# Tracking Users on the Internet with Behavioral Patterns: Evaluation of its Practical Feasibility

Christian Banse[1], Dominik Herrmann[2], and Hannes Federrath[2]

[1] Fraunhofer AISEC, Garching b. München, Germany
christian.banse@aisec.fraunhofer.de
[2] University of Hamburg, Department of Informatics, Germany
{herrmann,federrath}@informatik.uni-hamburg.de

**Abstract.** Traditionally, service providers, who want to track the activities of Internet users, rely on explicit tracking techniques like HTTP cookies. From a privacy perspective *behavior-based tracking* is even more dangerous, because it allows service providers to track users passively, i. e., without cookies. In this case multiple sessions of a user are linked by exploiting characteristic patterns mined from network traffic.
In this paper we study the *feasibility* of behavior-based tracking in a real-world setting, which is unknown so far. In principle, behavior-based tracking can be carried out by any attacker that can observe the activities of users on the Internet. We design and implement a behavior-based tracking technique that consists of a Naive Bayes classifier supported by a cosine similarity decision engine. We evaluate our technique using a large-scale dataset that contains all queries received by a DNS resolver that is used by more than 2100 concurrent users on average per day. Our technique is able to correctly link 88.2 % of the surfing sessions on a day-to-day basis. We also discuss various countermeasures that reduce the effectiveness of our technique.

## 1 Introduction

Tracking users on the Internet is perceived as a serious infringement of their privacy, especially when it is done *without their consent*. Nevertheless, tracking cookies, which allow content providers to link multiple sessions of a user, are used on many popular websites today [2]. This allows ad networks such as Google Analytics, Doubleclick or Facebook to create usage profiles from the websites a user interacts with over time. Service providers that require authentication can also track the actions of their users. Apart from the Internet Service Provider this applies to, among others, commercial anonymization services that rely on single-hop proxies or VPNs (e. g., anonymizer.com and ipredator.se) and the providers of browser toolbars (e. g., *Alexa* and *Web of Trust*). We call this practice *explicit tracking*, because it uses well-known techniques, relies on the cooperation of the web browser or the user and can be detected rather easily.

Instead of explicit tracking we are interested in the feasibility of **behavior-based tracking** techniques, which rely on characteristic patterns within the

This is the authors' version, which contains slightly revised figures. The original publication, which is available at SpringerLink, appeared in Proc. of SEC 2012, Springer AICT 376, pp. 235–248, 2012.

activities of the users. In contrast to explicit tracking it can be used to link multiple Internet sessions of a user *passively*, i. e., without the user's cooperation and even in the absence of unique identifiers or tracking cookies. Behavior-based tracking constitutes a considerable privacy threat, because it can not only be carried out *without consent*, but it is also *imperceptible*, i. e., it cannot be detected. As the recent discussions regarding the Do-not-Track initiative revolved mainly around tracking cookies, behavior-based tracking may come as a surprise even for well-informed users. For example, users of an alternative DNS resolver (like OpenDNS or Google DNS)[3], a service provider that cannot use explicit tracking for technical reasons, do probably not expect that their resolver can track their activities. As our results show, DNS resolvers are in a good position for behavior-based tracking, though.

Behavior-based tracking of users is "challenging" because many Internet Service Providers assign their customers dynamic, periodically changing IP addresses [25,24]. Recent research on user profiling techniques may help curious service providers to overcome this hurdle: given a database with traffic profiles of a set of users, data mining techniques can be used to map traffic with unknown origin to one of the users from the database [15,26,13]. The mapping becomes more difficult for larger numbers of users and for smaller amounts of training data [26]. The actual privacy threat of behavioral profiles is unknown so far, because previously published results were obtained with a limited number of concurrent users (up to 100 in [26]) in a closed-world evaluation setting. Therefore, we address the following research question in this paper: *To what degree is behavior-based tracking feasible in practice and how can users protect themselves against it?* This question can be tackled by applying techniques that are known to work well for behavioral profiling to traffic of a large number of Internet users. Existing profiling techniques are not suitable for our tracking problem, though (cf. Sect. 2). Therefore, based on previous work we start out by devising a tracking technique and establish its suitability for behavioral profiling. We then analyze the real-world feasibility within an evaluation setup that incorporates the challenges to be faced in practice.

**Our contribution.** We design a tracking technique that exploits behavioral characteristics, implement a scalable evaluation environment with a *MapReduce* framework and evaluate our technique in a realistic setting with a large-scale dataset. Given more than 2100 concurrent users on average per day, we find that behavior-based tracking is feasible with very high accuracy for a third of the users and with moderate accuracy for the remaining ones. On overall 88.2 % of all session are linked correctly. The high accuracy is achieved by resolving ambiguous results caused by fluctuating users. To the best of our knowledge we are the first to assess large-scale behavior-based tracking in a real-world setting.

