Behavior-based Tracking: Exploiting Characteristic Patterns in DNS Traffic[☆]

Dominik Herrmann[a,*], Christian Banse[b], Hannes Federrath[a]

[a] *University of Hamburg, Department of Informatics*
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
[b] *Fraunhofer Research Institution for Applied and Integrated Security*
Parkring 4, 85748 Garching b. München, Germany

Abstract

We review and evaluate three techniques that allow a passive adversary to track users who have dynamic IP addresses based on characteristic behavioral patterns, i. e., without cookies or similar techniques. For this purpose we consider 1-Nearest-Neighbor classifiers, a Multinomial Naïve Bayes classifier and pattern mining techniques based on the criteria support and lift.

For evaluation we focus on the case of a curious DNS resolver. Therefore, we analyze the effectiveness of the techniques using a common, large-scale dataset that contains the DNS queries issued by more than 3600 users over the course of two months. We find that behavior-based tracking is feasible: The best technique can link up to 85.4 % of the surfing sessions of all users on a day-to-day basis. Moreover, for tracking to be effective only the most significant features or the most popular hostnames have to be considered.

Our results indicate that users can degrade accuracy by changing their IP addresses more frequently, e. g., every few minutes. On the other hand, we find that the previously proposed DNS "range query" obfuscation techniques cannot prevent tracking reliably. Our findings are not limited to DNS traffic. Behavior-based tracking can be implemented by any adversary that has access to the web requests issued by users or their machines.

Keywords: privacy, anonymity, tracking, DNS, unlinkability, data mining, range queries, pseudonymization

[☆] A preliminary version of this paper appeared in (Banse et al., 2012). Note that this is the author's version of a work that was accepted for publication in *Computers & Security*. Changes resulting from the publishing process, such as editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was published in http://dx.doi.org/10.1016/j.cose.2013.03.012

[*] Corresponding author
Email addresses: herrmann@informatik.uni-hamburg.de (Dominik Herrmann), christian.banse@aisec.fraunhofer.de (Christian Banse), federrath@informatik.uni-hamburg.de (Hannes Federrath)

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 (Ayenson et al., 2011). 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 *active tracking with explicit identifiers* because the service provider transmits a unique identifier to each client, which in turn presents it henceforth. In most cases active tracking can be detected and avoided rather easily.

Instead of active tracking with explicit identifiers we are interested in the feasibility of **behavior-based tracking** techniques that rely on characteristic patterns within the activities of the users. They can be employed *passively*, i. e., without the user's cooperation, and 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. This may come as a surprise to many users. For example, users of a **third-party DNS resolver** (like OpenDNS or Google DNS)[1], a service provider that cannot use explicit identifiers for technical reasons, probably do not expect that their resolver can track their activities. However, as our results show, DNS resolvers are in a good position for behavior-based tracking. Other parties that could implement behavior-based tracking are, for instance: operators of HTTP proxies, proxy-based web anonymizers, advertising networks whose ads are included in a large number of web sites, and global operators of Internet access solutions (e. g., WiFi providers).

Tracking the behavior of users without explicit identifiers is "challenging" because many Internet Service Providers (ISPs) assign their customers dynamic IP addresses that change periodically. In Germany, for instance, market leader Deutsche Telekom issues a new IP address every 24 hours to its residential DSL customers. With behavior-based tracking an adversary can track users in spite of changing IP addresses by recording traffic before each change and applying data mining techniques to map the traffic after the change to the profiles stored in the database. We demonstrated the feasibility of behavior-based tracking in a preliminary version of this paper (Banse et al., 2012). The present version of the paper provides a more detailed analysis.

Our contribution. We review and design tracking techniques that exploit behavioral characteristics. All techniques are evaluated on a common, large-scale dataset. Our evaluation considers two scenarios, namely a controlled setting (cross validation on a stratified dataset) and a real-world setting. We ob-

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

serve that results and insights obtained in the former scenario are not necessarily indicative for the latter, i. e., judging the feasibility of behavior-based tracking solely based on controlled experiments would lead to wrong conclusions. This is partly due to the user fluctuation in the real-world scenario, which we account for by pruning superfluous predictions. We also study the impact of reducing the size of the feature space and the influence of the session length to gain a better understanding of the amount of data necessary for behavior-based tracking to be effective. Finally, we provide new insights into the privacy achievable by range queries, an obfuscation technique for DNS queries proposed previously.

The paper is structured as follows. We present related work in Sect. 2 before modeling the adversary and the tracking problem in Sect. 3. We describe our dataset in Sect. 4 and review the implemented tracking techniques in Sect. 5. Section 6 contains the results obtained in our experiments. We analyze protective countermeasures in Sect. 7 and discuss the generalizability of our results in Sect. 8 before we conclude in Sect. 9.

2. Related Work

Yang et al. study the feasibility of mining characteristic patterns from web surfing sessions (Padmanabhan and Yang, 2006; Yang and Padmanabhan, 2008). In her most recent work Yang analyzes visitation patterns of web users as an additional authentication in e-commerce applications (Yang, 2010). She assumes a service provider (e. g., a web shop) who has access to user profiles built from multiple surfing sessions. Each time a user logs on the provider is supposed to retrieve the most recent sessions from the user in order to confirm the user's identity. Yang constructs user profiles from the hostnames of the web sites visited within a surfing session. She proposes two profiling techniques and evaluates their effectiveness. She also includes a decision tree classifier (Hall et al., 2009) and Support Vector Machines (Cortes and Vapnik, 1995) in her benchmarking. In a *controlled setting* with up to 100 concurrent users the predictive accuracy of her techniques reaches up to 87 % when profiles are built using 100 training sessions. In case of a single training session, which is equivalent to our tracking scenario, accuracy drops to 62 %. Yang's results demonstrate that characteristic behavior can be exploited given a limited number of concurrent users. However, Yang concedes that she was "not able to extend the experiments to much larger number of users," and therefore one "cannot conclude that [her] method can be applied to large scale problems." (Yang, 2010, p. 269). Our implementations demonstrate that user re-identification and behavior-based tracking *are* feasible on a larger scale: Using off-the-shelf desktop machines we were able to apply the techniques to 3000 concurrently active users, obtaining recall values of up to 70.1 % in case of a single training instance (cf. Sect. 6.1 and Fig. 2, in particular). One cannot extrapolate from the results obtained with our dataset to results attainable for larger user groups, but we expect accuracy to degrade rather slowly with increasing numbers of users. Moreover, the resource consumption of our techniques is no prohibitive bottleneck as it can be limited by attribute selection (cf. Sect. 6.4).

Kumpošt and Matyáš describe their efforts to re-identify users based on the destinations of their HTTP, HTTPS and SSH connections (Kumpošt, 2007; Kumpošt and Matyáš, 2009). In their study a user profile consists of a sparse access frequency vector that contains the number of connections to each destination IP address. This approach resembles our hostname-based profiles. Their re-identification technique involves clustering instances according to their *cosine similarity*, taking into account the *inverse document frequency* of hostnames as well as counting the number of commonly visited hostnames. Their evaluation deviates from our scenario, however: They operate with *monthly aggregated* NetFlow logs (cf. Kumpošt, 2009, p. 78), while we focus on shorter sessions with durations on the order of hours up to a few days. Moreover, it is difficult to assess the effectiveness of their technique as they do not provide statistics regarding the number of concurrent users or the size of their attribute vectors. Although monthly profiles should contain a considerable amount of information, they report rather high false-positive rates: 68 % for HTTP traffic and 21 % for SSH traffic (Kumpošt and Matyáš, 2009).

In previous publications we proposed a user re-identification technique based on the Multinomial Naïve Bayes classifier. In (Herrmann et al., 2012b) we evaluated the classifier using the HTTP traffic of 28 volunteering users. Given one training session per user, up to 73 % of sessions were re-identified correctly. More recently, we demonstrated the utility of extracting *n-grams* and post-processing the classification result using the cosine similarity metric (Banse et al., 2012).

This paper reviews our previous work and extends it in three ways: Firstly, we apply two additional data mining techniques to the behavior-based tracking problem and evaluate their effectiveness in a real-world setting using a common dataset, which is larger than the one used by Banse et al. (2012). Secondly, we also study the effect of feature selection to reduce the complexity of the attack and to gain insights into the relevance of individual attributes. Finally, we empirically analyze to what degree users can prevent behavior-based tracking using technical countermeasures.

3. Modeling Behavior-Based Tracking

Adversary Model and Adversarial Advantage. In this paper we evaluate the tracking techniques using data that is available to a **curious DNS resolver**, which wants to track its users. More generally, the behavior-based tracking attack can be carried out by **any adversary** that has access to the interactions of a set of users with a set of destination hosts H (the hostnames sent to the DNS resolver in our case). The presumed adversary remains **passive**. We consider two *advantages* an adversary may gain from behavior-based tracking:

Long-term profiling ($\mathbf{Adv}_1$) The adversary may be able to link multiple sessions of a single user, who is assigned a different IP address in each of his sessions, over a longer period of time. This will allow the adversary, for

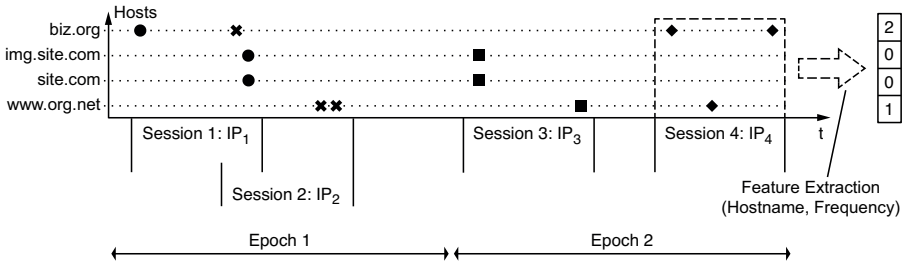


Figure 1: Behavior-based tracking scenario

instance, to construct a comprehensive user profile from the observed interactions.

