides a service where bookmarks are always available from the

Fig. 4: Record of a bookmark added in Delicious with its corresponding tags.

Web, multiple tags can be attached to the same bookmarks, and

bookmarks can share tags. We cannot forget that, by default, Delicious records are public by default because its aim is to promote sharing of information among users.

Flickr [26] is an online photo service that allows users to share their photos. It is basically the main service that Flickr offers. Another important characteristic is that users can manually mark their photos using tags. In the same manner as in Delicious, the tags describe the content (an image in this case) by providing a context or semantic information.

Youtube [27] is a web site where users can upload, share and watch videos. When a video is uploaded, a representative category has to be attached to the video. As described in the last two services, here, metadata are added to the video content in order to facilitate the subsequent location of the video by other users. This record may contain even more information than a simple tag, because the user chooses from a set of predefined categories the one which is more related to the content of the video. That is, when Youtube is profiling a user based on the videos he has watched, the process of information is easier because the work has already been done, by the owner of the video.

4.1.2 Privacy risks

From the previous summary about social tagging systems, it is clear that tagged content could involve more personal information than other "keywords" such as search queries. Although tags are commonly created by the user, and he could use non intelligible words, there are several research proposals to improve the existing tag recommendation mechanisms. In this line, Delicious includes a system that dynamically recommends tags to the user when a bookmark is added.

Additionally, the information that tags provide can be interpreted in two dimensions. From the perspective of who creates a tag (incorporating it to its profile) and from the perspective of who uses the tag (e.g. by accessing tagged content like Youtube videos).

Given that user-added tags are capable of exposing refined personal information, the risks for privacy get increased. In particular, sensitive tags may reveal enough information to obtain an accurate profile of the user.

4.1.3 Format of social tagging data

Interactions between users and Internet services are mostly performed by means of a web browser. When clicking links or sending web forms through the browser, users are essentially sending HTTP requests to web services. Similarly, users send their tags to tagging services after filling a web form where they indicate the tags to be added and, sometimes, other descriptive information. Web forms are submitted through an HTTP POST request whose content transports the information provided in the form, included the tags.

This HTTP POST task basically sends name/value pairs to the web service, identifying with names the filled parameters in an HTML web form.

Depending on the service, tags are grouped in a pair identified by a pre-defined name, and whose value is a string that groups the user-added tags. This is illustrated in Fig. 5.

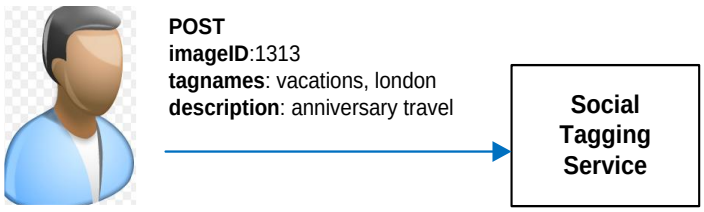


Fig. 5: Process of sending tags through HTTP POST messages.

4.2 Measuring user privacy in the browser

In [8] a proposal of a Firefox extension is done to measure the user's privacy in a web searching context. Our work is to firstly adapt this extension to other services, such as social tagging platforms, also collecting a vast amount of users data. The keywords involved in the user searches in [8] are processed internally in the web browser used to send them, and a user profile is created from this process. This user profile is then used as input of the measuring modules that quantify privacy according to the criterion studied in Chapter 3. Fig. 6 illustrates the operation of this tool.

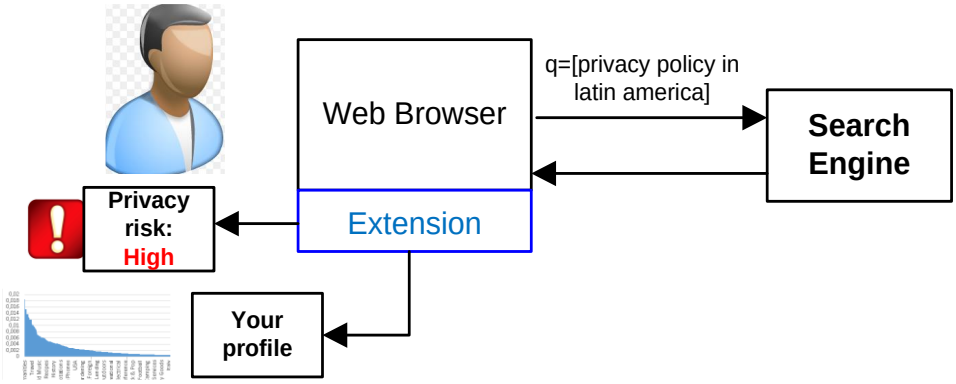


Fig. 6: Architecture of application proposed in [8] to measure privacy in the browser.

The tool proposed in [8] is implemented as an extension of Firefox browser, since browser extensions provide the interfaces and flexibility to crawl the browsing activity from the user side.

However, tagging data can transport, as explained above, a large amount of information that may be employed to obtain more accurate profiles of users.

That is how the first contribution of this work is done, by adding some functions to this extension, so that it can capture this tagging information.

4.3 Integration with Delicious

As already mentioned, Delicious is a social bookmarking service, one of the first that appeared, and with a large base of users. Even though it has a mass of users whose main interests are related to technology, the folksonomy that Delicious has generated has been subject of very deep analysis. In Delicious, bookmarks are closely linked to user-added tags and semantic concepts are involved. That is the reason why there is much interest in using this information to train recommendation technologies that have to do a semantic analysis of user keywords.

The integration of user tagging information from Delicious into the measurement of privacy consists on including user tags in the profiling process of the Firefox extension proposed in [8]. The objective is to enrich the user profile which is the basis to calculate the privacy level. To be included, this information must be captured by any available interface in the browser development framework.

In order to capture tagging information employed or created by a user in Delicious, we first have to analyze where it is generated or stored. For Delicious, specifically; there are three ways to capture user tags:

- Storing user tags in the user side to then access them locally.
- Capturing the outgoing traffic, the instant in which the tags are sent from the browser to Delicious.
- By extracting tags from the Delicious user profile.

4.3.1 Delicious extension as a local source of user information

Delicious BookMarks [28] is a browser extension for Delicious. This Firefox extension (also works in Chrome) allows the integration of Delicious bookmarks with the browser. It allows the user to have his Delicious tags stored and synchronized in his local machine.

As can be seen in Fig. 7 tags are tabulated in a section in the browser, according to the number of times that each of them has been used. The extension also provides some options to add tags and organize them in Firefox.

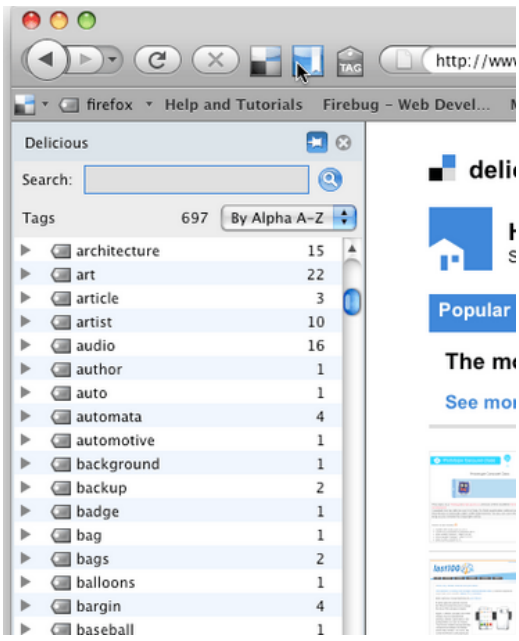


Fig. 7: Control panel of Delicious Bookmark Firefox extension. It shows the list of user tags and the number of times each tag has been used by the user.

The Delicious Bookmark extension stores the user tags (and some other information) in the hard drive, specifically in the folder of the Firefox profile where the extension was installed. A SQLite file, called *ybookmarks.sqlite*, contains the database corresponding to bookmarks extracted from the Delicious web site.

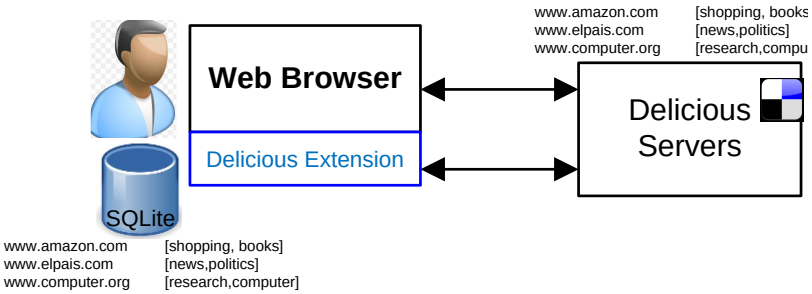


Fig. 8: Structure of Delicious Bookmark extension.

4.3.2 Capturing user-added tags

The *Delicious Bookmarks* extension also allows the user to add new bookmarks (and tags) from its control panel in the same way as it is possible from the Delicious website. In a similar way as *PrivMeter* in [8] captures search queries, we can also retrieve the information sent every time the user adds a new bookmark. This information is basically:

- the title of the website being bookmarked
- the user's tags

We reused some functions found in *Delicious Bookmarks* extension in order to retrieve these data, whenever it is generated by the user. The title added and the tags are stored in respective arrays to then be profiled, in the same way as user's queries in *PrivMeter*, so that their influence can be considered when the user's profile privacy is measured.

4.3.3 Tag import to the user profile

An additional mechanism is implemented here for the user to be able to import his tag history (already available tags) to his profile. In the same manner as the search history contributes to have a starting point for the user profile in *PrivMeter,* available tags are susceptible to be imported from the database of *Delicious Bookmarks* extension.