The paper is structured as follows. We present related work in Sect. 2 before modeling the tracking problem in Sect. 3 and describing our dataset in Sect. 4. Our tracking technique is illustrated in Sect. 5 and evaluated in Sect. 6. Finally, we discuss countermeasures in Sect. 7 before we conclude in Sect. 8.

[3] cf. http://www.opendns.com and http://code.google.com/speed/public-dns

## 2 Related Work

In the following we outline related research efforts that analyze the possibility to re-identify users based on their traffic or behavioral profiles. They differ from our work in terms of application, techniques and evaluation methodology as well as the size of their dataset.

Yang [19,27,26] analyzes visitation patterns of web users as an additional authentication factor to tackle identity theft and fraud in e-commerce applications. She assumes that an e-commerce provider (e. g., a web shop) has access to user profiles built from multiple surfing sessions of its users. Each time a user logs into the web shop the provider is supposed to retrieve the most recent surfing sessions from the user in order to confirm the user's identity. To this end Yang constructs user profiles from the hostnames of the web sites visited within a surfing session. She proposes two profiling techniques based on *support* and *lift*, which are common measures in the field of *association rule mining*. The effectiveness of the techniques is evaluated and compared with the J4.8 decision tree classifier from Weka [12] and with Support Vector Machines [8]. In a scenario with 100 concurrent users the predictive accuracy of her techniques reaches 87 % for profiles built from 100 training sessions. The accuracy drops to 62 %, if only one training session is available, which corresponds to our behavior-based tracking scenario. Yang's results demonstrate that characteristic behavior can be exploited for user re-identification in her e-commerce scenario with a limited number of users. Its feasibility for large-scale re-identification problems or behavior-based tracking with many users cannot be deduced from her results, though.

Kumpost's user re-identification technique relies on the destination IP addresses for HTTP(S) and SSH connections of each user [15]. His user profiles consist of sparse access frequency vectors that contain the number of connections to each destination IP address. This is similar to our hostname-based profiles. Kumpost's re-identification methodology employs the *inverse document frequency* transformation and the *cosine similarity metric*. Kumpost evaluates the effectiveness of the technique using *monthly aggregated* NetFlow logs. Unfortunately, he fails to provide statistics regarding the size of the dataset, i. e., the number of concurrent users. Although monthly profiles should contain a considerable amount of information, Kumpost reports rather high false-positive rates of 68 % for HTTP traffic and 21 % for SSH traffic.

In a previous publication we proposed a user re-identification technique based on the Multinomial Naïve Bayes classifier from Weka [13]. The classifier was evaluated using the HTTP traffic of 28 volunteering users. Given one training session per user, up to 73 % of sessions were classified correctly. In comparison to [13] the tracking technique presented in Sect. 5 utilizes an improved feature extraction technique with *n-grams* and incorporates an additional *cosine similarity decision engine*. Moreover, the evaluation in Sect. 6 is carried out on a much larger dataset.

## 3 Modeling Behavior-Based Tracking

Building on the promising results of previous studies described in Sect. 2, our behavior-based tracking scheme solely relies on the web sites visited within a surfing session. Order of requests and exact timings are neglected in this model.

We assume that each user is represented by a dynamic IP address that is replaced by a different one after a fixed amount of time, i. e., there are epochs $e_i \in E$ of constant length (e. g., 24 hours, beginning at 4:00 am each day). Moreover we assume that the service provider that wants to use behavior-based tracking can record the interactions of its users with a set of destination hosts $H$. Each interaction is stored as a triplet $(e, s, h)$ consisting of the epoch $e$ it was observed in, as well as the corresponding source address $s$ and the destination host $h$ (in our case: a hostname). A *user session* is then obtained by aggregating all events that share the same $e$ and $s$ to obtain the access frequency of each destination $h$. Each user session can be mapped to an $|H|$-dimensional vector, where the $i$-th component corresponds to the access frequency of the $i$-th destination according to a totally ordered list of all observed destinations $H$. Figure 1 summarizes the transition from the real-world view of the service provider to our model.