Presence detection ($\mathbf{Adv}_2$) Linking sessions may also allow the adversary to deduce the fact that a certain user is present at a specific location in the network. This information can be used to obtain context knowledge such as the time of day a user is typically active. Additionally, the adversary may infer the locations from which a roaming user is accessing the Internet, which in turn can be used to track the physical movements of users (using databases that contain the approximate geo-location of IP addresses).

Behavior-based tracking can be employed either by locally confined adversaries or by distributed adversaries. **Locally confined adversaries** run an individual service, which is accessed by users over a longer period of time, e. g., a manually configured third-party DNS resolver or a web proxy server. **Distributed adversaries** have access to the sessions of roaming users at multiple, geographically dispersed locations. An example of such an adversary is a service provider that offers free WiFi in multiple cities.

Modeling the classification problem. In our model each user is represented by a dynamic IP address (or another pseudonym) that is replaced by a different one after a fixed amount of time, i. e., there are epochs $e \in E$ of constant length Δ (e. g., $\Delta = 24$ hours, beginning at 4 a.m. each day).

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. Then, each user session is mapped to an $|H|$-dimensional vector $\vec{x}$, where the i-th component corresponds to the access frequency of the i-th destination host according to a totally ordered list of all observed destinations H. Figure 1 shows a scenario with two epochs, each of which contains two sessions. All requests within a session originate from the same source IP address. Modeling the sessions as

$S_1 = (e_1, s_1, (biz.org, site.com, img.site.com))$
$S_2 = (e_1, s_2, (biz.org, www.org.net, www.org.net))$
$S_3 = (e_2, s_3, (site.com, img.site.com, www.org.net))$
$S_4 = (e_2, s_4, (biz.org, www.org.net, biz.org))$

we can obtain $H = (biz.org, img.site.com, site.com, www.org.net)$ and thus the following attribute vectors: $\vec{x}_1 = (1, 1, 1, 0)$, $\vec{x}_2 = (1, 0, 0, 2)$, $\vec{x}_3 = (0, 1, 1, 1)$ and $\vec{x}_4 = (2, 0, 0, 1)$.

Note that for tracking to be effective these *raw* attribute vectors will be processed later on, which involves removing and adding attributes as well as transforming attribute values (cf. Sects. 5.3 and 6.4).

Based on this model we formulate the behavior-based tracking problem as a classification task (e. g., Han et al., 2011, p. 327). Every user session corresponds to a single *instance* $x \in X$, which is represented by its attribute vector $\vec{x}$. Each instance belongs to a class $c \in C$, which represents a certain user u. A set of *training instances* I_{train}, whose class assignments are known (Sessions 1 and 2 in Fig. 1), is used to build a predictive model. Based on the model, a *classifier* assigns the most probable class to each *test instance* $x \in I_{\text{test}}$, whose class is supposed to be unknown in the experiment (Sessions 3 and 4 in the example).

4. Dataset

We evaluate the feasibility of behavior-based tracking using *DNS queries*, i. e., from the viewpoint of a curious DNS resolver. In cooperation with the computer center of the University of Regensburg, Germany, we recorded the DNS queries received by the two resolvers on the campus network between February 1, 2010 and June 30, 2010. In total we logged 2,645,748,393 queries for 77,662,943 unique hostnames issued from 18,904 unique source IP addresses.

We aimed to protect the privacy of the users as much as possible. Therefore, 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 discarded 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 undermines our pseudonymization efforts. Therefore, we limit access to the logs to a small group of trusted researchers.

Our Datasets. As we strive to obtain meaningful and representative results, the datasets used to conduct the experiments presented in this paper consist only of well-defined parts of our log files. For most experiments we use the **TWO-MONTHS dataset**, which contains the queries between May 1, 2010 and June 30, 2010 (61 days). This choice is motivated by the fact that this period represents the middle of the summer term, i. e., it sports the largest daily query volumes and is affected by almost no user churn. Some experiments turned out to be too time-consuming, though. For them we resorted to the **TWO-DAYS** dataset that contains all queries observed within a busy two-day period (May 3 and May 4, 2010). Note that the absolute accuracy values obtained for

Table 1: Cumulative distributions of descriptives for TWOMONTHS dataset

Descriptive Statistic	p_0	p_{25}	p_{50}	p_{75}	p_{100}
Queries per user	1	24,068	59,367	121,185	28,857,393
Queries per hostname	1	1	2	4	9,781,157
Distinct hostnames per user	1	2,106	4,184	7,433	303,568
Active users per day	1,107	2,100	2,497	3,092	3,218
Queries per user and day	1	569	1,384	2,969	5,144,359
Hostnames per user and day	1	196	372	671	258,110
Number of active days per user	1	31	43	52	61

TWODAYS are not representative for the whole dataset. Furthermore, different results can only be compared if they have been obtained with the same dataset.

Both datasets contain queries originating from the *student housing network segment* only. Thus, at the time of recording our logs most users were enrolled students using the Internet for studying as well as for private browsing. In this network segment all users have never-changing, static IP addresses. When our dataset was recorded each device was assigned a *unique, static* IP address, and each resident was typically only allowed to register a single device. This means that each IP address in our dataset is *supposed* to be exclusively used by one resident. However, multiple persons may use a single machine, and some persons may issue requests using multiple machines, e. g., when students visit their neighbors. As we cannot reliably detect such practices, we have to accept that our dataset will not be perfectly *clean*, i. e., linking the affected sessions will be *more difficult*. Thus, the accuracy obtained on our dataset will slightly underestimate the accuracy achievable on a perfectly clean dataset.

During the evaluation of the tracking techniques we will simulate *dynamically changing IP addresses*, i. e., *a priori* the classifiers (cf. Sect. 5) do not know which sessions belong to a given user. Of course, due to the fact that our users are assigned static IP addresses, we *do* know all the sessions that belong to a given user. Thus, we can compare the predictions of the tracking techniques with the actual set of sessions of each user to measure their accuracy.

Descriptive Statistics. The TWOMONTHS dataset consists of 431,210,371 queries for 5,010,507 different hostnames issued by 3862 users. We present additional descriptive statistics in Table 1. For each statistic we show the minimum and maximum values (p_0 and p_{100}), the first and third quartiles of the cumulative distribution of values (p_{25} and p_{75}) as well as the median value (p_{50}).

A break-down of the queries by query type, as shown in Table 2, suggests that almost all of our "users" are desktop clients used by humans – and not machines running server software. The majority of queries stems from regular name resolutions for IPv4 addresses (A) as well as IPv6 addresses (AAAA), which are mainly caused by users browsing the Internet. Moreover, we observe a substantial amount of reverse lookups (PTR queries). Those are mostly issued by the network browser of the Windows operating system for the purpose

Table 2: Distribution of DNS query types

Type	Queries		Hostnames		Users	
	absolute	%	*absolute*	%	*absolute*	%
Total	431,210,371	100.000	5,010,507	100.000	3,862	100.000
A	236,210,050	54.778	3,668,822	73.223	3,860	99.948
AAAA	149,322,427	34.629	2,633,070	52.551	3,170	82.082
PTR	43,060,608	9.986	815,852	16.283	1,934	50.078
SRV	1,497,622	0.347	322	0.006	2,690	69.653
MX	474,827	0.110	252,953	5.048	45	1.165
ANY	281,023	0.065	7	0.000	1,526	39.513
SOA	226,975	0.053	131	0.003	351	9.089
TXT	115,300	0.027	8,715	0.174	680	17.607
NS	12,028	0.003	346	0.007	35	0.906
TKEY	4,518	0.001	2	0.000	1	0.026
NAPTR	4,281	0.001	10	0.000	14	0.363
SPF	512	0.000	236	0.005	1	0.026
CNAME	196	0.000	190	0.004	9	0.233
AXFR	2	0.000	1	0.000	1	0.026
NULL	2	0.000	1	0.000	1	0.026

of neighborhood discovery. Some versions of Windows issue ANY queries for the purpose of Internet connectivity detection (resolving *dns.msftncsi.com*) and proxy detection (resolving the hostname *wpad*). The SRV queries are mainly created by machines running Apple's Bonjour service discovery protocol. Some students are running their own mail servers on their machines (causing MX queries), some of which employ the Sender Policy Framework (SPF) protocol for the purpose of spam filtering. Finally, there is evidence of DNS servers (issuing SOA, NS, TKEY and CNAME queries) being installed on a very small number of machines. The TXT queries are mainly issued by anti-virus software looking up signatures and the current software version, while the NAPTR queries are caused by VoIP softphones trying to look up SIP gateways.

5. Behavior-Based Tracking Techniques

In this section we will review three promising techniques that can be employed to link the sessions of a user based on observed queries:

1. the 1-Nearest-Neighbor (1NN) classifier using the two distance metrics Jaccard coefficient and cosine similarity (Sect. 5.1),
2. the Multinomial Naïve Bayes (MNB) classifier (Sect. 5.2), and
3. Yang's behavioral pattern mining approach (PM) using profiles based on the two metrics support and lift (Sect. 5.4),

Furthermore, in Sect. 5.3 we will describe a number of data transformations that can be used in conjunction with the 1NN and MNB classifiers. We will evaluate the effectiveness of the techniques in Sect. 6.

An important difference between the techniques relates to feature selection: while 1NN and MNB take into account *all attributes* present in the instance vectors by default, an integral part of the PM techniques consists of extracting a subset of attributes before building user profiles. We will study different *feature selection strategies* for 1NN and MNB in Sect. 6.4.

5.1. 1-Nearest-Neighbor Classifier (1NN)

A straightforward technique for behavior-based tracking is the *1-Nearest-Neighbor classifier* **(1NN classifier)**, a variant of the *kNN classifier* (e. g., Manning et al., 2008, p. 297). Nearest-neighbor classifiers determine class membership by comparing a given test instance to all available training instances from the previous day. The test instance is assigned to the class of its closest neighbor from the training instances. While kNN classifiers base their prediction on a majority vote of the k nearest neighbors, 1NN considers the closest instance only. Choosing 1NN is motivated by the fact that the adversary has only a single training instance per user at his disposal in our scenario.