In order to accomplish this objective we leveraged the Javascript code of *Delicious Bookmarks* to have access to all the tags and their frequencies of usage. The function **yygetAllTags** was created to retrieve the user tags

from the database *ybookmarks.sqlite,* where all the tagging information is stored.

Once all the user tags are retrieved (using the code in Fig. 9), they are profiled, one by one, by means of the profiling module of Adnostic extension[29], also used in [8] to obtain the user profile whose privacy is measured.

```
yygetAllTags: function(){
    tagArray=ybBookmarksMenu._sqliteStore.getAllTags( null,"frequency" )
    var resArray = [];
    var tagLen = tagArray.length;
    for (var i = 0; i < tagLen; i++) {
     var tmp = tagArray.queryElementAt(i,Components.interfaces.nsIWritablePropertyBag);
    resArray[i] = new yyTag(tmp.getProperty("name"), tmp.getProperty("frequency"));
    }
    return resArray
}
```

Fig. 9: Function to retrieve all the user tags stored in the user machine by the Delicious extension.

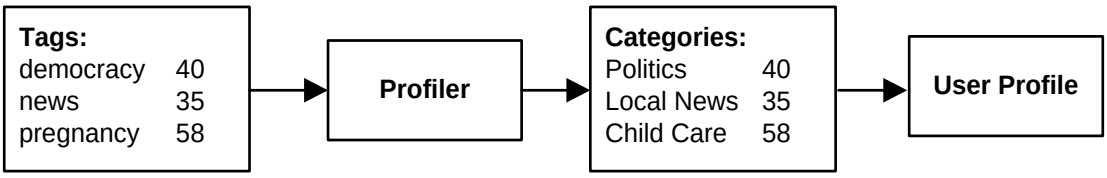


Fig. 9: Profiling process of Delicious tags.

The frequency of usage of each tag is also retrieved to weight the profiling process so that if a tag is used 100 times will contribute in a greater extent than a tag used only 10 times.

4.3.4 Dataset training to get percentile values

Once the browser (by means of our extension) can capture user's tags and is able to build a user profile based on these tags, privacy can be measured by using the metrics justified in Chapter 3.

The entropy of the user profile is the first element employed to measure its privacy risk, as justified in Chapter 3. But, to interpret the values of privacy (e.g. in a risk scale), percentiles are used to compare the current value of user's privacy with the corresponding value of privacy belonging to users in a population. This comparison will assign to the user a privacy level according to the behavior of other users, which will be more realistic.

Firstly, we need to obtain tagging information (tags) of a population. Then, these tags will be categorized by a training module which shall obtain the profiles of all users. Finally, entropies of all profiles are calculated and stored. This process is illustrated in Fig.13.

We used a Delicious dataset from [32]. This dataset contains tags from about 10.000 users. Each user has up to 5000 tags but the 90 percent of them has up to 500 tags. Some filtering was done over these tags, eliminating some special characters not understandable by the profiler.

Once having the dataset of Delicious tags we started the training function to get the user profiles, according to their tags. The main function used to profile this dataset is the following:

```
_startTraining : function(entorno_usuario) {

            // Get list of files containing user tags
            var filelist = [];
            _getUserFilesList(filelist);

            alert("Privacy TRAINING starts for range queries with "+filelist.length + "
users");

            for(var i=0; i<filelist.length; i++){
                userid = filelist[i];
                p_docLog = _getDocLogTraining(entorno_usuario, userid + '.xml');

                var usertags = [];
                _getUserTags(entorno_usuario, userid, usertags);

                for(var j=0; j<usertags.length; j++){
                    var tag = usertag[j];
                    _profileTag(tag)
                }
                _saveProfileTraining(userid + '.xml');
                p_docLog = null;
            }
}
```

Fig. 10: Main function used to profile the Delicious dataset.

Having user profiles, entropies were calculated, by means of the following code:

```
_getEntropies : function(entorno_usuario){
            var filelist = [];
            _getUserFilesList(filelist);
            entropias = [];
            for(var i=0; i<filelist.length; i++){
                var userfile = filelist[i]+".xml";
                var entropia = _getTotalEntropyByFile(entorno_usuario, userfile);
                entropias[i] = entropia;
            }
            return entropias;

}
```

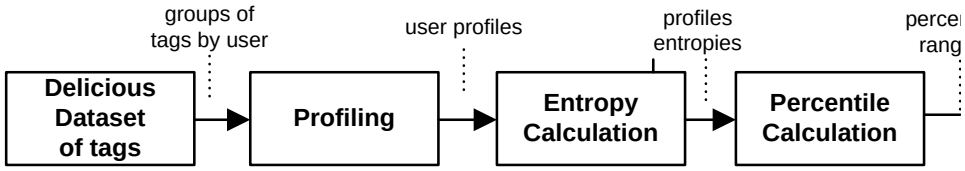
Fig.12: Javascript function used to calculate user entropies from users' profiles of Delicious datasets

The values of entropies of the population in this Delicious dataset are showed in Fig. 14. As we can see, most of the users have an entropy between 4 and 6, being 9.3 the maximum value.

Then, by using Matlab functions we processed the information about entropies and got the corresponding percentiles. In order for the percentiles to be used during the measuring process, the extension proposed in [8] has to locally store them. Each time the privacy level is calculated, the extension has to verify the percentile range to which such value belongs.

According to this percentile, the privacy risk level can be showed to the user in a more realistic fashion.

The whole process of training of this dataset is clearly illustrated in the Fig. 13.



ig

Fig. 11: Training process of Delicious dataset to get percentile ranges of its entropy values.

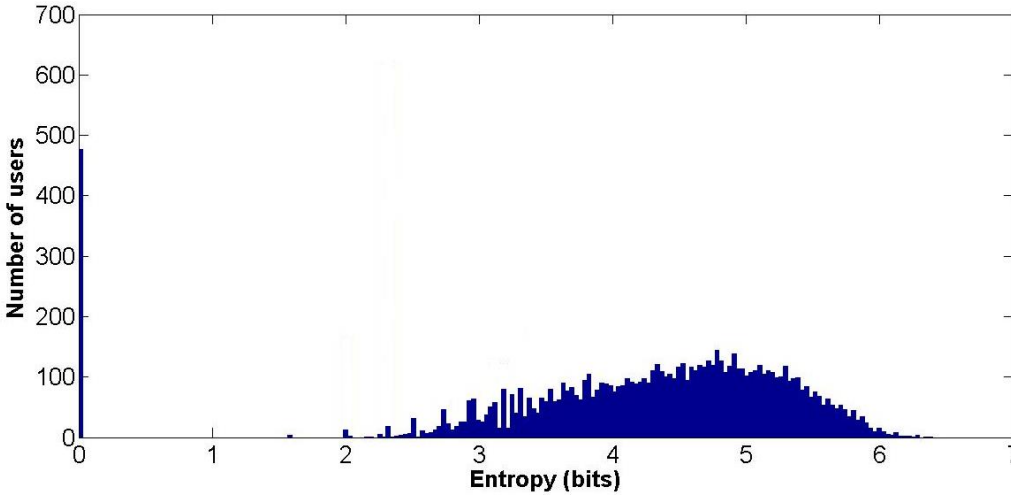


Fig. 12: Histogram of entropies of users in Delicious dataset [32].

4.4 Integration with Flickr

Flickr is another social tagging service, where users attach tags to their uploaded photos. The process by which tags are added is basically the same as with Delicious or any other website receiving data from a user; i.e. by means of web forms that, when submitted, generate HTTP POST requests containing the user tags. As explained in Section 4.1.3, the format used is a name/value as illustrated in Fig. 15.

tags=politics%20spain%20new

Fig. 13: Format of Flickr tags when sent from the browser to the web server.

To include (and capture) Flickr tags, as another input in the profiling process executed in the Firefox extension, the following code in Fig. 16 is added. Essentially, a listener is

created that detects HTTP requests to the Flickr website. When the requests are recognized to have user tags, those are captured, and the profiling module is called to categorize these tags.

If the user already had tags in his Flickr profile, they can be imported in the user profile in the Firefox extension, similarly as we did with Delicious. The Flickr site has a section where the whole list of user's tags are displayed.

In order to give the user the option to import his Flickr tags, the code in Fig. 17 was created. This code parses the HTML page where the user's tags are displayed to get all the tagging information. Then, this information is profiled according to the number of times each tag was used.

The import process, of course, must be launched by the user (pressing a button), after having installed PrivMeter. Once the tags have been imported, the extension will automatically categorize the following tags.

4.5 Integration with Youtube

Currently, Youtube incorporates tagging information for videos. To upload a video, it is compulsory to indicate the category to which the video is related. Since not many users upload videos in comparison with users who watch them, this category information may be of more interest to the privacy of the latters.

In order for this information to be considered during the profiling process, another listener was programmed in the extension to monitor the user while he watches Youtube videos. Youtube displays the category to which a video belongs, along with its description, when the video is reproduced.

