Behavior-Based Tracking of Internet Users with Semi-Supervised Learning

Dominik Herrmann*, Matthias Kirchler†, Jens Lindemann‡, and Marius Kloft†
*Universität Siegen, Germany, dh@exomail.to
†Humboldt Universität Berlin, Germany, matt.kirchler@gmail.com, kloft@hu-berlin.de
‡Universität Hamburg, Germany, lindemann@informatik.uni-hamburg.de

***Abstract*—Behavior-based tracking is an unobtrusive technique that allows observers on the Internet to monitor user activities over long periods of time – in spite of changing IP addresses. Our technique uses semi-supervised machine learning, which allows observers to track users without the need for multiple labeled training sessions. We present evaluation results obtained on a realistic dataset that contains the DNS traffic of 3,800 users. Given the traffic of one week, our simulated observers can link the sessions of up to 87 % of the users with surprisingly little effort. Our results indicate that observers can leverage unlabeled sessions to increase the robustness of existing tracking techniques. This makes it more difficult for users to protect their privacy on the Internet.**

I. Introduction

Behavior-based tracking allows observers on the network to link multiple sessions of Internet users, for instance for the purpose of behavioral profiling [1]. It exploits the fact that human behavior exhibits characteristic and recurrent patterns [2]. In contrast to existing tracking approaches, it does not rely on explicit identifiers such as (super-)cookies or client-side scripting, but is entirely carried out on the server side. As users have no means to detect behavior-based tracking on the client side, it constitutes a considerable threat to privacy.

Previous work [3] has raised awareness that behavior-based tracking could be used by *recursive name servers* (resolvers) in the Domain Name System (DNS) to link adjacent sessions of individual users in spite of changing IP addresses. This is especially relevant for users of popular *third-party* DNS resolvers such as Google Public DNS, which reportedly handles about 13 % of all DNS queries [4]. However, behavior-based tracking is not limited to DNS. The results of [3] indicate that it could also be adopted by *ad networks* such as Google Doubleclick [1] to resurrect deleted third-party cookies. Further, it can be used for forensic purposes [5].

We show that behavior-based tracking can be performed more accurately than reported previously. Existing work on user re-identification relies on *supervised* machine learning techniques and (rather optimistically) assumes that the observer has access to a large amount of labeled training instances of each user that is to be tracked. However, many Internet service providers assign short-lived IP addresses, often changing every 24 hours [6], [7]. As a result, the observer has access to only a single labeled training instance per user. We overcome this limitation with a **semi-supervised approach**, which combines unsupervised and supervised techniques [8]. This allows us to make use of unlabeled training data, which turns out to increase accuracy considerably. On average, the sessions of 87 % of the users can be linked correctly.

We review related work (Sect. II), describe our technique (Sect. III), and provide evaluation results (Sect. IV). We conclude in Sect. V.

II. Related Work

Kumpošt and Matyáš [9] re-identify users by comparing the cosine similarity of vectors that contain the number of connections to HTTP(S) and SSH hosts identified by their IP addresses. However, they only consider linking sessions that last over a full month and they report relatively high false-positive rates of 21 % (SSH) and 68 % (HTTP). Yang [2] tries to authenticate users based on the visited websites using decision trees, support vector machines, and association rules. Given 300 labeled sessions, she achieves an accuracy of up to 87 % for 100 concurrent users, but reports she was unable to scale up her experiment. Kim and Zhang [10] report a tracking accuracy of 98.74 %, but they assume a user's queries to be linkable across multiple days during training.

Behavior-based tracking of *individual* sessions has been studied in [3]. However, the proposed approach is limited to re-identifying users from one day to the next in a stateless fashion. The authors report tracking accuracies of up to 85.4 % on the dataset that is also used in this paper. However, this value also includes correct predictions about *inactive* users (accounting for about 15 percentage points), i. e., their results cannot be compared to ours.

III. Semi-supervised Tracking Technique

We assume a passive adversary (the *observer*) that is able to obtain the DNS queries of a group of users, either by operating their DNS resolver, by having access to its query log, or by eavesdropping on the network. Further, we assume that all sessions have a fixed duration of one day, that every user contributes at most one session per day (i. e., all traffic that originates from the same source IP address during a day belongs to the same user), that multiple users do not share a single IP address, and that traffic from different source IP addresses belongs to different users.

Our observer is assumed to operate as follows. At the end of every day, the observer aims to link the sessions of all users that have been active on this day τ_0 *(evaluation day)*

with their respective *previous* session. While for many users their previous session will have occurred on the immediately preceding day τ_{-1}, it may also lie further back in the past for less active users. Therefore, in addition to the sessions on τ_0, the observer considers all sessions from the last w days, the *training period* $\tau_{-1}, \ldots, \tau_{-w}$. Training period and evaluation day make up the **Period under Consideration (PuC)** of the observer, which consists of $D = w + 1$ days and is shifted from day to day in the course of time.