The choice of a suitable distance (or similarity) measure is crucial for the performance of 1NN. Similarity measures like Euclidean distance or the Pearson correlation coefficient do not work well for our problem, because our attribute vectors are very sparse (cf. Sect. 4): The mere fact that two instances have many 0 values in common, does not necessarily imply that they must be very similar (Han et al., 2011, p. 77). Therefore, we apply two related distance measures that focus on the attributes that two instances *do* have in common: the Jaccard coefficient and cosine similarity.

The **Jaccard coefficient** is a [0;1]-normalized similarity measure that expresses the fraction of non-zero attributes that two instances do have in common (e. g., Han et al., 2011, p. 71). It operates on the set representation of instances, which contains the hostnames h that have been accessed in the session of a user, e. g., $X = \{h_1, h_3, h_7\}$ and $Y = \{h_2, h_3\}$. The Jaccard coefficient is given by

$$s_{X,Y} = \frac{|X \cap Y|}{|X \cup Y|}$$

For the example sessions above we obtain $s_{X,Y} = \frac{1}{4} = 0.25$. Note that access frequencies are neglected during similarity computation with the Jaccard coefficient. Calculating the Jaccard coefficient is equivalent to calculating the cosine similarity of two binary attribute vectors that do not contain absolute access frequencies but a 1 value for each hostname that was accessed in a session.

Cosine similarity is a [0;1]-normalized similarity measure between two instance vectors $\vec{x}$ and $\vec{y}$ that reflects the angle θ which is spanned by them (Han et al., 2011, p. 77). A smaller θ corresponds to a larger value of the cosine similarity. The similarity s is equal to the quotient of the dot product of $\vec{x}$ and $\vec{y}$ and the product of their magnitudes:

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

In comparison to the Jaccard coefficient, cosine similarity utilizes more information from the instances, because access frequencies are taken into account.

We will refer to the two 1NN classifiers with the labels **1NN-JACCARD** and **1NN-COSIM** henceforth.

5.2. Multinomial Naïve Bayes Classifier (MNB)

In contrast to the 1NN classifiers, behavior-based tracking with the *Multinomial Naïve Bayes classifier* **(MNB classifier)** involves an additional training stage, in which a statistical model is learned for each class (e. g., Manning et al., 2008, p. 258). Based on this model the probability that a test instance x belongs to a certain class c_i is determined by computing the following conditional probabilities for all classes available in the training set:

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

The classifier assigns the test instance x to the class, which scores 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.

The MNB classifier is typically used for the purpose of text categorization, e. g., for the detection of unsolicited bulk e-mails. Choosing this classifier for behavior-based tracking is motivated by the fact that the features used in our model (access frequencies of hostnames) and the features used for text categorization (term frequencies) share an important characteristic: both of them are subject to power law distributions (Zipf, 1968; Adamic and Huberman, 2002). 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) (Cortes and Vapnik, 1995). Yang reports that SVMs increased classification runtimes by 300 %, while classification accuracy did not improve considerably (Yang, 2010).

5.3. Transformation of Frequency Vectors

The text categorization discipline makes use of numerous transformation techniques in order to boost classification accuracy. We implemented the most relevant techniques, which can be used in conjunction with the 1NN and MNB classifiers.

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 (Witten and Frank, 2005, p. 311). Additionally, the frequency vectors of all classes are normalized to a uniform Euclidean length (Witten and

Frank, 2005, p. 310). The application of the two transformations is indicated by the use of the label **TFN** in this paper. We indicate the absence of this transformation by **RAW**.

Secondly, the frequencies $f_{h,x}$ can be multiplied by the *Inverse Document Frequency* $idf_h = \log \frac{N}{n_h}$ before the vectors are normalized (Witten and Frank, 2005, p. 311). Here, N is the number of all instances and n_h the number of all instances containing the hostname h. Transforming the vectors with the *IDF* reduces the weight of common hostnames that are accessed by a large number of users. The combination of the IDF and TFN transformation is designated by **TFIDFN**.

Finally, *n-grams* are known to increase the accuracy in many text mining problems (Beesley, 1988; Cavnar and Trenkle, 1994; Damashek, 1995). They have also been used for traffic analysis problems with success (Rieck and Laskov, 2007). This technique obtains composite hostnames by concatenating the hostnames of n adjacent queries. It operates on the premise that multiple chronologically adjacent queries are issued when a website is visited. Thus, hostnames that are rather meaningless on their own (*ad.tracker.com*, for instance) become enriched with context information (e. g., the hostname of the website the user was visiting at that time). We denote the use of n-grams by appending a suffix to the configuration name. While the suffix **1** indicates that only the original hostnames are used, **2** indicates that only bi-grams are used. The suffix **1+2** indicates that – in addition to the original hostnames – bi-grams are added to the instances: If a user issues queries for the hostnames *www.biz.org*, *img.biz.org*, *ad.tracker.com* in this order, the resulting 1+2-instance will contain access frequencies for the following attributes: {*ad.tracker.com*, *img.biz.org*, [*img.biz.org, ad.tracker.com*], *www.biz.org*, [*www.biz.org*, *img.biz.org*]}.

5.4. Mining Characteristic Behavioral Patterns (PM)

We also include Yang's pattern mining (PM) techniques in our evaluation (Yang, 2010). These techniques are based on two concepts rooted in the problem domain of association rule mining: finding frequent item sets and generating strong association rules from them (cf. Han et al., 2011, p. 247). In principle, a characteristic pattern can consist of multiple items, i. e., multiple web queries of a user. For simplicity, Yang does not make use of composite patterns. Instead, a pattern consists of a single "web site" visited by a user within a session, and each user profile consists of a set of such patterns. We follow Yang's proposal and use individual hostnames as patterns.

In the following we briefly review the central aspects of the pattern mining techniques: extracting top patterns, constructing user profiles, and matching sessions based on these profiles.

Extracting Top Patterns. As in the case of the MNB classifier, using pattern mining for behavior-based tracking involves a dedicated training stage before user sessions are linked in the matching stage. In the training stage the set of patterns to be operated on during the matching stage is constructed. To this end we determine the n_{patterns} most significant patterns (hostnames) for each

user u by ranking the hostnames observed within I^u_{train} (the set of training instances available for u) according to their aggregated access frequencies within I^u_{train}. The top n_{patterns} hostnames are picked from each user and added to the set of *candidate patterns* P_{all}. In the matching stage only accesses to hostnames contained in P_{all}, i. e., the union of the top n_{patterns} hostnames of all the users that were present during training, are considered. This feature selection step reduces the size of the training set considerably. Yang performed her experiments with $n_{\text{patterns}} = 10$ and she reports that higher values of n_{patterns} did not contribute to significantly higher accuracy.

Profile Construction. A trained user profile R_u contains the hostnames of the intersection of P_{all} and the hostnames visited within the sessions presented during training by the user. For each hostname in R_u its significance in the profile is computed. Yang constructs two significance measures: support and lift. The **support** s^u_h of a hostname h captures the within-user strength of h for user u, i. e., the fraction of training sessions from I^u_{train} that contains requests to h. Note that support values will be either 0 or 1 in our real-world evaluation in Sect. 6.2, because in this scenario there is only a single training instance available per user.

The second measure considered by Yang is the **lift** l^u_h of a hostname, which is based on its support s^u_h, but takes into account whether other users also have accessed h in their training sessions. To this end the value of the within-user strength is adjusted by the *overall pattern strength* o_h to obtain

$$l^u_h = \frac{s^u_h}{o_h} \text{ with } o_h = \frac{|I^h_{\text{train}}|}{|I_{\text{train}}|}$$

The overall pattern strength o_h is obtained by computing the fraction of the training sessions of all users that contain h (denoted by I^h_{train}) and the total number of available training sessions of all users.

Note that our method for the extraction of the top n_{patterns} hostnames presented above slightly deviates from Yang's approach, who selects the n_{patterns} most significant patterns for each user (Yang calls this parameter X) based on their *support* value s^u_h. However, using Yang's approach would result in meaningless pattern selection in our behavior-based tracking scenario: As there is only a single training instance per user available in our scenario, all hostnames h present in the session of a user would be assigned the same support value $s^u_h = 1$, which makes it impossible to rank the hostnames. On the other hand, our approach is able to select the most significant hostnames based on a single session per user.

Matching Stage. The test instances with unknown class assignments are processed in the same way as the training instances during profile construction to obtain support and lift values. In the matching stage there is one feature vector (containing support or lift values for each hostname from P_{all}) for every user from the training set as well as one vector for the anonymous test instance. Matching is performed by pair-wise computation of the similarity between the

test vector and all the training vectors. The class of the test instance is predicted by finding the training instance with the highest similarity score. We implement the Euclidean distance for this purpose, as proposed by Yang in her design (cf. Yang, 2010).

We will refer to the two pattern mining classifiers with the labels **PM-SUPPORT** and **PM-LIFT**.

6. Evaluation

Due to the large size of the dataset we did not use off-the-shelf data mining tools such as *Weka* (Hall et al., 2009) for our experiments. Instead, based on the source code of Weka we ported the 1NN and MNB classifiers to Apache Hadoop, a *MapReduce* framework implemented in Java (Dean and Ghemawat, 2008; White, 2011). We also implemented the pattern mining techniques described in Sect. 5.4. We tuned the implementation of the classifiers for high memory efficiency, so that they can process the large instance vectors, which result when all features are considered. We ran the experiments on a cluster of 18 quad-core desktop machines (Intel Core i5, 3.1 GHz, 8 GB RAM, 1 TB HDD). In the following we will report our evaluation methodology and the most significant results.