By retrieving this category, from the *PrivMeter*, we can profile it to include this information as part of the user's profile whose privacy is measured. The code to capture these data is detailed in Appendix A.2.

```
    function logPostMessage (msg){
              if (!msg || msg.length<1) return;
              var consoleService = Components.classes["@mozilla.org/consoleservice;1"]
                .getService(Components.interfaces.nsIConsoleService);
              consoleService.logStringMessage(" POST DATA: " + msg);
              //return;
            }
    //create an nsIObserver implementor
    var listener = {
       observe : function(aSubject, aTopic, aData) {
          var httpChannel = aSubject.QueryInterface(Components.interfaces.nsIHttpChannel);
          if (aTopic == "http-on-modify-request") {
             try{
                if(httpChannel.requestMethod == "POST"){
                     var uploadChannel = httpChannel.QueryInterface(Components.interfaces.nsIUploadChannel);
                     var uploadChannelStream = uploadChannel.uploadStream;
                     uploadChannelStream.QueryInterface(Components.interfaces.nsISeekableStream).
                           seek(Components.interfaces.nsISeekableStream.NS_SEEK_SET, 0);
                     var stream = Components.classes["@mozilla.org/binaryinputstream;1"].
                         createInstance(Components.interfaces.nsIBinaryInputStream);
                     stream.setInputStream(uploadChannelStream);
                     var postBytes = stream.readByteArray(stream.available());
                     var poststr = String.fromCharCode.apply(null, postBytes);


                     orden=poststr.split('&').filter( function (x) { if(x.split('=')[0].toLowerCase()=='method') return (x.split('=')[1].toLowerCase)
;} )

          // If POST goes to www.flickr.com and the command is addTags
                     if(httpChannel.getRequestHeader("Referer")){
                         if(httpChannel.getRequestHeader("Referer").split('/')[2] == "www.flickr.com" && orden[0] == 'method=flickr.photos.addTags'){
                             //Split based on & and then filter tags array
                             var objetoParTag=poststr.split('&').filter( function (x) {
                                       if(x.split('=')[0].toLowerCase()=='tags') return (x.split('=')[1].toLowerCase) ;} )
                             var etiqueta = objetoParTag[0].split('=')[1]; // Retrieve tag from last object
                             etiquetas = [];
                             etiquetas = etiqueta.split('%20'); // If there are various tags
                             for(var j=0; j<etiquetas.length; j++) {
                                 // Categorize each tag
                                 ADNOSTIC.profiler._profileDelic(etiquetas[j]);
                                 logPostMessage(etiquetas[j]);
                             }
                         }
                     }
                     if(poststr.split('\n').length == 4){
                          poststr = poststr.split('\n')[3];
                     }
                     logPostMessage("POST :  query:| "+poststr);
                     //from mdn: Creating_Sandboxed_HTTP_Connections
                     var inputStream = Components.classes["@mozilla.org/io/string-input-stream;1"].
                         createInstance(Components.interfaces.nsIStringInputStream);
                     inputStream.data = poststr;
                     uploadChannel.setUploadStream(inputStream, "application/x-www-form-urlencoded", -1);
                     // order important - setUploadStream resets to PUT
                     httpChannel.requestMethod = "POST";
                }
             }catch(err){}
          }
          else if (aTopic == "http-on-examine-response") {

          }
       },

       QueryInterface : function(aIID) {
          if (aIID.equals(Components.interfaces.nsISupports) ||
                aIID.equals(Components.interfaces.nsIObserver))
             return this;
          throw Components.results.NS_NOINTERFACE;
       }
    };

    var observerService = null;

    var Init = {
       addObserver : function(){
          observerService = Components.classes["@mozilla.org/observer-service;1"].getService(Components.interfaces.nsIObserverService);
          observerService.addObserver(listener, "http-on-modify-request", false);
          observerService.addObserver(listener, "http-on-examine-response", false);
       },

       removeObserver : function(){
          observerService.removeObserver(listener, "http-on-modify-request");
          observerService.removeObserver(listener, "http-on-examine-response");
       }
    };
    Init.addObserver();
```

Fig. 14: Code added to the extension developed in [8] to detect when the user sends tags to Flickr, to capture and categorize these tags.

```
        _getFlickrTags : function() {
            var win =  _getRunningWindow();
            if (!win ) return;
            fTagObArray = [];
            var url = win.gBrowser.contentDocument.baseURI;
            url_array = [];
            url_array = url.split('/');
            if(url_array[2] == "www.flickr.com" && url_array[3] == "photos" && url_array[5] == "alltags"
){
                var tagElts = win.gBrowser.contentDocument.getElementsByClassName("linkout");
                var tagCounts = win.gBrowser.contentDocument.getElementsByClassName("PhotoCount");

                if (tagElts == null && tagElts.length == 0) return null;
                for(var i=0; i<tagElts.length; i++){
                    ftag = tagElts[i].childNodes[3].childNodes[0].nodeValue
                                    .replace(/[^a-zA-Z0-9ñ]/g,'');
                    fcount = tagCounts[0].childNodes[1].childNodes[0].nodeValue;

                    var tagObj = {
                        name: ftag,
                        count: fcount,
                    };
                    fTagObArray.push(tagObj);
                }
                return fTagObArray;
            } else {
                return null;
            }
        }
        _yysetProfileFromFlickr : function() {
            var win =   getRunningWindow();
                   if (!win ) return;
            var etiquetas = [];
            etiquetas = win.ADNOSTIC.editor._getFlickrTags();
            if(etiquetas == null) {
                alert("There was an error when importing your Flickr Tags")
                return;
            }

            for(var i=0; i<etiquetas.length; i++){
                etiqueta = etiquetas[i].name;
                conteo = etiquetas[i].count;
                alert(etiquetas.length + " etiquetas");
                win.ADNOSTIC.profiler._profileDelicCount(etiqueta, conteo);
            }
            alert("Flickr tags have been imported sucessfully");
            return;
        }
```

Fig. 15: Functions created to retrieve and profile the user's tags from his Flickr profile in the Web.

5. Integration & evaluation of a query obfuscation mechanism

This chapter describes how we used the privacy metrics exposed in Chapter 3 to evaluate the efficiency of a privacy protection tool, TrackMeNot.

It is a fact that, in the privacy research literature, several proposals have been made to protect privacy. At the user level, however, not many available tools implement those privacy protection approaches. Instead, we can find some mechanisms, based on heuristics, trying to isolate the user from the privacy risks by means of blocking certain interactions with the Web.

A first step towards the developing of more efficient privacy protection applications is, thus, evaluating the existing ones. Determining their real benefits would allow us to take advantage of well-designed mechanisms and to detect wrong strategies.

Evaluating privacy protection applications means comparing privacy levels offered before and after the mechanism was implemented. A sort of privacy gain can be calculated, interpreted as the potential benefits of applying a given tool. The problem here is that measuring privacy is not a simple task; multiple variables could be taken into account and, in the same manner, multiple approaches of protection are proposed both theoretically and in the practice. Evidently, though, very few of the existing tools could be evaluated based on the same parameters.

Along this chapter we explain the methodology used to evaluate one of these privacy protection tools, by using the privacy metrics proposed in [9] and some modules implemented in *PrivMeter* [8].

5.1 Queries as user identifying information

Information search has become a very common activity for users when browsing on Internet. By themselves, search queries can profoundly reflect our interests, our worries or problems. In combination with social network

interactions and tagging activities, search querying could precisely reveal our identity. It was publicly demonstrated when released "anonymized" search logs in 2006 were used by the New York Times to expose the identity of woman [7]. Not in vain Google, the biggest search engine, obtains most of its profits from advertising. Then it is evident that they are doing a pretty good job in profiling users to accurately personalize ads.

As with the rest of providers of personalized information services, Google argues that maintaining such amount of user data helps them to improve services and prevent fraud [30]. But, as we argued in the Introduction, the risk lays especially in the cooperation that companies like this one are offering to governments and other external entities.

As already stated in Section 2.5, data perturbation is a privacy preserving technique that can be applied in the user side, independently from third parties. Perturbation of queries or query obfuscation is a mechanism that has been implemented in the TrackMeNot Firefox extension in order to protect individuals against profiling activities when they browse on the Web.

5.2 TrackMeNot

TrackMeNot (TMN) is a Firefox browser extension whose aim is to obfuscate the user's profile, through introducing bogus material in the search query stream. This mechanism is based on the artificial generation of query-like phrases that are then sent to search engines via HTTP requests. In order to prevent the search engines to detect an automatically generated query from the real ones, some mechanisms are also implemented in TMN to simulate a human search behavior.

The archicture of this application is illustrated in Fig. 18 and some of its involved modules are briefly described below.

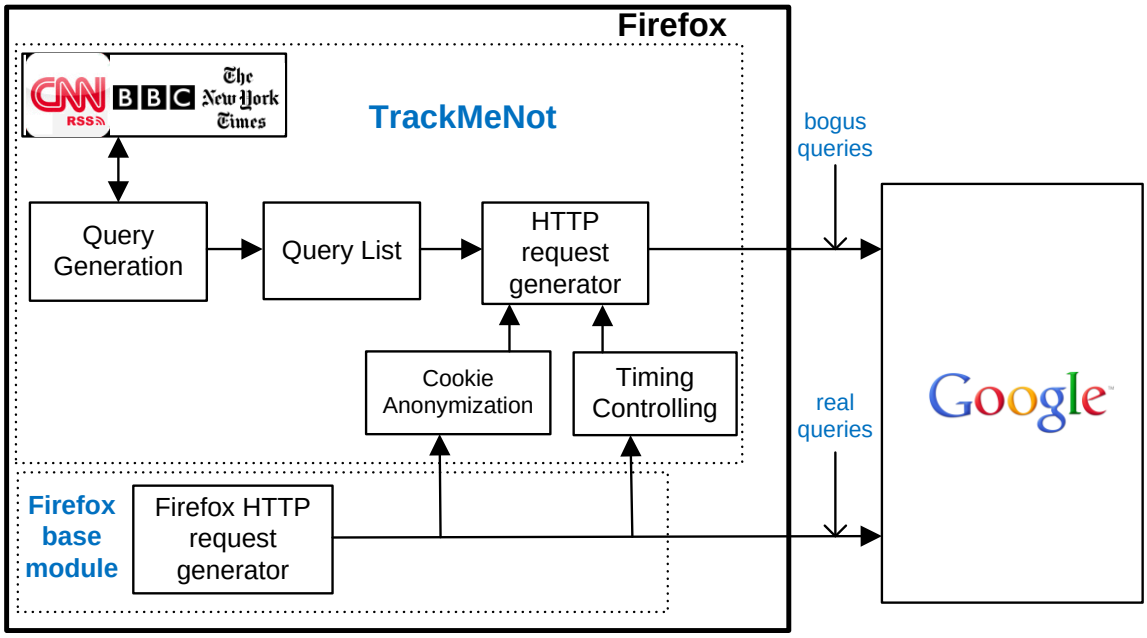


Fig. 16: TrackMeNot Architecture.