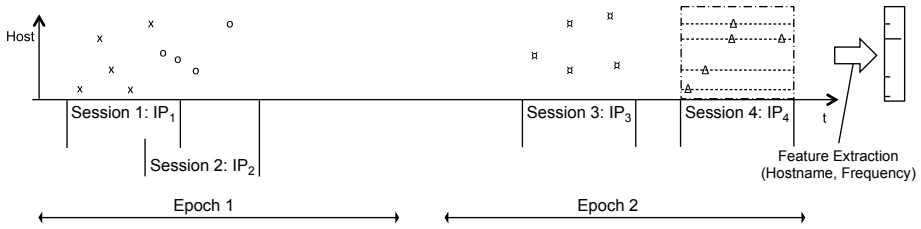



Fig. 1: Behavior-based tracking scenario

## 4 Dataset

We evaluate the feasibility of behavior-based tracking using *DNS queries*, i. e., from the viewpoint of a curious DNS resolver. Our dataset was obtained in cooperation with the computer center of the University of Regensburg, Germany. It contains all DNS requests, which were received by the two DNS resolvers of the university network between February 1, 2010 and June 30, 2010. In order to maintain a high level of privacy, every source IP address $s$ was replaced by a pseudonym $u = h(s|r_{\text{const}})$ using a hash function $h$ before the record was stored. The actual salt value $r$ was not disclosed to us and thrown away once data collection was completed. Nevertheless, this practice cannot ensure a satisfactory level of privacy for all users: some users may issue personally identifying queries (e. g., for their weblog at *username*.blogspot.com), which undermine our

pseudonymization efforts. Therefore, we limit access to the dataset to a small group of researchers.

In total the dataset contains 2,645,748,393 queries for 77,662,943 unique hostnames issued by 18,904 unique source IP addresses. To maintain a high data quality we only consider queries originating from the student housing network segment in this paper. Each network device must be personally registered by its resident and is assigned a *unique, static* IP address.[4] The majority of the 4153 considered users in our dataset are students of the University of Regensburg, which use their Internet access for research and studying as well as private browsing. On average 2123 users were active per day.

During the evaluation of the tracking technique we simulate *dynamically changing IP addresses*, i. e., *a priori* our classifier (cf. Sect. 5) does not know which sessions belong to a given user. Of course, due to the static IP addresses used within the campus network, we *do* know all the sessions that belong to a given user. Thus, we can compare the predictions of our tracking technique with the actual set of sessions of each user to measure its accuracy.

## 5 Behavior-Based Tracking Technique

### 5.1 Multinomial Naïve Bayes Classifier

The behavior-based tracking problem can be formulated as a classification problem [18]. Every session corresponds to a single *instance* $x \in X$ which is represented by its attribute vector (cf. Sect. 3). Each instance belongs to a class $c \in C$, which corresponds to a certain user $u$ (cf. Sect. 4). A set of *training instances*, whose class assignments are known (Sessions 1 and 2 in Fig. 1), is used to build a predictive model. Based on the model the *classifier* assigns the most probable class to each *test instance*, whose class is supposed to be unknown in the experiment (Sessions 3 and 4 in the example).

For the *Multinomial Naïve Bayes classifier* **(MNB classifier)** [18] used in this paper the probability that a test instance $x$ belongs to a certain class $c_i$ is defined as

$$P(c_i|x) \sim \prod_{h \in H} P(h|c_i)^{f_{h,x}}.$$

The classifier assigns the test instance $x$ to the class with the highest probability $P(c_i|x)$. In our case $P(h|c_i)$ represents the probability that a hostname $h$ is to be found in the training instances of class $c_i$. $f_{h,x}$ specifies the frequency with which $h$ occurs in the given test instance $x$. The rationale behind this formula is: the more often frequently accessed hosts seen during training of a class do appear in the given test instance, the more likely does the test instance belong to that class.

[4] Each address is supposed to be exclusively used by one resident; in some rare cases devices may be used by multiple users, though.

The selection of the MNB classifier for behavior-based tracking may seem unintuitive at first sight. After all, it is usually only applied to text categorization problems (e. g., in spam filters) [18]. One aspect that motivates the application of the MNB classifier for the problem at hand is the fact that the features used in our model (access frequencies of hostnames) and the features used for text categorization (term frequencies) have an important similarity: both of them are subject to power law distributions [1,29]. Another argument in favor of the MNB classifier is its little computation complexity in comparison to more sophisticated classifiers such as *Support Vector Machines* (SVMs) [8]. In Yang's experiments the application of SVMs increased classification runtimes by 300 % [26].

### 5.2 Transformation of Frequency Vectors

The text categorization discipline makes use of numerous transformation techniques in order to boost classification accuracy. We implemented the most promising techniques as described below.

The first technique consists of a transformation of the individual access frequencies $f_{h,x}$ of an instance vector. In our case $f_{h,x}$ specifies how often a hostname $h$ occurs in a session $x$. The frequencies are scaled down by a sub-linear transformation $\log(1+f_{h,x})$ in order to minimize the influence of extremely large frequency values [23, p. 311]. Additionally, the frequency vectors of all classes are normalized to a uniform Euclidean length [23, p. 310]. The application of the two transformations is indicated by the use of **TFN** in this paper.