The evaluation proceeds in two stages: At first, we present results obtained from **cross validation** in Sect. 6.1. These experiments are carried out with a specially crafted, homogeneous dataset, so that we can appreciate the effectiveness of the tracking techniques in a controlled setting. After that, we will apply the techniques to the TWOMONTHS dataset in a day-to-day manner. This allows us to assess how well the techniques cope with **actual traffic in the real world** (Sect. 6.2). Scrutinizing the results we discovered that the classifiers particularly struggle with *user fluctuation*. We elaborate on this issue in Section 6.3, and we also present our ideas for optimization. We are also interested in reducing the complexity of behavior-based tracking via feature selection. We will review three different approaches in Sect. 6.4.

6.1. Cross Validation

Firstly, we analyze the effectiveness of the techniques with a series of *cross validation* experiments in a *controlled setting*. Cross validation ensures that the results obtained are not biased due to selection of peculiar training instances (Kohavi, 1995). 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 randomly selected 3000 users $u \in U$ from the TWOMONTHS dataset, who were able to contribute at least 20 instances of 24 hours length. 20 instances were randomly selected from each user (class), resulting in a cross validation dataset D of 60,000 instances in total. The choice of 20 instances per user is the result of looking for a good trade-off between maximizing the *number of instances per user* (in order to observe the potentially beneficial effects of a large number of training instances) and at the

same time maximizing the *number of users contributing to the cross validation dataset* (in order to make the problem more challenging).

We perform *10-fold* stratified cross validation, which means that the whole dataset is partitioned into 10 equal-sized sets, the "folds". Each fold contains the same number of instances from each class. Therefore, we split the 20 instances of each class randomly, yet equally, into 10 disjoint sets: $D = D_1 \cup D_2 \cup \ldots \cup D_{10}$. Given 20 instances per user, each set contains 2 instances of each user. The actual evaluation takes place in 10 *runs* $k \in 1, \ldots, 10$. In the k-th run the instances $D_{\text{train}} = D \setminus D_k$ are used to train the classifier. D_{train} contains $\frac{9}{10}$ of all instances of every user u_i. With $I^{u_i}_{\text{train}}$ 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 predictive accuracy is then obtained as the average of the 10 runs.

For each experiment we compute the average *precision* and *recall* values, two metrics, which are typically used to analyze the predictive 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, an adversary that tries to track a user as extensively as possible over a period of time is primarily interested in high recall values. On the other hand, if the primary objective is to minimize the number of false mappings, a high precision value is desired.

As a baseline we start out with the classical 10-fold cross validation approach; after that we will adapt it to operate with fewer training instances. The classical approach provides the classifier with 9 out of 10 folds for training and evaluates it on the remaining fold in each of the 10 runs. Given our dataset with 20 instances per class, we have $|I^{u_i}_{\text{train}}| = 18$ training and 2 test instances per user, respectively. Table 3 lists the results observed with the pattern mining techniques for varying numbers of top patterns n_{patterns}. PM-SUPPORT outperforms PM-LIFT, which is in line with (Yang, 2010). However, in contrast to her findings, we observe a significant impact of n_{patterns}. Table 4 shows the results for different configurations of the 1NN and MNB classifiers. For conciseness we provide only results for 1-grams and 2-grams. Higher-order n-grams ($n > 2$) did have no more beneficial effects on precision and recall in our experiments.

Results for Less Training Instances. The high accuracy obtained in the previous experiment is of little relevance for behavior-based tracking in practice, because we do not expect the adversary to have $|I^{u_i}_{\text{train}}| = 18$ training instances for each user at his disposal. Hence, we repeated the cross validation experiments with smaller numbers of training instances using the configurations that offered the best precision and recall for $|I^{u_i}_{\text{train}}| = 18$ (printed in bold in Tables 3 and 4). Figure 2 illustrates the decay of the recall values as observed during these experiments. With the decreasing number of training instances it

Table 3: Cross validation results for pattern mining techniques with $|I^{u_i}_{\text{train}}| = 18$

n_{patterns}	PM-SUPPORT		PM-LIFT	
	precision	*recall*	*precision*	*recall*
1	0.814	0.432	0.345	0.221
5	0.899	0.464	0.503	0.298
10	0.910	0.468	0.575	0.331
20	**0.915**	**0.470**	0.628	0.353
30	0.911	0.468	0.657	0.364
40	0.910	0.468	0.669	0.368
50	0.910	0.468	**0.674**	**0.369**
60	0.909	0.467	0.664	0.363

Table 4: Cross validation results for 1NN and MNB classifiers with $|I^{u_i}_{\text{train}}| = 18$

Configuration	1NN-JACCARD[a]		1NN-COSIM		MNB	
	precision	*recall*	*precision*	*recall*	*precision*	*recall*
RAW-1	0.832	0.808	0.806	0.780	0.098	0.078
RAWIDF-1	–	–	0.795	0.768	0.098	0.078
RAW-2	0.843	0.818	0.750	0.716	0.304	0.246
RAW-1+2	**0.862**	**0.837**	0.839	0.817	0.139	0.109
TFN-1	–	–	0.881	0.859	0.856	0.836
TFIDFN-1	–	–	0.860	0.834	0.891	0.859
TFN-2	–	–	0.839	0.805	0.861	0.832
TFN-1+2	–	–	**0.903**	**0.882**	0.883	0.861
TFIDFN-2	–	–	0.871	0.851	0.908	0.885
TFIDFN-1+2	–	–	0.892	0.871	**0.916**	**0.892**

[a]1NN-JACCARD is unaware of access frequencies; TFN and IDF cannot be applied.

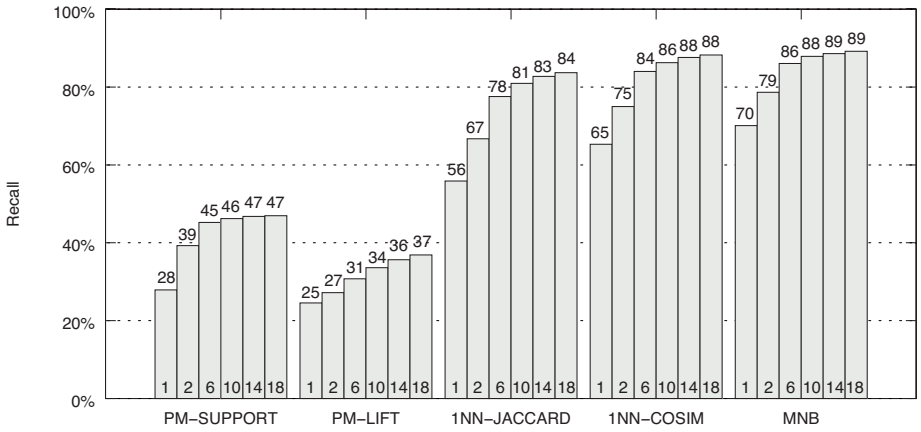


Figure 2: Observed recall values for varying number of training instances using the best configurations from Tables 3 and 4

Table 5: Cross validation results for pattern mining techniques with $|I^{u_i}_{\text{train}}| = 1$

n_{patterns}	PM-SUPPORT		PM-LIFT	
	precision	*recall*	*precision*	*recall*
1	0.448	0.268	0.234	0.147
5	**0.527**	**0.303**	0.347	0.212
10	0.516	0.294	0.383	0.232
20	0.486	0.278	0.403	0.243
30	0.481	0.273	**0.418**	**0.250**
40	0.455	0.259	0.412	0.247
50	0.439	0.252	0.416	0.249
60	0.439	0.251	0.407	0.244

becomes increasingly difficult for the classifier to assign the test instances to the correct class. For instance, the MNB (TFIDFN-1+2) classifier achieved a recall of 89.2 % with $|I^{u_i}_{\text{train}}| = 18$, while for $|I^{u_i}_{\text{train}}| = 1$ its recall decreases to 70.1 % (precision to 71.8 %). Additionally, we re-evaluated all the configurations shown in Tables 3 and 4 with $|I^{u_i}_{\text{train}}| = 1$. The fact that the best results are now achieved for different values of n_{patterns} (cf. Table 5) suggests that the *pattern mining techniques* are quite sensitive to this change. On the other hand, the ranking of the configurations for the *classifiers* remains mostly unchanged (cf. Table 6). A prominent exception is the 1NN-COSIM classifier: In the end the TFIDFN-1+2 configuration outperforms the TFN-1+2 configuration in case of $|I^{u_i}_{\text{train}}| = 1$.

Table 6: Cross validation results for 1NN and MNB classifiers with $|I^{u_i}_{\text{train}}| = 1$

Configuration	1NN-JACCARD[a]		1NN-COSIM		MNB	
	precision	*recall*	*precision*	*recall*	*precision*	*recall*
RAW-1	0.504	0.485	0.461	0.434	0.042	0.036
RAWIDF-1	–	–	0.458	0.437	0.020	0.017
RAW-2	0.577	0.548	0.456	0.424	0.088	0.072
RAW-1+2	**0.583**	**0.559**	0.523	0.507	0.043	0.036
TFN-1	–	–	0.603	0.591	0.603	0.591
TFIDFN-1	–	–	0.636	0.609	0.636	0.612
TFN-2	–	–	0.595	0.566	0.595	0.569
TFN-1+2	–	–	0.670	0.653	0.661	0.652
TFIDFN-2	–	–	0.675	0.662	0.681	0.671
TFIDFN-1+2	–	–	**0.709**	**0.686**	**0.718**	**0.701**

[a]1NN-JACCARD is unaware of access frequencies; TFN and IDF cannot be applied.

6.2. Evaluation in Real-World Setting

We proceed with experiments that study the feasibility of behavior-based tracking in the real-world setting. To this end, we have implemented a fully-automated experimental environment with the Hadoop framework that allows us to simulate an adversary that tries to track all users on a day-to-day basis, i. e., the *epochs* (cf. Sect. 3) start at midnight and last 24 hours. For now, we focus on linking sessions of two *consecutive days*. Therefore, in our experiment we iterate 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 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, i. e., $t+1$. Later on, at the end of Sect. 6.3, we will also study the effect of increasing the *offset* δ between training and test i. e., selecting test instances from $t + \delta$ with $\delta > 1$.