- **Dynamic Query-lists.** TMN needs to dynamically generate search queries to make it more difficult for search engines to accurately identify users by profiling these keywords. TMN essentially uses publicly available sources of information (RSS feeds) to build a seed list of query terms from which the search queries will be obtained and then sent to the search engine. This list evolves in the time according to the RSS feeds configured in the TMN options, especially because a new list is retrieved each time de Firefox browser is restarted. Also, given that the default RSS feeds belong to on-line newspapers' sites, the available words will change in a daily basis.
- **Real-time search awareness.** Some search engines like Google are capable of detecting automated requests, fundamentally by identifying too many requests in a short period of time. In order to prevent this, TMN provides a module that monitors the user search behavior so that it sends a number of bogus queries whenever a real user query is detected.
- **Live Header Maps.** To more closely mimic the user's search behavior, TMN adapts its HTTP requests headers according to the headers used by the user to send the most recently query. Recall that, depending on the web browser used, information in headers can also be used to identify a user in a region [33].

- **Burst-Mode Queries.** This mode of working is implemented in TMN to send a batch of fake queries in the moment that a user sends his. This mechanisms contributes to mimic the user behavior that usually submits several queries in a short period of time.
- **Cookie Anonymization.** This TMN module blocks cookies for user's search queries but, instead attaches them to the (artificially) generated search queries, so that the search engine registers only the fake queries in the user profile.

TrackMeNot is a widely known tool that uses an innovative mechanism to simulate the user search behavior in order to obfuscate the user profile seen by search engines. Not many other tools are available for the user that involve obfuscation of queries or tags, even when there exists theoretical contributions that proposes such strategies.

5.3 Evaluation methodology

The evaluation of TMN is basically done by means of a comparison of the levels of user's privacy, before and after that TMN is used, so that a gain value can be calculated. If the privacy gain is positive, it would mean that the implemented mechanism is effective. If there is a gain, we shall interpret if this gain is enough to protect the user against the external adversaries (i.e. if it is efficient).

The main steps, performed to estimate the efficiency of TMN with the objective of increasing privacy protection, are the followings.

- Obtain a significant sample of real user query logs, whose privacy will be measured.
- Generate sets of fake queries by means of the mechanism offered by TMN.
- Obfuscate the users' real queries by mixing real queries with fake ones.
- Obtain users' real and obfuscated profiles, from the corresponding query logs.
- Measure privacy of real and apparent profiles, using metrics already described.

- Determine the privacy gain of user profiles after obfuscation.

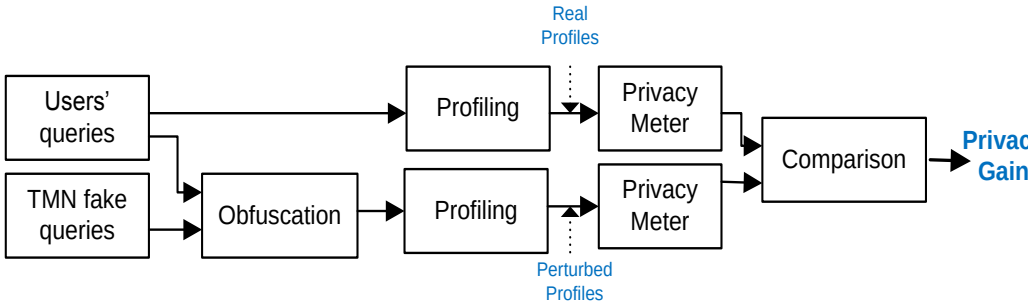


Fig. 17: TMN evaluation process.

As indicated in Fig. 19, in order to start the evaluation process we first obtained a sample of query logs belonging to a significant number of users (the users' queries). We used, with this goal, the AOL dataset [7] released in 2006, which contains around 20 million queries of 650 thousand users. The sample we took belongs to 6674 users, each of them having from 501 to 1000 queries. From these 6674 users we chose the 50 users with the greatest amount of queries.

Then, fake queries were obtained by means of the TMN query generation process. The code from TMN extension were hooked to create a function capable of generating 700 search queries from the RSS feeds configured by default (feeds of CNN, BBC, The Register and The New York Times).

As showed in Fig. 19, the obfuscation of real user queries was done by mixing these queries with the already generated fake queries (we got the obfuscated queries for each user).

Both real and obfuscated queries were separately profiled to get the corresponding real and obfuscated profiles. At this point we had the real profiles whose owners would be interested in hiding and the obfuscated (apparent) profiles which would be the resulting profiles of users after implementing TMN.

Evidently, the last step was to measure the privacy (according to metrics explained in Chapter 3) of each type of profiles to finally compare the values obtained for each user. This comparison determined a value of gain of privacy that allowed us to know if the obfuscation was beneficial for the users' privacy.

Some of these steps will be explained with more detail along the next headings.

5.4 Profiling user information

We used two existing mechanisms in order to get a user profile (as described in Section 3.1) susceptible to be measured by the metrics described in Chapter 3.

- **Adnostic's profiling module.** This module was used in [8], as part of *PrivMeter*. It builds a user profile from the continuous categorization of the user's search queries. We have modified its functions so that it can receive all the users' queries from our sample to obtain the user profiles.

  The categories in which the text can be classified do not include sensitive categories related to health, racism or porn. The scheme of categorization is based on Google hierarchy that contains 602 categories.

- **TextWise** [31]**.** It is a semantic technology implemented on the Web to be freely accessed through a web API (although with some restrictions in the usage). Among other services, TextWise offers a categorization service that accepts requests containing keywords and returns up to 5 categories in which the sent text can be classified.

  Unlike Adnostic's profiling module, TextWise uses a hierarchical schema based on ODP (Open Directory Project) that is built of 770 categories.

  We built a program to use this API in order to categorize each query from the AOL sample. This information allowed us to create user profiles according to this strategy of categorization.

  The code of the program, written in Python, to consume the TextWise API and build the profile of a list of users is showed in the Appendix. It receives a set of files containing search queries (each one represents a user and contains his queries), and returns the corresponding profiles.

As stated in Chapter 3, the measurement of privacy may depend on the capabilities of the adversary. If we measure the privacy of users whose profiles are obtained by using two strategies of profiling, we are modeling two different adversary capabilities. Hence, it is interesting to see if the

impact of the privacy enhancing mechanism is the same no matter what profiling strategy is considered.

5.5 Privacy Measuring

During the TMN evaluation, as showed in Fig. 19, once both real and apparent profiles are available, the next step is measuring their privacy. Privacy metrics used in this project have been already introduced in Chapter 3. Essentially, the entropy of a user profile is the first way to measure the user's privacy. Also, the divergence of the user's profile with respect to the profile of a predefined population group can be used to measure his privacy.

From the users' profiles gotten after the profiling, both of the last metrics were obtained (illustrated in Fig. 20):

- **Entropy** of the user profiles
- **Divergence** of the user profiles relative to the average population profile obtained from Google Ad Planer.

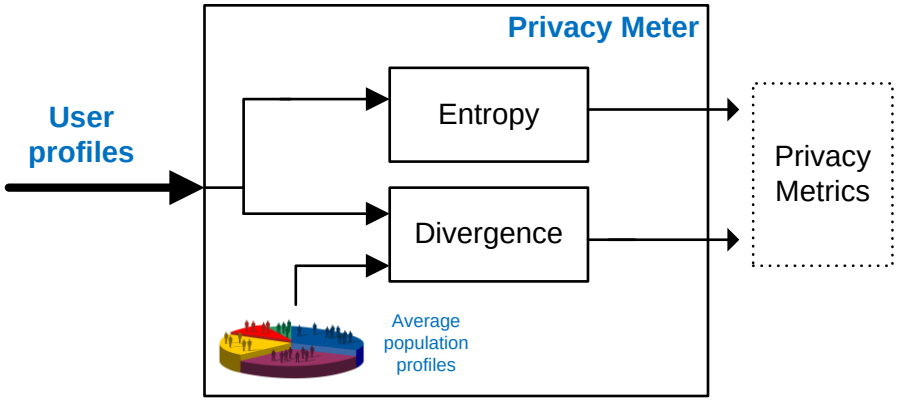


Fig. 18: Structure of privacy meter module, which receives the user's profiles and returns privacy measurements based on entropy and KL divergence.

5.6 Privacy gain

According to the analysis in Section 3.3.1, an increasing value of entropy means a higher level of user's privacy. When getting the divergence of the user's profile with respect to the average population profile, a privacy gain is obtained when this divergence is decremented.

The privacy gain level is, then, calculated by comparing the privacy before and after the TMN perturbation mechanism was implemented, using the following expressions.