Additionally, the frequencies $f_{h,x}$ can be multiplied by the *Inverse Document Frequency* $idf_h = \log \frac{N}{n_h}$ [23, p. 311], where $N$ is the number of all instances and $n_h$ the number of all instances containing the hostname $h$. The application of the *IDF* reduces the weight of common hostnames that are accessed by a large number of different users. The combination of the IDF and TFN transformation is designated by **TFIDFN**.

Finally, *n-grams* are known to increase the accuracy for many text mining problems [3,6,9]. They have also been used for traffic analysis problems with success [21]. This technique combines the hostnames of $n$ adjacent events to obtain a composite hostname. We denote the usage of n-grams by appending a suffix to the configuration name, e. g., **TFN-1+2** indicates that – in addition to the original hostnames – bi-grams are added to the instances.

### 5.3 Resolving Ambiguous Predictions with Cosine Similarity

According to our model only a single user can issue queries from a certain IP address within an epoch (cf. Sect. 3). Therefore, not more than one instance should be mapped to each class within an epoch. If the classifier *does* assign more than one test instance to a given class, we resort to an additional decision criterion to resolve the ambiguous prediction, i. e., to single out the instance that fits best (cf. Sect. 6.3). For this purpose we employ the *cosine similarity metric*.

The cosine similarity is a [0;1]-normalized similarity measure between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ that reflects the angle $\theta$ which is spanned by them. A smaller $\theta$

corresponds to a larger value of the cosine similarity. The similarity $s$ is equal to the quotient of the dot product of $\boldsymbol{x}$ and $\boldsymbol{y}$ and the product of their magnitudes:

$$s_{\boldsymbol{x},\boldsymbol{y}} = \cos\theta_{\boldsymbol{x},\boldsymbol{y}} = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|}$$

Ambiguous predictions are resolved by computing the similarity values between the training instance of the class and all the test instances that have been assigned to it. The test instance that achieves the highest similarity value is assigned to the class, all other instances are discarded.

## 6 Evaluation

Due to the large size of the dataset we did not use off-the-shelf data mining tools such as *Weka* [12] for the evaluation. Instead, we ported the MNB classifier based on the Weka source code to Apache Hadoop [22], a *MapReduce* [10] framework implemented in Java. We obtained the results presented in this paper on a cluster of 12 quad-core machines (Intel Core i5, 3.1 GHz, 8 GB RAM). The running time of individual experiments was below 24 hours most of the time, which enabled us to try out many parameters. In the following we will report our evaluation methodology and the most significant results.

### 6.1 Cross Validation

Firstly, we analyze the effectiveness of the classifier and the transformations presented in Sect. 5.2 with a series of *cross validation* experiments. Cross validation ensures that the results obtained are not biased due to selection of peculiar training instances [14]. In order to obtain meaningful results, a *stratified* dataset is used, which contains the same number of instances for all classes.

For the cross validation experiments we focused on the very active period between March 1, 2010 and June 30, 2010, to include as many users as possible. We randomly selected 3000 users $u \in U$, which were able to contribute at least 20 instances of 24 hours length in the mentioned period. 20 instances were randomly selected from each user resulting in a cross validation dataset $D$ of 60,000 instances in total.

In the following we illustrate the evaluation procedure for the case of *10-Fold* Stratified Cross Validation: first of all, the 20 instances of each class are split randomly, yet equally to obtain 10 disjoint sets (folds): $D = D_1 \cup D_2 \cup \ldots \cup D_{10}$. The evaluation takes place in 10 runs $k \in 1, \ldots, 10$. In the $k$-th experiment the instances $D_{\text{train}} = D \setminus D_k$ are used to train the classifier. $D_{\text{train}}$ contains $\frac{9}{10}$ of all instances of every user $u$. With $I^u_{\text{train}}$ denoting the training instances of one user, we have $D_{\text{train}} = I^{u_1}_{\text{train}} \cup \ldots \cup I^{u_{3000}}_{\text{train}}$. Based on the supplied training data the classifier has to predict the appropriate classes for the instances in $D_k$. The overall accuracy is finally obtained as average of all 10 runs.