In the case that user u_i is active on two consecutive days, the simulated observer scores a *correct mapping* (C_1) 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). Another type of *non-detectable error* (E_2) occurs when the classifier does not assign any instance to c_i, although u_i was active on day $t + 1$; this means that the adversary loses track of the user. We record a *detectable error* (E_3) 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* (C_2) 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 $(|C_1|+|C_2|)/(|C_1|+|C_2|+|E_1|+|E_2|+|E_3|)$.

Table 7: Real-world results (accuracy) for PM classifiers

n_{patterns}	PM-SUPPORT	PM-LIFT
5	**0.538**	0.375
10	0.524	0.385
15	0.510	**0.387**
20	0.498	0.384
30	0.478	0.377
50	0.456	0.378

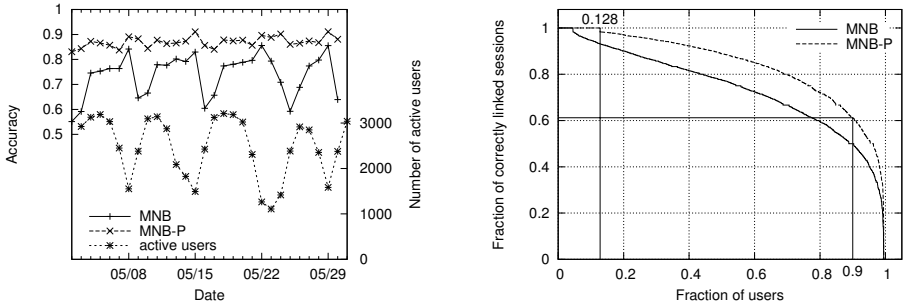
Table 8: Real-world results for 1NN and MNB classifiers

Configuration	1NN-JACCARD[a] *accuracy*	1NN-COSIM *accuracy*	MNB *accuracy*
RAW-1	0.562	0.547	0.366
RAWIDF-1	–	0.547	0.360
RAW-2	**0.659**	0.570	0.485
RAW-1+2	0.640	0.612	0.447
TFN-1	–	0.668	0.672
TFIDFN-1	–	0.672	0.673
TFN-2	–	0.690	0.692
TFN-1+2	–	0.723	0.722
TFIDFN-2	–	0.732	0.731
TFIDFN-1+2	–	**0.744**	**0.744**

[a]1NN-JACCARD is unaware of access frequencies; TFN and IDF cannot be applied.

We evaluate the tracking techniques with this scheme using the TWO-MONTHS dataset with the configurations already used for cross validation (cf. Sect. 6.1). As we want to report results that are significant for typical users, in each epoch we ranked the users according to their query volume and removed the instances of the top 5 % and bottom 5 % of the list for all the experiments in the remainder of the paper. We observed that accuracy decreases by about 2–3 percentage points, if all users are considered. This decrease is mainly due to users with extremely little activity that the classifiers tend to lose track of.

Tables 7 and 8 show the results in the real-world setting. Interestingly, the behavior of the pattern mining (PM) techniques diverges from the results observed during cross validation: PM-SUPPORT achieves its highest accuracy value (53.8 %) with $n_{\text{patterns}} = 5$, while PM-LIFT achieves an accuracy of up to 38.7 % using $n_{\text{patterns}} = 15$ (cf. Table 7). The results of the remaining classifiers are mostly in line with cross validation for $|I^{u_i}_{\text{train}}| = 1$ (cf. Table 8): A notable exception is 1NN-JACCARD, which achieves its highest accuracy (65.9 %) for



(a) Fluctuation of users

(b) Cumulative distribution of accuracy

Figure 3: Real-world tracking with MNB and MNB-P classifiers

RAW-2. 1NN and MNB both achieve 74.4 % using TFIDFN-1+2.

6.3. Dealing with Fluctuating Activity

The real-world setting is more challenging than the controlled setting because of 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. As the adversary does not have access to any context information in our model, the classifiers will still 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. When the students first become active again accuracy drops to about 60 %, while later in the week it reaches more than 80 % (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 13.4 % of all cases (52.3 % of all errors), and for 88.6 % of the ambiguous mappings the correct test instance was in fact part of the set of assigned test instances.

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, the adversary can try to resolve this ambiguous prediction, i. e., to single out the instance that fits best.

To this end we implemented the option to **prune superfluous predictions** with the help of the 1NN-COSIM classifier. 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. Only the test instance with the highest similarity value is assigned to the class in the end; all other instances are discarded. We denote this post-processing step by appending the suffix **P** to the classifier label.

The effect of this optimization is shown in the graph labeled *MNB-P* in

Fig. 3a. Overall the accuracy increases from 74.4 % to **85.4 %** (MNB-P, TFIDFN-1+2) and **85.3 %** (1NN-COSIM-P, TFIDFN-1+2), respectively.

Although the overall values suggest that behavior-based tracking is feasible in general, we remark that the technique is not effective for all users with the same high accuracy. Figure 3b shows the accuracy levels that can be obtained for a certain fraction of users with the TFIDFN-1+2 configuration and the MNB-P classifier. For 12.8 % of the users the classifier establishes a correct mapping for *all their sessions*, and for 90 % of the users it links at least 61.2 % of their sessions correctly. We observed similar values for the 1NN-COSIM-P classifier.

Epochs starting at other times of day. In our experiments each epoch starts at 12 a.m. midnight. We analyzed whether our results are biased due to this choice. To this end we set up additional experiments with the MNB-P classifier (TFIDFN-1+2, TWOMONTHS dataset) with epochs starting every three hours from 12 a.m. until 9 p.m. According to the results the time of day *does have* some influence: Accuracy values obtained during these experiments varied between 85.4 % (for epochs starting at 12 a.m.) and 86.2 % (for epochs starting at 6 p.m.). This result can be explained by varying usage intensities throughout the day. In the evening there are more active users than during the night. Thus, in case of epochs starting at 6 p.m. it is more likely that users issue related queries that span two epochs (see also Sect. 7.1).

Increasing the offset between training and test. We have focused on linking the sessions of consecutive days so far, i. e., $\delta = 1$. In practice, an adversary may not be able or not be willing to track users on a day-to-day basis. Therefore, we studied the effect of increasing the offset δ. Figure 4 shows a stacked breakdown of the results for the MNB-P classifier (TFIDFN-1+2 configuration, TWOMONTHS dataset) for values of δ between 1 day and 28 days (4 weeks). The breakdown provides insight into the distribution of all the predictions of the classifier over all days and users. We differentiate between four prediction cases as defined in Sect. 6.2. The bottom half of the figure consists of *correct predictions* (C_1 and C_2). The majority of correct predictions consists of correctly mapped sessions (C_1). In the remaining cases the classifier correctly predicted that there was no instance of a given user in the test set (C_2). The sum $|C_1| + |C_2|$ equals the value of our *accuracy metric*. The top half of the figure consists of *errors* (E_1 and E_2): Firstly, there are incorrect re-identifications (E_1), i. e., mapping a test instance to the wrong class. In the remaining cases the classifier erroneously predicted that there was no instance of a given user in the test set, although, in fact, there is (E_2). Note that all errors are *non-detectable* due to the pruning step of the MNB-P configuration, i. e., there are no E_3 errors. As shown in Fig. 4 accuracy decays moderately with increasing values of δ, ranging from 85.4 % ($\delta = 1$) to 75.6 % ($\delta = 28$). The breakdown illustrates that correct re-identification (C_1) becomes more difficult for greater offsets, while the share of the C_2 cases remains quite stable. According to these results behavior-based tracking remains feasible even if user profiles are not updated daily.

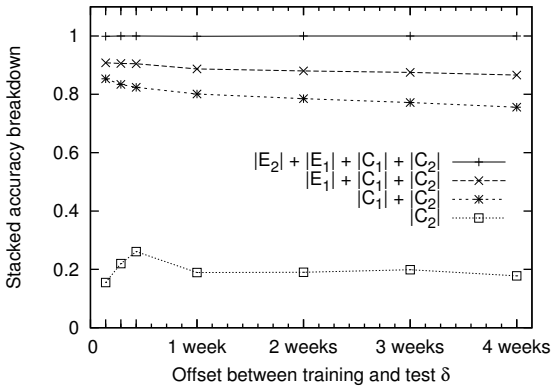


Figure 4: Stacked breakdown of correct (C_1 and C_2) and erroneous (E_1 and E_2) predictions using MNB-P (TFIDFN-1+2) for different offsets δ between training and test

6.4. Feature Selection Strategies

In this section we will analyze the effects of three different feature selection strategies that can be used in conjunction with the 1NN and MNB classifiers. Feature selection serves two purposes: Firstly, it can be used to improve accuracy by presenting only the most expressive and relevant features to the classifier, while dropping noisy features. Secondly, it is a means to reduce the high dimensionality of the attribute space and thus computation complexity. Apart from its potential benefits, feature selection also poses the risk of removing hostnames that would in fact be useful for tracking, which may in turn degrade accuracy. While PM-SUPPORT and PM-LIFT proposed by Yang (2010) already include a dedicated feature selection step before building profiles, we applied the 1NN and MNB classifiers to the *whole instance vectors* so far. On average these whole instance vectors contain 1250 non-zero attributes, and the total number of distinct attributes is 27,333,593 (including 2-grams; without the 5 % most and least active users, cf. Sect. 6.2). For conciseness, we will limit our discussion to the results obtained for the MNB-P classifier (TFIDFN-1+2 configuration, TWOMONTHS dataset).