Being *q* the user profile, *t* the obfuscated user's profile, *p* the average population profile and *M* the privacy gain level, we have:

for the privacy measured as the entropy of the user's profile (*H(q)*), the privacy gain relative to the initial privacy level is

$$M_H = \frac{H(t)-H(q)}{H(q)},$$

and for privacy measured as the divergence of the user's profile with respect to the average population's profile (*D(q|p)*)

$$M_D = \frac{D(q|p)-D(t|p)}{D(q|p)}.$$

These values would give us an initial view of how privacy is being enhanced when using obfuscation of queries. Calculating the percentiles in the population profile to which these values belong will give us a more realist measure of the level of privacy.

5.7 Evaluation environment

Recall that the objective of this evaluation was to measure the privacy gained after using the TMN obfuscation mechanism, but also, trying to determine which of its configuration parameters have an important influence on the effect of enhancing privacy.

The main inputs of this environment were the queries obtained as a sample from the AOL dataset. The fake queries used to obfuscate the users' profiles were artificially generated from the TMN mechanism. Basically, we programmed TMN to generate different amounts of fake queries, during 5 days, in order to then verify how the proportion of fake queries with respect to the real ones influenced the user´s privacy gain. Hence, being ρ the relation of the number of fake queries with respect to the number of real ones, we determined how privacy gets increased with ρ.

$$\rho = \frac{Number\ of\ fake\ queries}{Number\ of\ real\ queries}$$

Some evaluations were also done by modifying the RSS feeds that TMN uses as source to generate the fake queries.

The evaluations were done considering both the entropy and divergence as metrics of the users' profiles privacy, as explained in Section 5.5.

5.8 Results of privacy measuring

5.8.1 Analysis of fake queries

The 700 fake queries obtained (during 5 days) from TMN are analyzed first in this section. As if those generated queries were user queries, we profiled them to analyze how these profiles could impact on the user profiles. This was done by means of the Adnostic's profiling module.

Each profile, obtained from the fake queries, was built of about 140 categories (from the 602 available in the categorizer).

Figs. 21 to 25 show the histograms of the fake queries obtained along 5 days (only the 30 most popular categories are showed).

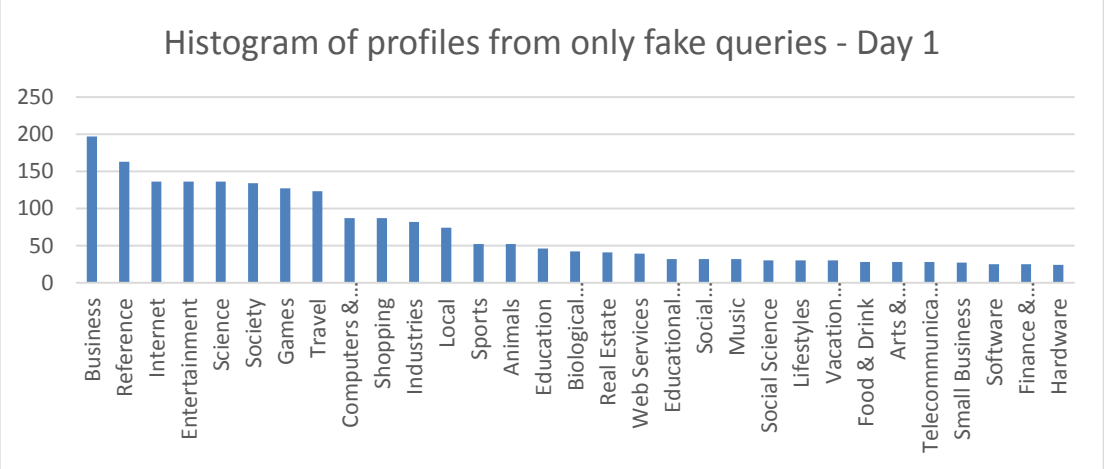


Fig. 19: Histogram of categories obtained after the profiling of TMN fake queries, generated using the default 5 RSS feeds on day 1 (only the 30 most popular categories are showed).

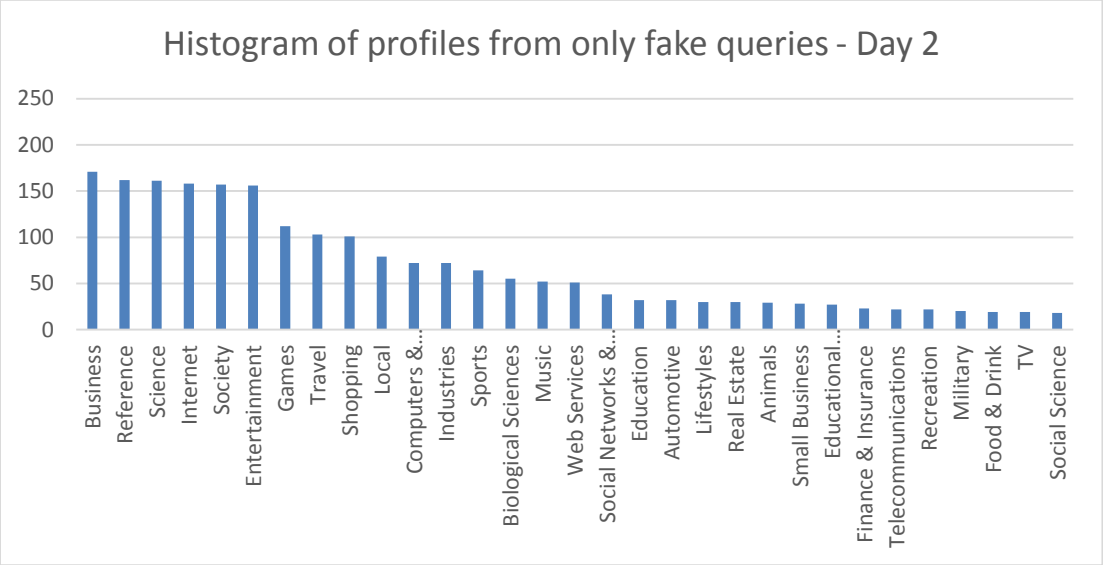


Fig. 22: Histogram of categories obtained after the profiling of TMN fake queries, generated using the default 5 RSS feeds on day 2 (only the 30 most popular categories are showed).

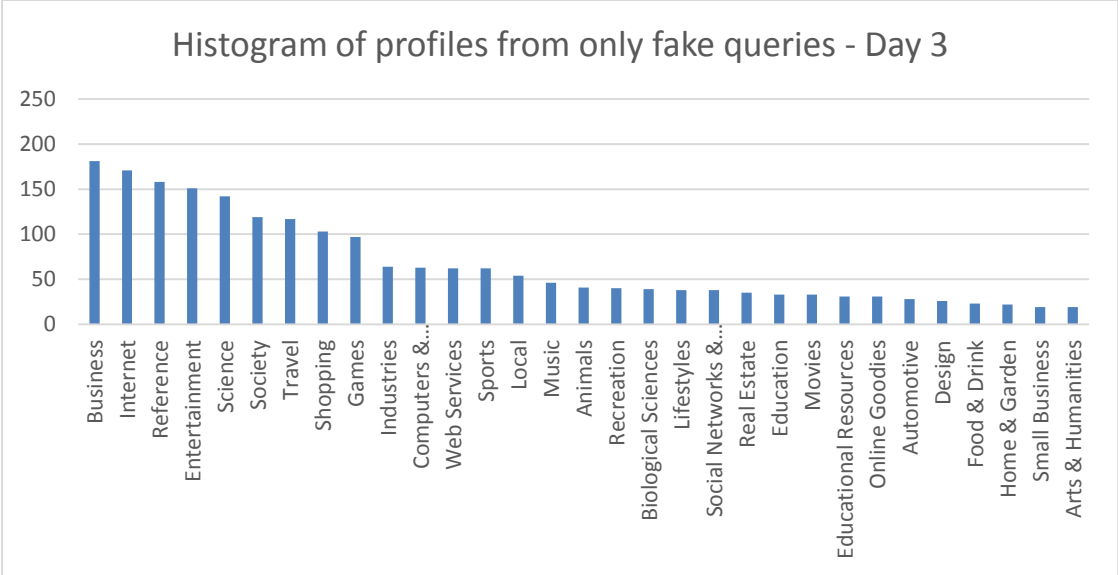


Fig. 23: Histogram of categories obtained after the profiling of TMN fake queries, generated using the default 5 RSS feeds on day 3 (only the 30 most popular categories are showed).

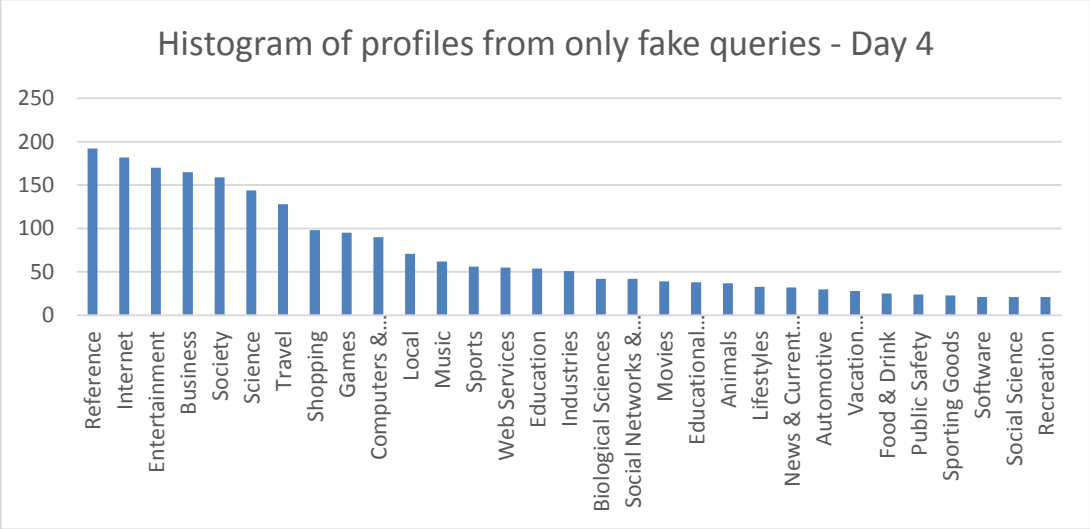


Fig. 21: Histogram of categories obtained after the profiling of TMN fake queries, generated using the default 5 RSS feeds on day 4 (only the 30 most popular categories are showed).