For each experiment we compute the average *precision* and *recall* values, two metrics, which are used to analyze the prediction accuracy of data mining

techniques. Both metrics are bound to the interval [0;1]. Assuming the classifier assigned $n$ instances to some class $c_i$, where $m \leq n$ of the instances actually do belong to $c_i$, then the precision of this result is given by the ratio $\frac{m}{n}$, i. e., it expresses the pureness of the result. On the other hand, the recall expresses whether the classifier "found" all test instances of $c_i$: given $l$ test instances of $c_i$ the recall value is equal to $\frac{m}{l}$. Thus, a service provider that tries to track a user as extensively as possible over a period of time is primarily interested in high recall values. If the objective is to minimize the number of false mappings, a high precision value is of high importance.

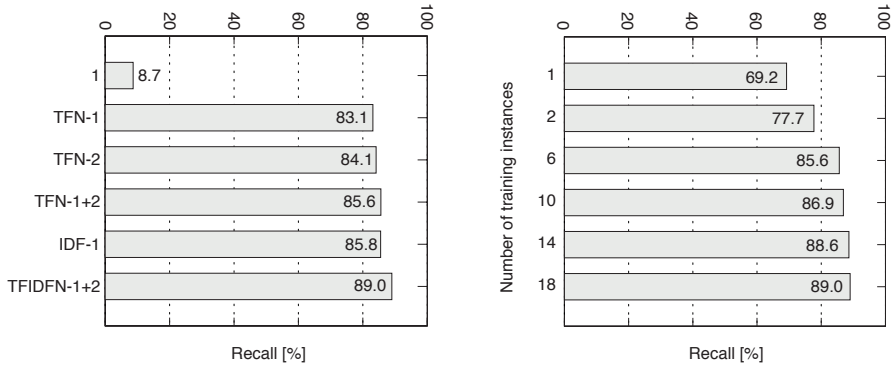
For ease of exposition we only report the average recall values in the following. In all experiments the average precision values were constantly above the recall values by about 3 percentage points. Figure 2a shows the results for different configurations of the classifier with $|I_{\text{train}}| = 18$ training and 2 test instances per user. For 1-grams we obtain a recall of 83.1 % using the *TFN* transformation. The remaining transformations increase the recall further. The configuration *TFIDFN-1+2* achieves the best result with an average recall of 89.0 %. Higher-order n-grams have no more positive effects.

According to the results the selected text mining techniques are suitable for behavioral user profiling. The high accuracy obtained in the previous experiment is of little relevance for behavior-based tracking in practice, though, because we do not expect the service provider to have $|I_{\text{train}}| = 18$ training instances for each user at his disposal. Hence, we repeated the cross validation experiments with a smaller amount of training data. As expected, the recall values of these experiments show that with a decreasing number of training instances it is increasingly difficult for the classifier to assign the test instances to the correct class. In case of *a single training instance* the recall drops to 69.2 %. Even for a large user group the behavior of most Internet users seems to exhibit sufficiently characteristic attributes, which are present in a large share of their sessions.

*Exploited Characteristics.* We manually reviewed a sample of the trained model to understand the patterns the classifier relies on. As expected the classifier mainly exploits patterns in the web surfing behavior of the users, which are reflected by the DNS queries pretty well; only repeated queries for the same hostnames cannot be observed due to caching. For some users the classifier also relies on patterns introduced by the installed applications on their machine (visible due to update checks) or environmental peculiarities (e. g., the Windows Network Browser looking up its neighbors).

### 6.2 Evaluation in Real-World Setting

After having demonstrated the fundamental suitability of our technique in Sect. 6.1, we proceed with experiments that apply it to the actually recorded traffic. This will allow us to study its feasibility in a real-world setting. To this end, we have implemented a fully-automated experimental environment with the Hadoop framework that allows us to simulate a service provider that tries to track all users from one day to the following day, i. e., the *epochs* (cf. Sect. 6.2)




(a) $|I_{\text{train}}| = 18$ (b) TFIDFN-1+2

Fig. 2: Cross validation: observed recall values for 3000 users

start at midnight and last 24 hours.[5] The simulation iterates over all days $t$ in chronological order. On a given day $t$ a class $c_i$ is set up for each active user $u_i$ and the MNB classifier is trained with all instances $x$ present on that day. The learned model is used to predict the most probable classes for all instances from the following day $t + 1$, i. e., to link the sessions of the two consecutive days.

In the case that user $u_i$ is active on two consecutive days, the simulated observer scores a *correct mapping* ($C$) within an iteration, if the classifier predicts that the instance of user $u_i$ on day $t + 1$ belongs to the class $c_i$, which was set up using the instance of $u_i$ from the previous day, *and* if no other instance from $t + 1$ is assigned to $c_i$. If the classifier assigns *exactly one* instance $x_j$, which belongs to a different user $u_j, j \neq i$, to $c_i$, we record a *non-detectable error* ($E_1$). We record a *detectable error* ($E_2$) for ambiguous results, i. e., if instances from multiple users (possibly including $u_i$) are assigned to $c_i$. In the case that user $u_i$ is not active on day $t + 1$ any more, a *correct mapping* is scored, if no instances are assigned to $c_i$ at all. The overall *accuracy* of the classifier is given by counting all mappings of an experiment and computing $|C|/(|C|+|E_1|+|E_2|)$.