The first strategy selects the **top k attributes per instance**. We evaluate this strategy to analyze how many attributes in an instance are actually necessary to establish correct mappings. At first sight this resembles the feature selection step in PM-SUPPORT and PM-LIFT (cf. Sect. 5.4). However, it differs in an important aspect: The two pattern mining techniques operate on the top n_{patterns} *hostnames* obtained from the raw instance vectors. In contrast, we select the top k attributes based on the attribute values resulting *after the application of the vector transformations*. Due to the application of the IDF and inclusion of 2-grams the top k attributes are very different from the top n_{patterns} hostnames. To obtain the top k attributes we rank all attributes within an in-

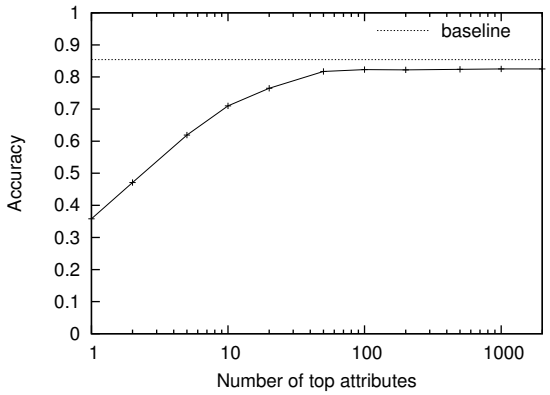


Figure 5: Feature selection: top k attributes (MNB-P TFIDFN-1+2)

stance according to their attribute values and keep only the k attributes with the highest values; all remaining attributes are set to 0. In effect, the classifier will only operate on the union of all top k attributes of all the users that were present during training.

Figure 5 illustrates the effectiveness of this strategy. Without feature selection this configuration achieves an accuracy of 85.4 % (baseline). Using only the $k = 5$ most significant attributes, we still observe an accuracy of 61.9 % (as opposed to 53.8 % for PM-SUPPORT with $k = 5$). The accuracy levels off at $k = 100$ with a value of 82.3 %. Even for large values of k, such as $k = 10{,}000$ the accuracy does not increase any further. As the classifier does not reach the accuracy obtained using all attributes (85.4 %), we can conclude that the classifier needs the less significant attributes in order to correctly link a small fraction of the sessions in our dataset. On the other hand, this feature selection strategy is able to reduce the size of instances considerably: for $k = 100$ instance vectors contain only 615 attributes on average and the total number of attributes is decreased to 4,424,818.

The **overall hostname popularity** feature selection strategy is based on a total order of all observed hostnames according to the number of accesses they attracted from all users. Table 9 shows the accuracy that the MNB-P classifier achieves for two cases. The results for "only top n" are obtained if only the most popular hostnames were to be considered by the adversary (dropping all queries for the remaining hostnames from the dataset). Given this reduced set of hostnames the adversary builds n-grams as usual, i. e., the effective number of attributes will increase beyond the value of n. For small values of n the classifier fails in most cases, but accuracy increases steadily. At $n = 1000$ accuracy passes 80 %, leveling off at 85 %. Interestingly, for $n = 100{,}000$ we observe an accuracy of 85.6 %, which is slightly above the value of 85.4 % observed with all attributes (cf. Sect. 6.3). This result indicates that even such a simplistic feature selection strategy may succeed in filtering noisy attributes that confuse

Table 9: Feature selection: overall hostname popularity (MNB-P TFIDFN-1+2)

top n	10	50	100	500	1,000	5,000	10,000	50,000	100,000
only	0.152	0.407	0.578	0.774	0.806	0.817	0.829	0.851	0.856
skip	0.851	0.845	0.838	0.803	0.784	0.745	0.713	0.610	0.544

the classifier. Like with the previous feature selection strategy, the size of the instance vectors is decreased considerably: for $n = 1000$ the average number of attributes (including 1-grams and 2-grams) is 525 (maximum: 2054), for $n = 10{,}000$ it is 891 (maximum: 4295).

On the other hand the "skip top n" case illustrates the effect of *removing* the queries for the popular hostnames from the dataset, only retaining the long tail of infrequently accessed hostnames. This case refers to adversaries that only log queries for *less popular* hostnames (e. g., to save space used for data retention). Accuracy decreases when n is increased. As expected the size of the dataset can be reduced considerably with this strategy: for $n = 1000$ 51.2 % of all queries are skipped, and 93.2 % for $n = 100{,}000$.

Finally, we consider the strategy of dropping infrequent attributes based on the absolute frequency they occur within an instance. Such a **minimum attribute occurrence** m feature selection strategy is based on the observation that hostnames that attract only one or two queries within a session will likely never be encountered again anyway. Thus, removing them should have little effects on accuracy. In fact, dropping infrequently accessed hostnames promises a considerable reduction of the feature space: for about 38 % of all 5,010,507 hostnames there is only a single request in the TWOMONTHS dataset, and for 28 % there are only two requests. However, our results indicate that minimum attribute occurrence is not a suitable feature selection strategy: For $m = 2$, i. e., dropping all hostnames and 2-grams from a session that occur only once in it, accuracy already decreases considerably from 85.4 % to 77.9 %; for $m = 5$ it decreases to 61.4 %.

6.5. Discussion of Results

According to the cross validation results, the techniques appear to be suitable for building user profiles based on DNS queries. As expected, the 1NN and MNB classifiers profit from all applied feature transformations. In contrast, the PM techniques have to be carefully tuned to fit the given traffic profiles (n_{patterns}). Nevertheless they fall behind on our dataset – offering comparatively low recalls, even when 18 training instances per user are available. As these techniques performed quite well in Yang's study, we attribute this finding to the considerably larger number of concurrent users in our experiment.

The results obtained with $|I^{u_i}_{\text{train}}| = 1$ suggest that the behavior of most Internet users exhibits sufficiently characteristic attributes and that these characteristic attributes are present in a large share of their sessions. Nevertheless, as

these results have been obtained in a controlled setting, one cannot draw conclusions about the (in-)effectiveness of a specific behavior-based tracking technique in practice. In particular, we have seen that the PM techniques behave very differently in the real-world setting and have to be re-tuned. Moreover, the low recall values underestimate the actually attainable accuracy in the real-world setting, which is a result of user fluctuation.

MNB-P and 1NN-COSIM-P achieve a very similar accuracy with the TFIDFN-1+2 configuration (85.4 % and 85.3 %, respectively). This can be explained by the fact that MNB and 1NN-COSIM already perform equally well for looking up the "candidate" class labels, and the final decision in the pruning step employing cosine similarity in both cases (cf. Sect. 6.3). Even though MNB-P performs marginally better in the end, an adversary might actually prefer to use 1NN-COSIM, which achieves its high accuracy more efficiently: with 1NN there is no need to train a predictive model, which reduces runtime and storage required for tracking.

7. Countermeasures

In this section we discuss various countermeasures that potentially mitigate the effectiveness of behavior-based tracking. We will start out with two generic countermeasures (Sects. 7.1 and 7.2). While they can be used with any protocols in principle, for the sake of conciseness we describe them in the context of our DNS scenario. After that we will mention three techniques that are of special interest to prohibit tracking efforts by DNS resolvers (Sect. 7.3).

We do not consider generic anonymization tools like Tor or AN.ON in this paper (Dingledine et al., 2004; Berthold et al., 2001). These systems provide privacy at the cost of performance by encrypting and relaying traffic over multiple routers. To be sure, analyzing the feasibility of behavior-based tracking in the context of anonymizers and studying the privacy and performance implications of using such systems to protect DNS traffic are interesting areas of future work, but they are beyond the scope of this paper.

7.1. Shorter Sessions

Motivated by the reconnection interval of residential DSL plans (cf. Sect. 1) the duration of epochs was fixed to $\Delta = 24\,\text{h}$ in all previous experiments. Now we will study the effect of **changing IP addresses more frequently**. Therefore, we pretend that all users change their IP addresses multiple times per day in a synchronized manner. According to the results this strategy proves to be quite effective. Using the MNB-P classifier (TFIDFN-1+2, TWOMONTHS dataset) accuracy drops from 85.4 % ($\Delta = 24\,\text{h}$) to 64.7 % for $\Delta = 3\,\text{h}$, decreasing further to 53.9 % for $\Delta = 1\,\text{h}$. Figure 6 shows this effect.

Figure 6 also contains results for $\Delta < 1\,\text{h}$ obtained with the TWODAYS dataset. Accuracy decreases from 51.3 % ($\Delta = 1\,\text{h}$) to 42.1 % for $\Delta = 30\,\text{min}$ and to 30.4 % for $\Delta = 5\,\text{min}$. For very short sessions with $\Delta = 1\,\text{min}$ accuracy increases again to 34.0 %. This interesting result can be explained by three

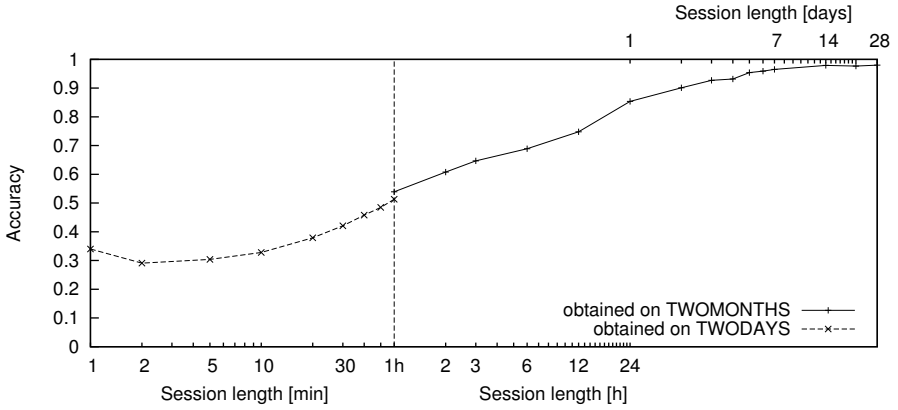


Figure 6: Influence of session length on accuracy

effects: Firstly, for such short epochs it becomes more likely that queries issued by browsing activities of a user span two adjacent epochs, which is beneficial for linking sessions; secondly, decreasing Δ also reduces the number of concurrently active users within an epoch, simplifying the classification problem. Finally, with decreasing Δ the fraction $|C_2|/(|C_1|+|C_2|)$, i. e., correct decisions that refer to the fact that a specific user was *inactive* on the test day, increases: For $\Delta = 1\,\text{min}$ this ratio is 75 % – in contrast, for $\Delta = 24\,\text{h}$ it is only 18 %. Therefore, accuracy values have to be interpreted with care as they tend to overestimate the feasibility of tracking for small values of Δ.