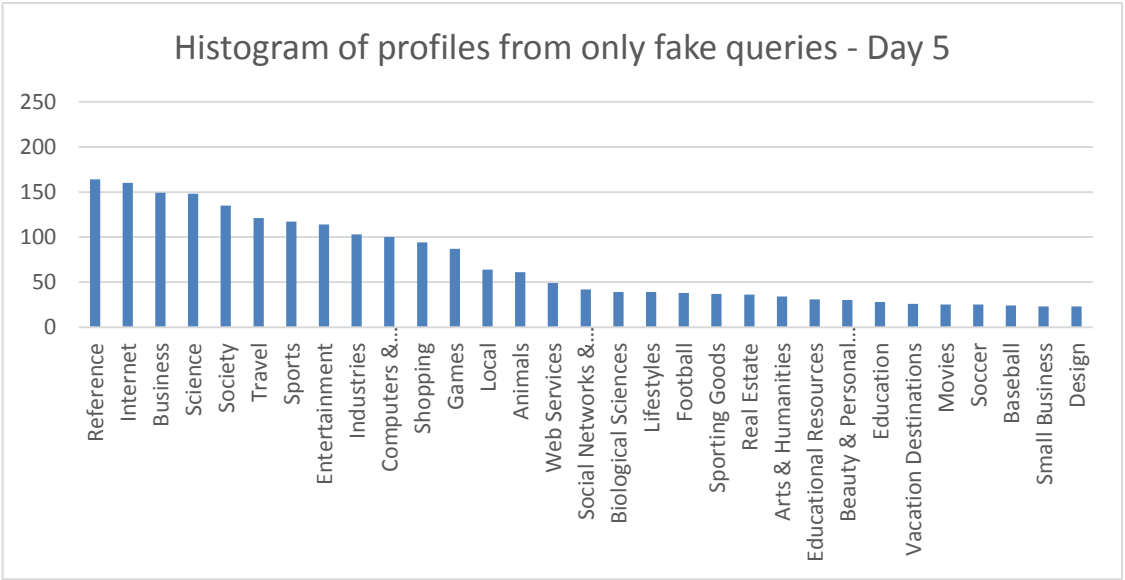


Fig. 20: Histogram of categories obtained after the profiling of TMN fake queries, generated using the default 5 RSS feeds on day 5 (only the 30 most popular categories are showed).

We observed that, during the experiment, the categories to which the TMN fake queries belong are basically the same, and their impact (popularity) is similar during the 5 days. In addition, the category "References" repeatedly appears, having an important influence in these histograms. From our experience with the Adnostic's profiling module we

have seen that "References" is a sort of generic category used (but not strictly) to classify all the queries related to sensitive information (e.g. health condition). This minimizes the influence of such categories in the profile; so it does not provide much information.

The profiles are very similar, which means that, at least during these five days, the topics of fake queries do not change much. This would make easier the work for an adversary when trying to separate the influence of such queries in the obfuscated profile, in order to obtain the real user's profile.

5.8.2 Privacy gain using default TMN´s RSS feeds and Adnostic's profiling module

There is actually a gain in users' privacy when obfuscation of their queries was implemented using TMN, both in terms of users' profiles entropy and in terms of users' profiles divergence with respect to the average population's profile.

In terms of entropy, the mean gain was 14.58% and 20.17% in terms of divergence, taken from the 5 consecutive days. This results from using 100% of fake queries (the same number of real queries as the number of fake ones).

The entropy values of users' profiles after obfuscation were also compared with the original values by using non-central position measures (percentiles in this case). It was found that after obfuscation, the users' privacy value (users' profiles entropies) get increased in about 50 percentiles, according to the distribution of entropies obtained from the whole population where the sample queries were taken. This is a considerable gain since a user whose privacy was in the 4^{th} percentile, after obfuscation it went to the 50^{th}.

5.8.3 Privacy gain using default TMN's RSS feeds and Textwise-based profiling module

This test was implemented by using TextWise as the categorization engine. As it involves a wider scheme of categories (about 770), it was supposed to obtain more accurate users' profiles from their queries.

In terms of entropy the average gain obtained from the obfuscation process was of 24.19%, also during a 5-day period. The benefits, are about 5% higher than the ones obtained using a somewhat more limited profiling mechanism

5.8.4 Privacy gain with respect to ρ

Privacy gain was also measured as a function of ρ (described in Section 5.6). We observed how the privacy gain got increased with the percentage of fake queries, both in terms of entropy and divergence. This measures of relative privacy gain are illustrated in Fig. 26 and Fig. 27.

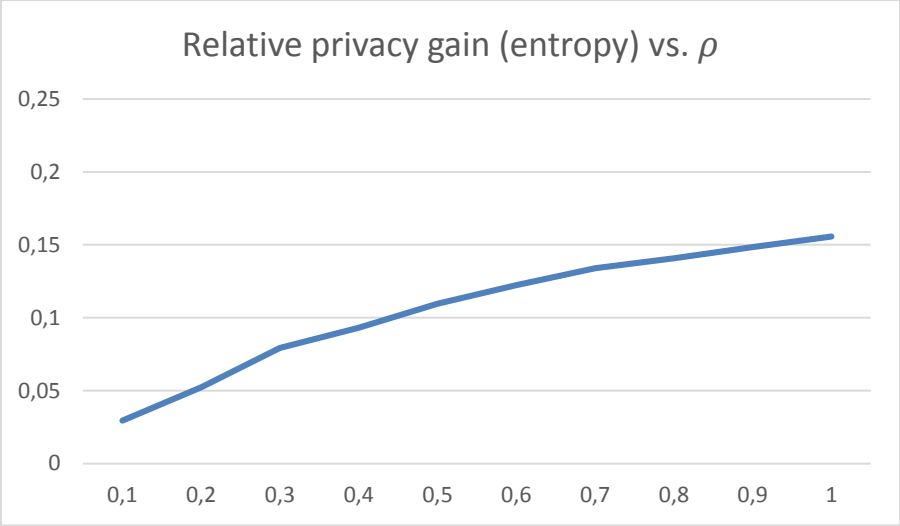


Fig. 23: Increasing of relative privacy gain (in terms of entropy) according to ρ. The greater the amount of fake queries, the higher the privacy gain.

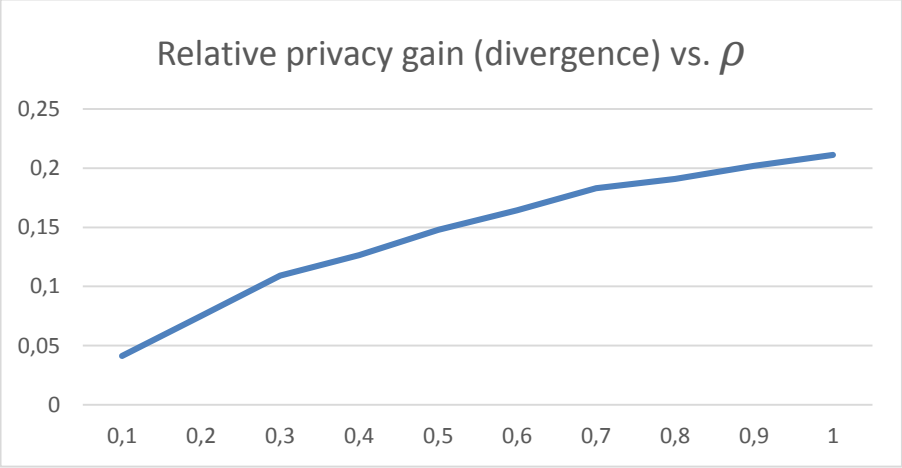


Fig. 22: Increasing of relative privacy gain (in terms of divergence) according to ρ. The greater the amount of fake queries, the higher the privacy gain.

5.8.5 Privacy gain against classification attacks

As stated in Section 3.3.2, the KL divergence could be used as a metric against classification attacks. If we have the profile of a population group, we can classify a user profile as part of such group if the divergence between the two profiles is small enough.

We measured the KL divergence of the users' profiles with respect to the profiles of some population groups (age and gender groups) to classify these users in one of these groups. Almost no user was changed from its original category after the obfuscation process, which means that this mechanism was not successful against classification attacks.

The impact of TMN's fake queries do change the users' profiles, increasing their discrepancy with the category where the users were originally classified. But, this impact is not enough to classify the users in a different category. This suggests that false queries should not be randomly generated because if so, apparently, the influence that they have may be so scattered along multiple categories that the final effect is almost null. Instead, a more directed strategy, say, with the specific objective of provoke the classification of the user in a predefined category, would give better results.

So, the challenge is to implement an intelligent generation of fake queries capable of efficiently obfuscating the users' profiles against classification attacks. This means that the obfuscation process must be adaptable to the particular privacy needs of the users. This process of obtaining keywords (which then would be combined to get fake queries) related to certain topics, without having to depend on third parties, is a very interesting research field.