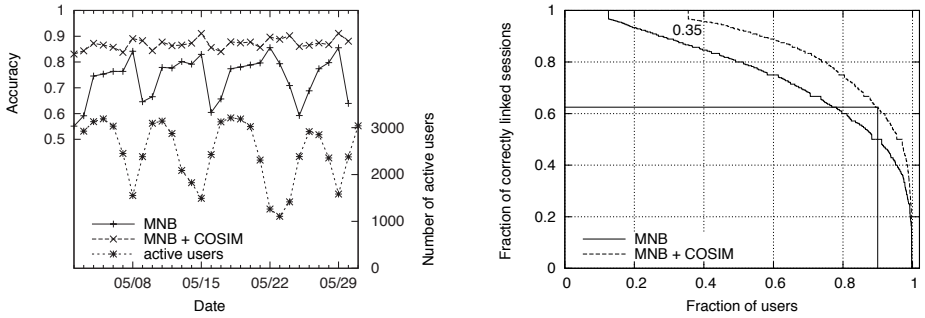
We evaluate the MNB classifier with this scheme for the month of May, which sports the largest number of active users. As we want to report results that are significant for typical users we ranked the users according to their daily query volume and removed the instances of the top 5 % and bottom 5 % of the list. Accuracy decreases by about 2.5 percentage points if this step is skipped.

Using the *TFIDFN-1+2* configuration the classifier achieves an average accuracy of 76.6 %. Thus, tracking users in the real-world setting works better than the recall value of 69.2 % obtained for the cross validation experiment with a single training instance suggests. This counterintuitive result can be explained by

[5] We found that the actual time of day at which the epochs start has a negligible effect on the accuracy.

the average number of active users per day (average: 2412, standard deviation: 695) being smaller than the 3000 concurrent users used in Sect. 6.1.

*Robustness of profiles.* We also studied the effect of the age of the trained profiles. To this end we randomly drew tuples with two dates $(t_i, t_{i+\delta})$, which were exactly $\delta \in \{5, 15, 30, 60, 90\}$ days apart. The instances on $t_i$ were used for training and the classifier had to predict the classes of the instances on $t_{i+\delta}$. Accuracy values degraded rather slowly: from 78.3 % for $\delta = 5$ to 44.7 % for $\delta = 90$. The results suggest that the behavior of many users is rather stable, i. e., tracking is also possible with outdated profiles.




(a) Fluctuation of users

(b) Fraction of correct links

Fig. 3: Tracking in the real-world setting

### 6.3 Dealing with Fluctuating Activity

The real-world setting is more challenging than the cross validation experiment because the classifier is faced with user fluctuation. The classifier will encounter instances, for which no class has been trained on the previous day, because the respective user was inactive on that day. Nevertheless, the default implementation of the MNB classifier will assign such an instance to the most likely class, which causes a (non-)detectable error. In our dataset the fluctuation is caused by students that leave the city on weekends. Once they become active again accuracy drops to about 60 %, while it reaches more than 80 % during the week (cf. Fig. 3a, *MNB* graph). Further analysis of the results of the initial experiment from Sect. 6.2 reveals the extent of this problem: ambiguous mappings account for 13.6 % of all cases (44.6 % of all errors), and in 90.4 % of the ambiguous mappings the correct test instance was in fact part of the set of assigned test instances. Therefore, resolving ambiguous results with the cosine similarity is promising (cf. Sect. 5.3). The effect of this optimization is shown in the graph labeled *MNB+COSIM* in Fig. 3a. On overall the accuracy increases to 88.2 %, i. e., about 85 % of the ambiguous results are resolved correctly.

Not all users exhibit enough characteristic patterns for the behavior-based tracking to be effective. Figure 3b shows the accuracy levels that can be obtained for a certain fraction of users. The *MNB+COSIM* configuration is able to establish a correct mapping for all sessions of 35 % of the users. For 90 % of the users at least 60 % of their sessions were linked correctly.

### 6.4 Future Work

According to our results tracking students based on their DNS queries works quite well. A globally distributed service provider, such as Google's DNS resolver, is faced with additional challenges, though. The queries of multiple users may be issued from a single source IP in case they are hidden behind a NAT gateway, a typical scenario for domestic broadband connections. Moreover, users may be located in different time zones, i. e., queries must be assigned to epochs depending on the location of the respective user. In future work we will study the impact of these issues on the feasibility of our technique.