We also studied linkability for $\Delta > 24\,\text{h}$. As expected tracking becomes easier, if users change their IP address less frequently: For sessions that span two days the classifier achieves an accuracy of 90.1 %, which increases to 96.5 % for 7 days. If addresses are changed after four weeks (28 days), the classifier is able to correctly link up to 98.0 % of the sessions.

7.2. Range Queries

We also evaluated **range queries**, a technique that has been originally proposed to protect the privacy of DNS queries (Castillo-Perez and García-Alfaro, 2009; Zhao et al., 2010; Lu and Tsudik, 2010), but can also be adapted for other protocol layers. It is inspired by the principle of Private Information Retrieval (Chor et al., 1998; Kushilevitz and Ostrovsky, 1997) and aims to achieve privacy by obfuscating each *real query* of a user by adding n fake queries for dummy hostnames. The client queries the DNS resolver for each of the $n+1$ hostnames and receives $n+1$ replies from the server, discarding all but the desired one. Obviously, range queries will only be effective, if the adversary cannot easily differentiate between real queries and dummy queries. Choosing appropriate dummy hostnames is a challenging problem in its own regard (cf. Balsa et al., 2012), which is beyond the scope of this paper. Our basic selection strategy is described below.

We want to study to what degree range queries can prevent behavior-based tracking. Therefore, we implemented a range query engine, which adds dummy queries to the dataset. The hostnames available for dummy generation are obtained as follows:

1. We construct the set of all possible dummy hostnames D, which contains the one million hostnames that attracted the highest number of queries in our dataset.
2. From D we draw a random subset, the *dummy pool*, $P \subset D$ with size $|P|$. During each experiment all users use the same dummy pool.

This construction ensures that the set of dummy hostnames overlaps with the set of real queries within the majority of user sessions. Thus, dummy queries will not only introduce additional attributes into the instance vectors, but they will also obfuscate the access frequencies of real queries (referred to as *collisions* below).

Based on previous work we consider two distinct dummy generation strategies: While Castillo-Perez and García-Alfaro (2009) and Zhao et al. (2010) propose to draw dummies randomly and independently for each real query, the construction by Lu and Tsudik (2010) results in static dummies. Generating **static dummies** means that a given real query for some hostname h will always be accompanied by the same, yet randomly chosen dummy requests. We use a pseudo-random number generator, whose seed depends on the real hostname for this case. In case of **random dummies**, multiple real queries for the same hostnames are accompanied by different sets of dummy queries.

We found that the effectiveness of range queries depends on the following parameters:

- The number of dummy queries n, which is added to each real query,
- the size of the dummy pool $|P|$, and
- the dummy generation strategy (random dummies or static dummies).

We analyzed the influence of the three parameters in the real-world model using the *TWODAYS dataset* (cf. Sect.4). All results presented in the following were obtained by training the 1NN-COSIM classifier (TFN-1) and a session length of 24 hours. Without range queries the classifier achieves an accuracy of 69.6 % (*baseline*).

Influence of Dummy Pool Size. Figure 7 shows the influence of the size of the dummy pool P with $n = 5$. The results suggest that static dummies are quite ineffective: Characteristic patterns present in the real queries are re-inforced by the dummies.

The effectiveness of random dummies depends on $|P|$: Increasing $|P|$ also increases the effectiveness of range queries at first (decreasing the observed accuracy of the classifier). Accuracy reaches its minimum for values of $|P|$ between

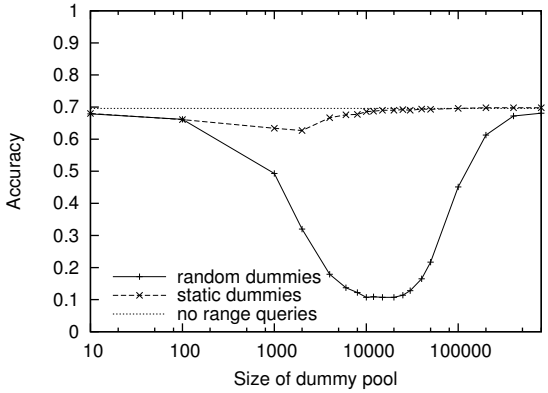


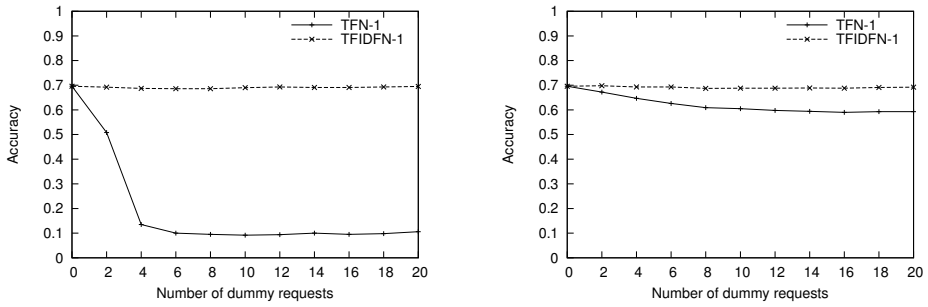
Figure 7: Influence of pool size $|P|$ (for $n = 5$ dummies per real query)

10,000 and 15,000 hostnames (10.7 %). Increasing the dummy pool further reduces the effectiveness of range queries again. For $|P| = 400{,}000$ we observed an accuracy of 67.2 %, which is close to the baseline (indicated by the dotted line).

The observed dependence on $|P|$ can be explained as follows: If a user issues m real queries within a session, he will send $m \cdot n$ dummy queries. For random dummies almost all available dummy hostnames from the pool will be used if $N \approx m \cdot n$. Because all users share the same dummy pool, their resulting profiles become quite similar to each other, which makes tracking more difficult. Moreover, there is a high likelihood for collisions because practically all hostnames of real queries that are also in the dummy pool will be subject to noise. At this point the combined weight of the dummy hostnames overpowers the real queries.

For $N < m \cdot n$, several dummy hostnames will be drawn multiple times from the common pool. Thus, again, instances from different users will have the dummy attributes in common, but different volumes of real queries will cause the access frequencies of the dummy hostnames to vary among users. This difference can be utilized by the 1NN-COSIM (TFN-1) classifier, which takes into account normalized access frequencies within instances. The weight of the real queries overpowers the dummies in this case. Collisions with real queries are less likely because the intersection of the set of real queries and the dummy pool becomes smaller.

On the other hand, for $N > m \cdot n$ with increasing N it becomes more likely that each user is drawing different hostnames from the dummy pool, which will neither be present in other sessions of him nor of other users. These dummies do not affect classification at all. Moreover, due to the sparseness of the real instance vectors, collisions with real queries become more unlikely, because the dummy pool contains more unpopular hostnames and because of the fact that access frequencies follow a power law.



(a) Random dummies (b) Static dummies

Figure 8: Influence of number of dummies per real query n (with $|P| = 50000$)

This reasoning is supported by the underlying data: In the TWODAYS dataset the average number of real queries per user equals $\bar{m} = 2186$ in the training session and $\bar{m} = 2127$ in the test session, which suggests that effectiveness should be high for $N \approx 10{,}000$, which is in line with the results presented above.

Influence of Number of Dummies. Another crucial factor for the effectiveness of range queries is the number of dummy queries n that accompany each real query. We evaluated the influence of n using the 1NN-COSIM (TFN-1) classifier for random and static dummies for a constant pool size $|P|$. We show the results of experiments with the pool sizes, for which we observed the highest effectiveness in Fig. 7: 15,000 hostnames for random dummies and 2,000 hostnames for static dummies.

Figure 8a shows, that the accuracy drops rather rapidly from 69.6 % (no dummy requests) to 50.8 % for $n = 2$ and 10.0 % with $n = 6$. Increasing the number of dummy queries further does not offer any additional benefits. The same influence, albeit less obvious, can be observed for static dummies in Fig. 8b.

In both cases, however, the effect of range queries is almost completely neutralized by the **application of the IDF transformation** (cf. results for TFIDFN-1 in the figures). This finding – which we observed regardless of the value of $|P|$ – can be explained by the fact that all users draw dummies from the same P in our experiments. The IDF transformation is effective in this case, because it reduces the weight of attributes that appear in instances of a large number of users. While using a different pool P_i for each user u_i would reduce the utility of the IDF, it would also introduce new issues. This becomes apparent considering the consequences of non-overlapping pools $P_i \cap P_j = \emptyset$: In this extreme case each user would use a set of very characteristic dummies, which might actually assist during tracking. We leave a more detailed analysis of the ramifications of non-overlapping pools for future work.

7.3. Specific Countermeasures for DNS

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 the actual query volume from the adversary. Motivated by the fact that many DNS records have a time-to-live of 24 hours, we simulate a cache, whose entries expire at the end of each day. For sessions with a duration of 24 hours this results in binary attributes that only indicate whether a user has accessed a hostname on that day or not. However, the effect of such a cache is very limited: accuracy only drops from 85.4 % to 83.1 % for MNB-P (TFIDFN-1+2) and from 85.3 % to 83.0 % for 1NN-COSIM-P (TFIDFN-1+2), respectively. In light of the comparatively poor result of 1NN-JACCARD these figures are quite surprising as they suggest that behavior-based tracking *doesn't necessarily rely* on different usage intensities of individual web sites.

In order to prohibit behavior-based tracking carried out by DNS resolvers, users could also **distribute their queries** over multiple DNS resolvers. We have shown in previous work that distributing queries randomly can in fact reduce the effectiveness of tracking, but only if sufficiently large numbers of non-collaborating servers are available (Herrmann et al., 2012b).

Users could also **refrain from using a (third-party) DNS resolver** altogether, i. e., they could either run their own DNS resolver locally, or send their DNS queries directly to authoritative name servers instead. However, this may introduce new privacy issues, because this practice will disclose the IP addresses of the users to the authoritative name servers.