5.8.6 Privacy gain using a more specific RSS feed

The privacy gain was also measured when using more specific RSS feeds to generate the fake queries. Default RSS feeds belong to well-known newspapers and point to sections were a summary of the more important news is published. Instead, we configured RSS feeds that point to sections of these newspapers related to Sports. The category Sports tends to be a category of greater interest for male public. The intention was to obfuscate

the users' profiles with bogus 'male' queries so that the profiles initially classified as female, could be categorized as male profiles after obfuscation.

This strategy effectively modified the female profiles, reducing its divergence with respect to the male profile. Once again, it was not enough to change the category where the users were originally classified.

5.9 Discussion about parameters involved in the evaluation

As seen in the last section, several elements are involved in the evaluation of this particular privacy-enhancing technology. From the side of entropy and divergence of the users' profiles with respect to the average population's profile, the privacy is efficiently improved but at the cost of additional traffic. The more privacy is needed, the more bogus queries the system must generate. This may also have an important impact on personalized services offered by search engines.

The way the users are modeled is also an important factor when measuring privacy (see Section 3.1), but it depends on the adversary capabilities of user profiling. So, evaluating a privacy-enhancing mechanism by using different profiling methods will give us a better idea of the performance of the mechanism against diverse adversaries. The task of profiling, however, is not trivial, and neither it is to precisely know what technics search engines are using.

Query processing is part of the profiling capabilities we must simulate, both to measure the user's privacy, and so to evaluate a protection mechanism. The idea is again to mimic the adversary intentions of discovering the user's interests, even when the queries are written with typos, to more accurately profile the user. Considering that about 20% of the queries of the obfuscated set were not successfully classified, improving their processing would contribute to enhance the quality of the profiles obtained from them.

The availability of real user information, in the form of queries or tags, may greatly help to interpret the results of the evaluation of this privacy-enhancing technology. If the privacy is measured as a value relative to the values in a population, a more realistic evaluation can be done of the risk levels.

Finally, this evaluation shows that query obfuscation is not efficient enough against more sophisticated attacks (e.g. classification attacks, see Section

3.3.2) if the fake queries are randomly generated. The solution would be the implementation of a strategy of queries generation based on the user's profile and his particular needs.

5.10 TrackMeNot integration with PrivMeter

As showed by the evaluation of TMN, its efficiency on improving privacy level of the user, basically depends on the number of fake queries that are generated with respect to the real ones. This means that a great part of TMN benefits will depend on the periodicity of fake query generation and on the number of queries sent by the user.

The mentioned parameters will change for each user (fake query generation can be configured by the user). Thus, in order for the user to be aware of the relative gain of privacy obtained thanks to TMN, it is helpful to integrate TMN and *PrivMeter* so that privacy enhancements thanks to obfuscation can be measured and showed to the user.

We rewritten some of the functions of *PrivMeter* used to create and update the user's profile. These functions were used to build the obfuscated user's profile, also called apparent profile.

The apparent user profile has the same structure as the real profile (hierarchical scheme of categories and corresponding scores). The difference is that the queries being categorized to build this profile are the user's queries and the queries generated by TMN. This means that our code detects both real and fake search queries, categorizes them and updates the user apparent profile.

Once both real and apparent profiles are available, the relative privacy gain can be obtained similarly to what we done during the TMN evaluation. This gain value can be calculated in terms of the entropy increasing or in terms of divergence (with respect to average population profile) reduction.

Fig. 28 shows how PrivMeter is integrated with TMN in order to take advantage of the privacy measuring available mechanisms in the browser extension.

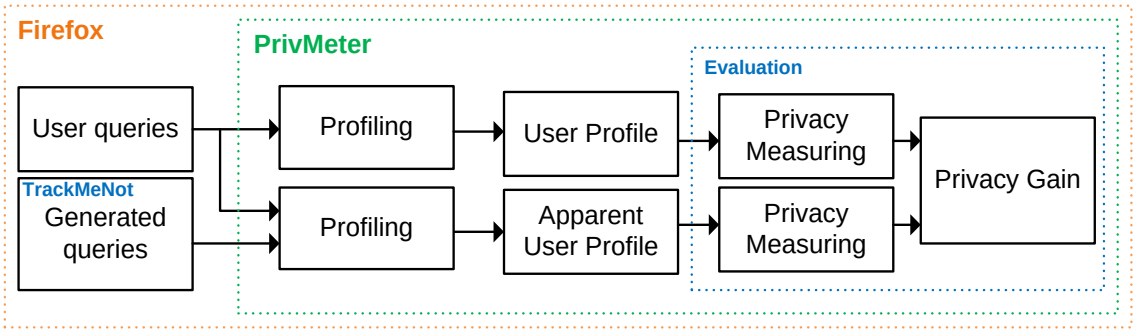


Fig. 28: PrivMeter and TMN modules integrated to include a measure of privacy gain to be showed to the user.

In order for the user to visualize the impact that TMN is having, we incorporated two information items to the privacy bar in *PrivMeter*: the entropy of the apparent user profile and the privacy gain measured in terms of entropy. The Fig. 29 illustrates how this bar looks after including the information about TMN efficiency.

Fig. 29: Privacy risk bar of PrivMeter showing information related to the efficiency of TMN

6. Conclusions

6.1 Conclusions

Thanks to new revolutionary technologies, information has become an invaluable commodity. Governments and big companies realized this phenomenon some time ago, when Internet appeared, but common users still fail to see how their traces are feeding a huge business at the expense of their own privacy.

The fact is that each movement on Internet is susceptible to be traced, processed and used to tailor personalized services according to the user's interests. Search queries, tags, clicks and e-mails are part of these traces, and a collusion of these information services would suppose a serious risk to the user's privacy.

Many theoretical proposals have been made to address these risks of privacy, but available applications mostly concentrate on blocking some user interactions to prevent leakage of certain traces. In [8], a novel purpose was done for the development of a user agent (a Firefox extension) capable of measuring (and showing) the user's privacy. This measuring is based on a profiling process made, basically, over the search queries sent by the user.

We contributed to complement the privacy measuring tool proposed in [8], by developing modules to include social tags and not only search queries, as part of the measuring process. Strategies of potential adversaries goes beyond the collection of search queries, since it also may include tags, ratings, fingerprints, etc. Tags are rich pieces of information that involve much semantic when they are submitted to social tagging services, so their contribution is very important to accurate measure the user's privacy.

Among the available privacy protection applications, we have TrackMeNot. It is a widely known Firefox extension which implements a data perturbation mechanism to obfuscate the user's profile as seen by a potential attacker. It generates fake queries which are sent along with the user's queries in order to mimic the user search behavior.

We evaluated the efficiency of TrackMeNot on its way to increase the user's privacy. Basically the privacy gain was measured for 50 users (obtained from a public dataset of query logs) by comparing the privacy levels before

and after applying the perturbation mechanism. Some TMN configuration parameters were modified along the evaluation to determine their impact on the privacy gain.

Two available profiling services were used in order to simulate different adversary models: the Adnostic's profiling module and the TextWise categorization API.

TrackMeNot obfuscation mechanism worked very well by increasing the privacy of users in about 20% in average when it was measured in terms of entropy and divergence (with respect to the average profile of the population). This result was obtained when the amount of bogus queries was the same as the real ones. Against classification attacks, however, the results of the evaluation were not so successful, since the impact of fake queries were not enough to change the original classification of the users.

As expected, the level of privacy was increased as the number of bogus queries was greater. This clearly defined the inherent tradeoff between privacy and total traffic produced by the additional queries.

6.2 Future work

The inclusion of user data from other social tagging systems would increase the accuracy of the user profiling in the browser and, as consequence, the privacy measurement. For example, Twitter messages including hashtags could provide more information to build the local user profile, although processing such information may be more difficult.

As we saw during the evaluation of TMN, information including search query logs or statistics about global preferences is very valuable to measure the user's privacy, especially when we consider a classification attack. Sadly, this information is in the hands of the potential adversaries.

It seems that a smarter mechanism of search query generation must be developed; a mechanism capable of changing its behavior according to the particularities of the user profile. Probably, it should accept direct configuration from the user, who could tune it to adapt it to his personal needs.

Even when it works very efficiently, the TextWise API offers a limited service, so the implementation of another profiling engine will really help in improving the quality of measured user's profiles. Natural language

processing and machine learning are very wide research fields that could be really useful to develop a profiling module like this.

A pending work is definitely the analysis of the tradeoff between privacy and additional data transmitted, even when the traffic generated by search requests is not so important in comparison with other kinds of traffic.