Another important area of work is the evaluation of our technique in different attack scenarios. Especially concerning is the question whether third-party tracking networks such as Doubleclick, Google Analytics or Facebook can exploit behavioral analysis to overcome privacy-enhancing techniques. Today, visits of different websites within a session can be mapped to a specific user due to a tracking cookie. If users delete their cookies between sessions or if they enable the Private Browsing mode, recurring visits in different sessions cannot be linked any more. With our behavior-based technique third-party trackers may be able to overcome this limitation, though.

## 7 Countermeasures

In this section we present initial findings from our search for countermeasures that potentially mitigate the effectiveness of behavior-based tracking. The results reported below are obtained by repeating the experiment described in Sect. 6.3.

A generic protective measure is to use **anonymizers** like Tor or AN.ON (JonDonym) [11,4] that hide communication patterns from eavesdroppers. We will not discuss anonymizers any further in this paper, as we are interested in countermeasures that directly target the behavior-based tracking technique.

If the data requested by the users is valid for a certain amount of time, which is the case for many DNS records, a **caching system** may be a viable countermeasure to hide repeated queries from the observer. Thus, the user profiles would consist of queries for data that hasn't been requested before and queries for expired entries only. Motivated by the fact that many DNS records have a time-to-live (TTL) value of 24 hours, we can simulate a cache, whose entries expire at the end of each day. As a result instance vectors do not contain the actual number of queries per hostname any more. The effect of such a cache is very limited, though: accuracy only drops from 88.2 % to 80.5 %. While keeping records in the cache for a longer time may further decrease accuracy, this practice may introduce usability and security issues due to stale information.

Another countermeasure that reduces the amount of information available to the classifier may be to **change IP addresses frequently**. We study the effect of shorter sessions by pretending that our users change their IP addresses multiple times per day. This strategy proves to be quite effective: for sessions with a length of three hours we observe an accuracy of 60.4 %, which decreases further to 49.5 % for a length of one hour. Time-based address changes have the drawback that all existing connections are terminated, though. The large address space of IPv6 may allow the design of more sophisticated countermeasures in the future [20]. The extreme case would be to issue each query from a different IP address (each having a different IPv6 address prefix) to minimize linkability.

Finally, we consider the effectiveness of **Range Queries**, a technique that has been proposed to protect the privacy of DNS queries [5,28,17]. It is based on the principle of Private Information Retrieval [7,16] and aims to achieve privacy by hiding each query of a user within a set of $n$ random dummy queries. We implemented a Range Query engine, which adds dummy queries to the instances, and found that the effectiveness of Range Queries largely depends on the values of various parameters, among them the number of dummies $n$, the size of the pool of hostnames to choose the dummies from ($N$), and the actual hostnames used as dummies. As a first result we can report that accuracy can be reduced to as low as 10 %, if $n = 5$ dummies are used per query and dummies are drawn from a set of $N = 5000$ randomly selected hostnames. In future work we will search for configurations that offer a good trade-off between security and bandwidth.

## 8 Conclusion

The behavior-based tracking scheme studied in this paper enables service providers, which have access to the (web) requests of users, to link multiple sessions without cookies or other explicit identifiers. Our analysis shows that behavior-based tracking is feasible in a real-world setting for a large user group: in our DNS query log, in which more than 2000 users are active each day, we correctly link up to 88.2 % of all sessions on a day-to-day basis. Our design manages to achieve this high accuracy, because it is able to resolve ambiguous classification that are caused by fluctuating numbers of users.

Daily changing IP addresses, which are sometimes cited to be important for users' privacy, offer only limited protection against behavior-based tracking. While there are no practical countermeasures so far, we see the introduction of IPv6 as an opportunity to design effective and lightweight techniques that help to prevent profiling and tracking users without their consent.

**Acknowledgments.** We thank Martin Wimmer, head of the computing center of the University of Regensburg and his employees, who enabled the compilation of our dataset. We are grateful to the anonymous reviewers, to Arvind Narayanan, to Christopher Piosecny and to our colleagues Karl-Peter Fuchs and Christoph Gerber for their critical feedback and constructive comments.