7.4. Discussion of Results

At first sight, a countermeasure that manages to reduce the accuracy of behavior-based tracking will also prevent the adversary from gaining Adv_1 (insights from long-term profiling) and Adv_2 (insights due to detecting the presence of a user). However this implication does not apply universally, as illustrated by the following discussion.

The results for our experiments with shorter session lengths are promising. The extreme case of issuing each query from a different address may be infeasible today, but will become possible with the advent of IPv6 (cf. Herrmann et al., 2012a, for a discussion of ramifications). If the adversary loses track of a user due to short-lived sessions, this measure can prevent the adversary from gaining Adv_1. In principle it is also effective against *presence detection* efforts (Adv_2). Note, though, that the adversary *will* gain Adv_2, if users (or their machines) issue queries for characteristic hostnames that are not issued by anyone else (this also applies to range queries).

Range queries have been designed to obfuscate the real queries of a user. However, according to our results, it is rather difficult to find a configuration that succeeds in preventing behavior-based tracking. Moreover, our results indicate that in case of a shared dummy pool P an adversary can still link multiple sessions of the same user due to the IDF transformation. Thus, the variants we studied cannot prevent the adversary from gaining Adv_2. Being ineffective

against behavior-based tracking, range queries could still be considered as a suitable tool to counter long-term profiling efforts (Adv_1): after all, they produce a distorted user profile, which should prevent the adversary from inferring the true interests of a user. However, this conjecture is doubtful. Recent work on private web search indicates that obfuscating interest profiles is far more difficult than assumed so far (Balsa et al., 2012).

Finally, we assumed during the evaluation of range queries that all users issue dummy queries in the same way. Tracking protection degrades further, if only individual users employ range queries, or if users choose different values for n and $|P|$. In the worst case range queries could actually assist the adversary in tracking a user, e. g., because of peculiar query bursts or the increased query volume.

8. Generalizability of Findings and Future Work

While our results have been obtained using DNS traffic, our methods and findings (summarized below) are not limited to the scenario of a curious DNS resolver. There are a number of potential adversaries (cf. Sect. 1) that comply with our adversarial model (cf. Sect. 3).

Moreover, we have found no evidence that our findings depend on the peculiarities of DNS traffic. DNS differs from other protocols such as HTTP in two regards: Firstly, **not all requests issued by the user will result in DNS queries**, which is due to caching of DNS entries. It is reasonable to assume that more complete profiles will not result in lower accuracy. Secondly, **not all DNS queries are actually issued by the user**; his machine may issue queries on its own: We observed type A queries from, among others, applications and anti-virus tools checking for updates, browser toolbars loading new banners, and from e-mail clients. We also observed queries for other types (cf. Sect. 4). While we cannot accurately assess the impact of automated queries with our dataset, we can at least report that they *do* have a beneficial effect: We discarded all queries that are definitely not issued from within the web browser, namely all queries with query types different from type A, and observed that accuracy degrades slightly (MNB-P, TFIDFN-1+2) from 85.4 % to 79.5 %.

Our main findings are summarized below:

1. We showed that judging the feasibility of behavior-based tracking solely based on controlled experiments may lead to wrong conclusions (Sect. 6.5).
2. In the real-world setting, the PM techniques and 1NN-JACCARD are less effective than 1NN-COSIM and MNB, which are aware of access frequencies. On the other hand, accuracy degrades only slightly, if the actual access frequencies are unavailable (Sect. 7.3).
3. Pruning superfluous predictions utilizing 1NN-COSIM increases accuracy in the real-world setting.
4. Profiles only age rather slowly. Behavior-based tracking remains feasible even if user profiles are not updated daily.

5. Decreasing the session length also decreases the attainable accuracy. Even changing one's IP address every few minutes cannot fully prevent behavior-based tracking.
6. The majority of users can be tracked with success using only a fraction of all available attributes (e. g., the 10 most significant attributes per instance or the 10,000 most popular hostnames).
7. To offer protection from behavior-based tracking, range queries must be carefully tuned, which is difficult due to the complex relationships between the pool size and the range size as well as the dependency on the volume of the traffic to be obscured. Static dummies are less effective than random dummies.
8. If all users share the same set of dummy queries, the application of the IDF transformation renders range queries useless.
9. Preventing long-term profiling efforts (Adv_1) may be possible, but preventing presence detection (Adv_2) is more difficult (Sect. 7.4).

Future Work. This research can be extended in various ways. First of all, we could relax some aspects of our model (cf. Sect. 3), such as the aspect that there are fixed epochs. Instead, we could model users that **change their IP addresses arbitrarily**, and study whether the adversary can still capture concise sessions. Furthermore, it would be interesting to **understand the reasons for trackability**, i. e., why some users can be tracked more thoroughly than others and the parameters this depends on. Furthermore, having a closer look at the role of **automated, periodically issued queries** would be worthwhile.

9. Conclusion

The behavior-based tracking techniques studied in this paper enable adversaries, who have access to the (web) requests of users, to link multiple sessions unobtrusively, without cookies or other explicit identifiers. The techniques exploit the fact that most users or their machines issue a characteristic set of queries repeatedly. An adversary can gain two advantages via behavior-based tracking: Firstly, he can use it for the same purpose as conventional cookies, i. e., to create long-term profiles capturing users' interests. Secondly, the adversary can use behavior-based tracking to detect the presence of a certain user, which may allow him to track the movements of roaming users on the network or even geographically.

Our evaluation on a large-scale DNS dataset indicates that behavior-based tracking is feasible in practice. Dynamic IP addresses, which are sometimes cited to be important for privacy, offer only very limited protection against this attack: Our best technique achieves an accuracy of 85.4 % for sessions with a duration of 24 hours. Changing IP addresses multiple times per day would reduce the effectiveness of the attack considerably. In contrast, the range query techniques could not prevent behavior-based tracking in our experiments.

While we have evaluated behavior-based tracking only for the case of a curious DNS resolver, our results are of general interest. The described tracking

techniques can be adopted by any adversary that has access to a significant share of the requests of users.

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 Elmo Randschau. Finally, we thank our colleagues Karl-Peter Fuchs and Christoph Gerber for their critical feedback and constructive comments.

References

Adamic L, Huberman B. Zipf's Law and the Internet. Glottometrics 2002;3(1):143–50.

Ayenson M, Wambach DJ, Soltani A, Good N, Hoofnagle CJ. Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. Available at SSRN: http://ssrn.com/abstract=1898390; 2011.

Balsa E, Troncoso C, Díaz C. OB-PWS: Obfuscation-Based Private Web Search. In: IEEE Symposium on Security and Privacy. IEEE Computer Society; 2012. p. 491–505.

Banse C, Herrmann D, Federrath H. Tracking Users on the Internet with Behavioral Patterns: Evaluation of its Practical Feasibility. In: Gritzalis D, Furnell S, Theoharidou M, editors. Information Security and Privacy Research – 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings. Springer; volume 376; 2012. p. 235–48.

Beesley KR. 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. 1988. p. 12–6.

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. New York, USA: Springer-Verlag; 2001. p. 115–29.

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. Washington, DC, USA; 2009. p. 411–6.

Cavnar WB, Trenkle JM. N-Gram-Based Text Categorization. In: Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. 1994. p. 161–75.

Chor B, Kushilevitz E, Goldreich O, Sudan M. Private Information Retrieval. J ACM 1998;45(6):965–81.

Cortes C, Vapnik V. Support-Vector Networks. Machine Learning 1995;20(3):273–97.

Damashek M. Gauging Similarity with n-Grams: Language-Independent Categorization of Text. Science 1995;267(5199):843–8.

Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM 2008;51(1):107–13.

Dingledine R, Mathewson N, Syverson PF. Tor: The Second–Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium. 2004. p. 303–20.

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. SIGKDD Explorations Newsletter 2009;11:10–8.

Han J, Kamber M, Pei J. Data Mining: Concepts and Techniques. 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

Herrmann D, Arndt C, Federrath H. IPv6 Prefix Alteration: An Opportunity to Improve Online Privacy. 1st Workshop on Privacy and Data Protection Technology (PDPT 2012) colocated with Amsterdam Privacy Conference (APC 2012) 2012a;URL: http://arxiv.org/abs/1211.4704.

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. Berlin, Heidelberg: Springer-Verlag; 2012b. p. 136–54.

Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: International Joint Conference on Artificial Intelligence. Morgan Kaufmann; volume 14; 1995. p. 1137–43.

Kumpošt M. Data Preparation for User Profiling from Traffic Log. 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies 2007;:89–94.

Kumpošt M. Context Information and user profiling. Ph.D. thesis; Faculty of Informatics, Masaryk University, Czeck Republic; 2009.

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. Berlin, Heidelberg: Springer-Verlag; 2009. p. 1–10.

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. IEEE Computer Society; 1997. p. 364–73.

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). IEEE; 2010. p. 1–10.

Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge, UK: Cambridge University Press, 2008.

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; 2006.

Rieck K, Laskov P. Language Models for Detection of Unknown Attacks in Network Traffic. Journal in Computer Virology 2007;2(4):243–56.

White T. Hadoop – The Definitive Guide: Storage and Analysis at Internet Scale. 2nd ed. O'Reilly, 2011.

Witten IH, Frank E. Data Mining. Practical Machine Learning Tools and Techniques. San Francisco: Elsevier, 2005.

Yang Y. Web user behavioral profiling for user identification. Decision Support Systems 2010;49:261–71.

Yang Y, Padmanabhan B. Toward user patterns for online security: Observation time and online user identification. Decision Support Systems 2008;48:548–58.

Zhao F, Hori Y, Sakurai K. Analysis of Existing Privacy–Preserving Protocols in Domain Name System. IEICE Transactions 2010;93-D(5):1031–43.

Zipf GK. The psycho-biology of language. An introduction to dynamic philology. 2nd ed. Cambridge/Mass.: M.I.T. Press, 1968.