7. Appendix

7.1 Profiling module based on TextWise categorization service

This script, programmed using Python contains functions to receive a list of text files containing user queries, to then profile them by means of the TextWise categorization API. Finally XML files are built to contain the profile of every user, according to the categorization previously performed.

```
   import xml.dom.minidom
   import math
   import datetime
   from time import gmtime, strftime
   import urllib2
   import json
   import time
   # ---- Create a DOM object
   def get_a_document():
       doc = xml.dom.minidom.Document()
       perfil_root_element =
doc.createElementNS("http://www.upc.edu/perfil", "perfil")
       doc.appendChild(perfil_root_element)
       return doc, perfil_root_element

   # ---- Save a DOM document
   def save_doc(doc,archivo):
       text_file = open(archivo, "w")
       text_file.write(doc.toxml())
       text_file.close()

   # ---- Update DOM
   def actualizar_categoria(doc,perfil_root_element,cat_name):
       if doc.childNodes[0].getElementsByTagName(cat_name).length !=0:
           sconteo =
doc.childNodes[0].getElementsByTagName(cat_name)[0].getAttribute("count")
           conteo = int(sconteo) + 1

doc.childNodes[0].getElementsByTagName(cat_name)[0].setAttribute("count",st
r(conteo))
       else:
           category_element =
doc.createElementNS("http://www.upc.edu/perfil",cat_name)

category_element.setAttributeNS("http://www.upc.edu/perfil","count",'1')
           perfil_root_element.appendChild(category_element)

       if doc.childNodes[0].getElementsByTagName(cat_name) != []:
           sconteo =
doc.childNodes[0].getElementsByTagName(cat_name)[0].getAttribute("count")
           conteo = int(sconteo) + 1
```

```
doc.childNodes[0].getElementsByTagName(cat_name)[0].setAttribute("count",st
r(conteo))
            return True
        else:
            category_element =
doc.createElementNS("http://www.upc.edu/perfil",cat_name)

category_element.setAttributeNS("http://www.upc.edu/perfil","count",'1')
            perfil_root_element.appendChild(category_element)
            return False

    # ---- Read query files
    def get_queries(archivo):
        with open(archivo) as f:
            queries = f.read().splitlines()
        for x in range(0,len(queries)):
            queries[x]=queries[x].replace(" ","%20")
        return queries

    # ---- Categorize query
    def categorizar_query(consulta):
        response =
urllib2.urlopen('http://api.semantichacker.com/7pbzu5f0/category?content='+consulta+'&showLabe
ls=true&useShortLabels=true&format=json&nCategories=5')
        data=json.load(response)
        # Return list of queries
        lista = data[u'categorizer'][u'categorizerResponse'][u'categories']
        listacats = []
        time.sleep(0.5)
        for i in lista:
            listacats.append(i[u'label'])
        for i in range(0,len(listacats)):
            listacats[i]=listacats[i].replace('/','.')
            if all(c in listacats[i] for c in ',')==True:
                listacats[i]=listacats[i].replace(',','')
            if all(d in listacats[i] for d in "'")==True:
                listacats[i]=listacats[i].replace("'",'')
        return listacats


    # ---- Get the user profile from a file that contains his queries
    def profile_user(archivo):
        doc, perfil_root_element = get_a_document()
        queries = get_queries('fakequeries\\'+archivo)
        i=0
        nocat=0
        for q in queries:
            listacats = categorizar_query(q)
            if listacats != []:
                for cat in listacats:
                    actualizar_categoria(doc,perfil_root_element,cat)
                    i=i+1
            else:
                i=i+1
                nocat=nocat+1
        print str(nocat)+' de '+str(i)+' sin categoria\n'
        save_doc(doc,'fakeprofiles\\'+archivo+'.xml')
        return doc
```

```
  # Save log messages
  def tologfile(mensaje):
      with open("logfile.txt", "a") as text_file:
          text_file.write(strftime("%a, %d %b %Y %H:%M:%S", gmtime())+' :
'+mensaje+'\n')

  # ---- Main execution

  # Leer lista de usuarios (ids)
  with open('listofusers') as f:
      ids = f.read().splitlines()
  # de cada archivo en la lista hacer el profiling
  for i in ids:
      tologfile('INICIANDO PROFILING DE USUARIO: '+i)
      profile_user(i)
```

7.2 Code to capture and profile the category tag from Youtube video

This code is executed each time a new web page is displayed on the browser. It detects if the user is browsing on the Youtube site and if so it retrieves the category string that the owner assigned to the video our user is watching. The code processes the string and profiles so that the corresponding category is updated in the user's profile. This sort of label is identified by an ID called eow-category.

```
var win =  ADNOSTIC.utils._getRunningWindow();
var url = win.gBrowser.contentDocument.baseURI;
url_array = [];
url_array = url.split('/');
if(url_array[2] == "www.youtube.com"){
     var tagObj = win.gBrowser.contentDocument.getElementById("eow-category");
     tags= tagObj.childNodes[0].childNodes[0].data
     tagArray = tags.split("&")
     for (var i=0;i<tagArray.length;i++){
                        tagArray[i] = tagArray[i].toLowerCase().replace(/[^a-zA-Z0-9ñ]/g,'');
                        win.ADNOSTIC.profiler._profileDelic(tagArray[i])
                        alert(tagArray[i])
        }
}
```

8. References

1. The Guardian, "NSA Prism program taps in to user data of Apple, Google and others", URL http://www.guardian.co.uk/world/2013/jun/06/us-tech-giants-nsa-data.
2. Omer Tene , "What Google Knows: Privacy and Internet Search Engines", 2008.
3. Y. Elovici, C. Glezer, and B. Shapira, "Enhancing customer privacy while searching for products and services on the World Wide Web," Internet Res., vol. 15, no. 4, pp. 378–399, 2005.Obfuscation-Based Private Web Search.
4. Visual News, "How Much Data is Created Every Minute", URL http://www.visualnews.com/2012/06/19/how-much-data-created-every-minute/
5. TechCrunch, "AOL Proudly Releases Massive Amounts of Private Data", URL http://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/.
6. Arvind Narayanan y Vitaly Shmatikov, "Robust De-anonymization of Large Sparse Datasets (How to Break Anonymity of the Netflix Prize Dataset)". Security and Privacy, 2008. SP 2008. IEEE Symposium on, C1, 2008.
7. Michael Barbaro and Tom Zeller Jr., "A Face Is Exposed for AOL Searcher No. 4417749". En The New York Times, Technology, URL http://www.nytimes.com/2006/08/09/technology/09aol.html?page wanted =all, August 2006.
8. Estrada-Jiménez, José, "Implementation of a Firefox Extension that Measures User Privacy Risk in Web Search", Master Project, Universitat Politecnica de Catalunya, 2013.
9. J. Parra-Arnau, D. Rebollo-Monedero, J. Forné, "Measuring the Privacy of User Profiles in Personalized Information Systems". Future Generation Computer Systems, 2013.

10. Wang, Y., & Kobsa, A. "Privacy-enhancing technologies. Social and Organizational Liabilities" in Information Security, 203-227, 2006.
11. Shen, X., Tan, B., & Zhai, C., "Privacy protection in personalized search". In ACM SIGIR Forum (Vol. 41, No. 1, pp. 4-17). ACM, 2007.
12. Ostrovsky, R., & Skeith III, W. E., "A survey of single-database private information retrieval: Techniques and applications". In Public Key Cryptography–PKC 2007 (pp. 393-411). Springer Berlin Heidelberg, 2007.
13. Canny, J., "Collaborative filtering with privacy". In Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on (pp. 45-57). IEEE, 2002.
14. Chaum, D. L., "Untraceable electronic mail, return addresses, and digital pseudonyms". Communications of the ACM, 24(2), 84-90, 1981.
15. Reiter, M. K., & Rubin, A. D., "Crowds: Anonymity for web transactions". ACM Transactions on Information and System Security (TISSEC), 1(1), 66-92, 1998.
16. Erola, A., Castellà-Roca, J., Viejo, A., & Mateo-Sanz, J. M., "Exploiting social networks to provide privacy in personalized web search". Journal of Systems and Software, 84(10), 1734-1745, 2011.
17. Rebollo-Monedero, D., Forné, J., & Domingo-Ferrer, J., "Query profile obfuscation by means of optimal query exchange between users". Dependable and Secure Computing, IEEE Transactions on, 9(5), 641-654, 2012.
18. Chow, C. Y., Mokbel, M. F., & Liu, X., "A peer-to-peer spatial cloaking algorithm for anonymous location-based service". In Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems (pp. 171-178). ACM, 2006.

19. Rebollo-Monedero, D., & Forné, J., "Optimized query forgery for private information retrieval". Information Theory, IEEE Transactions on, 56(9), 4631-4642, 2010.
20. Howe, D. C., & Nissenbaum, H., "TrackMeNot: Resisting surveillance in web search". Lessons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society, 417-436, 2009.
21. Domingo-Ferrer, J., Solanas, A., & Castellà-Roca, J., "*k*-private information retrieval from privacy-uncooperative queryable databases". Online Information Review, 33(4), 720-744, 2009.
22. Polat, H., & Du, W., "Privacy-preserving collaborative filtering using randomized perturbation techniques". In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on (pp. 625-628). IEEE, 2003.
23. D. Rebollo-Monedero, J. Parra-Arnau, Claudia Diaz and J. Forné, "On the Measurement of Privacy as an Attacker's Estimation Error". Springer, International Journal of Information Security, vol. 12, n. 2, pp. 129-149, 2013.
24. Millen, David R., and Jonathan Feinberg. "Using social tagging to improve social navigation." Workshop on the Social Navigation and Community based Adaptation Technologies. 2006.
25. Delicious, URL http://www.delicious.com.
26. Flickr, URL http://www.flickr.com.
27. Youtube, URL http://www.youtube.com.
28. Delicious BookMarks, URL https://addons.mozilla.org/es/ firefox/ addon/delicious-bookmarks/ reviews/.
29. V. Toubiana, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy Preserving Targeted Advertising ∗". Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS), 2009.
30. E. Schmidt, "Global Privacy Standards", URL http://www.peterfleischer.blogspot.com/, 2007.

31. TextWise, URL http://www.textwise.com/
32. Delicious dataset, URL http://www.dai-labor.de/en/competence%20centers/irml/datasets/
33. Eckersley, Peter. "How unique is your web browser?." Privacy Enhancing Technologies. Springer Berlin Heidelberg, 2010.
34. Bogers, Toine, and Antal Van den Bosch. "Collaborative and content-based filtering for item recommendation on social bookmarking websites." Submitted to CIKM 9 (2009).