## References


1. Adamic, L., Huberman, B.: Zipf's Law and the Internet. Glottometrics 3(1), 143–150 (2002)
2. Ayenson, M., Wambach, D.J., Soltani, A., Good, N., Hoofnagle, C.J.: Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. Available at SSRN: http://ssrn.com/abstract=1898390 (2011)
3. Beesley, K.R.: Language identifier: A computer program for automatic natural-language identification of on-line text. In: Language at Crossroads: Proceedings of the 29th Annual Conference of the American Translators Association. pp. 12–16 (1988)
4. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: a system for anonymous and unobservable Internet access. In: International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability. pp. 115–129. Springer-Verlag, New York, USA (2001)
5. Castillo-Perez, S., García-Alfaro, J.: Evaluation of Two Privacy–Preserving Protocols for the DNS. In: Proceedings of the Sixth International Conference on Information Technology: New Generations. pp. 411–416. Washington, DC, USA (2009)
6. Cavnar, W.B., Trenkle, J.M.: N-Gram-Based Text Categorization. In: Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. pp. 161–175 (1994)
7. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private Information Retrieval. J. ACM 45(6), 965–981 (1998)
8. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning 20(3), 273–297 (1995)
9. Damashek, M.: Gauging Similarity with n-Grams: Language-Independent Categorization of Text. Science 267(5199), 843–848 (1995)
10. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
11. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The Second–Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium. pp. 303–320 (2004)
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. 11, 10–18 (2009)
13. Herrmann, D., Gerber, C., Banse, C., Federrath, H.: Analyzing Characteristic Host Access Patterns for Re-Identification of Web User Sessions. In: Proceedings of the 15th Nordic Conference on Secure IT Systems (NordSec 2010), Lecture Notes in Computer Science, LNCS 7127. Springer-Verlag, Berlin, Heidelberg (2012)
14. Kohavi, R.: A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: International Joint Conference on Artificial Intelligence. vol. 14, pp. 1137–1143. Morgan Kaufmann (1995)
15. Kumpošt, M., Matyáš, V.: User Profiling and Re-identification: Case of University-Wide Network Analysis. In: TrustBus '09: Proceedings of the 6th International Conference on Trust, Privacy and Security in Digital Business. pp. 1–10. Springer-Verlag, Berlin, Heidelberg (2009)
16. Kushilevitz, E., Ostrovsky, R.: Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. In: Proceedings of the 38th annual IEEE Symposium on Foundations of Computer Science. pp. 364–373. IEEE Computer Society (1997)

17. Lu, Y., Tsudik, G.: Towards Plugging Privacy Leaks in the Domain Name System. In: Proceedings of the Tenth International Conference on Peer–to–Peer Computing (P2P). pp. 1–10. IEEE (2010)
18. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK (2008)
19. Padmanabhan, B., Yang, Y.: Clickprints on the Web: Are there signatures in Web Browsing Data? Available at http://knowledge.wharton.upenn.edu/papers/1323.pdf (October 2006)
20. Raghavan, B., Kohno, T., Snoeren, A.C., Wetherall, D.: Enlisting ISPs to Improve Online Privacy: IP Address Mixing by Default. In: Proceedings of the 9th International Symposium on Privacy Enhancing Technologies (PETS '09), Lecture Notes in Computer Science, LNCS 5672. pp. 143–163. Springer-Verlag, Berlin, Heidelberg (2009)
21. Rieck, K., Laskov, P.: Language Models for Detection of Unknown Attacks in Network Traffic. Journal in Computer Virology 2(4), 243–256 (2007)
22. White, T.: Hadoop – The Definitive Guide: Storage and Analysis at Internet Scale. O'Reilly, 2nd edn. (2011)
23. Witten, I.H., Frank, E.: Data Mining. Practical Machine Learning Tools and Techniques. Elsevier, San Francisco (2005)
24. Xie, Y., Yu, F., Abadi, M.: De-anonymizing the internet using unreliable IDs. In: Proceedings of the ACM SIGCOMM 2009 conference on Data communication. pp. 75–86. ACM, New York, NY, USA (2009)
25. Xie, Y., Yu, F., Achan, K., Gillum, E., Goldszmidt, M., Wobber, T.: How dynamic are IP addresses? In: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '07). pp. 301–312. ACM, New York, NY, USA (2007)
26. Yang, Y.: Web user behavioral profiling for user identification. Decision Support Systems 49, 261–271 (2010)
27. Yang, Y., Padmanabhan, B.: Toward user patterns for online security: Observation time and online user identification. Decision Support Systems 48, 548–558 (2008)
28. Zhao, F., Hori, Y., Sakurai, K.: Analysis of Existing Privacy–Preserving Protocols in Domain Name System. IEICE Transactions 93-D(5), 1031–1043 (2010)
29. Zipf, G.K.: The psycho-biology of language. An introduction to dynamic philology. M.I.T. Press, Cambridge/Mass., 2nd edn. (1968)