The PuC contains n sessions (t, u, x) from an unknown number of $k \leq n$ users. We organize the session vectors in a data matrix X with dimensions $n \times d$, whose rows are the vectors $x_1, \ldots, x_n$, $n = \sum_{t=1}^{D} \#\{\text{active users on day } t\}$, and $d = \#\{\text{domains}\}$. Thus, $X_{i,j}$ is the number of times that domain j was queried in session i. Each user may be responsible for 1 to D sessions. Therefore, the aim is to find a classifier $f : \{\text{sessions}\} \subset \mathbb{R}^d \to \{\text{users}\} \hat{=} \{1, \ldots, k\}$ that correctly maps the sessions on the evaluation day to their users.

Our tracking technique uses a **cluster-based ensemble classifier (CBEC)** that consists of two stages. Stage 1 applies unsupervised learning to partition the sessions into (ideally) k clusters, each of which represents a single user. The cluster of a user u who was active on all days should contain all sessions $(1, u, *), \ldots, (D, u, *)$, but no sessions of other users. These clusters make up a classifier that is used in Stage 2 to link the sessions observed on the evaluation day with sessions from the training period. Using different initialization methods, we are able to train different instances of this k-means classifier and aggregate their output into a final prediction.

A. Stage 1: Clustering Sessions

In Stage 1 we use an adapted (spherical) k-means clustering algorithm [11]. The classic k-means method consists of three steps: *initialization*, *assignment*, and *update*.

1) Initialization: The initialization sets up k cluster centroids $c_1, \ldots, c_k$ as an initial guess. We use different initializations in parallel to obtain multiple classifiers that serve on a committee of classifiers (*ensembling*), which is a common technique to increase robustness of classification [12]. All classifiers are trained and evaluated on their own and the prediction of the ensemble is obtained by a majority vote.

Intuitively, it makes sense to initialize the centroids with a single session of every user. We consider two alternatives to estimate the number of users (clusters). Estimate k_1 is set to be the maximum number of sessions per day within the PuC, which assumes that all users have been active on one day. The second estimate is obtained by computing $k_2 = (1+p)k_1$, i. e., it reserves some space for additional users. The value of p has to be found heuristically by the observer via analysis of the dataset or comparison to historic data. However, we found that p has only an insignificant influence on accuracy: For $p \in [0.05; 1.00]$ accuracy varies by less than two percentage points. We use $p = 0.1$ for our experiments in Sect. IV.

We investigate two ways of initializing the k-means algorithm. The first method is **bootstrap sampling** [12]. For this technique we determine the value k_1 of the PuC and set up k_1 clusters. We set up D different instances of k-means by varying the day $t_i = 1, \ldots, D$ whose sessions are used to initialize the centroids. Given the day, we draw k_1 sessions from t_i independently with replacement, thus receiving a resampling sample. The second method, **optimistic sampling**, chooses k_2 as the number of clusters, but instead of drawing a bootstrap sample, we initialize a subset C of the centroids by choosing all sessions from the evaluation day $t = D$, without using sessions more than once or omitting sessions. Then we determine the missing $k_2 - \#C$ centroids by selecting those sessions from the training period, i. e., with $t < D$, that are farthest away from the existing centroids in C. This yields a single instance of k-means. We create more instances by iterating over the days $t_i = 1, \ldots, D-1$ and initializing the missing $k_2 - \#C$ instances with sessions (that are, again, farthest away from the centroids in C) from a single day t_i.

2) Assignment, Reassignment, and Update: As in the classical k-means algorithm, *assignment* and *update* steps are executed alternately until convergence, but inbetween each assignment and update step we insert a *reassignment* step.

Assignment is identical to the spherical k-means algorithm, which uses the cosine distance $1 - \frac{\langle x,y \rangle}{||x||\ ||y||}$ instead of the standard euclidean distance. Each session $x_1, \ldots, x_n$ is assigned to the nearest centroid, with respect to this measure. In the *update step*, each centroid is recomputed as the mean of all sessions assigned to it. For the *reassignment step* we use two different versions. The *soft restriction* limits the size of individual clusters to D, because each user can have at most D sessions in a PuC of length D. For the *strict restriction* we also restrict each cluster to at most one session per day. These restrictions are enforced by reassigning the most remote sessions to other clusters until the restriction is satisfied.

B. Stage 2: Associating Sessions with Users

The clusters from Stage 1 are used in Stage 2 to classify the sessions from the evaluation day, i. e., to predict which session belongs to which user (identified via a unique ID).

The association consists of two steps. In the first step we narrow down our choices. For a session $(D, u_?, x_?)$ to be classified, its corresponding cluster c_i is determined by finding the centroid with the minimal cosine distance to $x_?$. In the second step we focus on the sessions in c_i in order to make the prediction. Ideally, c_i contains mostly sessions of a single user. If c_i contained sessions that have been classified and labeled with user IDs by the observer in the past, the observer could associate $x_?$ to a user by calling a majority vote. However, in this paper we will assume that the observer does not have any knowledge about sessions from the training period. Therefore, we have to make the prediction by picking an arbitrary session from c_i. Our classifier picks the *most similar session*.

IV. Evaluation

A. Dataset and Preprocessing

We evaluate our technique with the dataset that has been used in [3]. The dataset was created at University of Regensburg (Germany) by logging all DNS queries originating

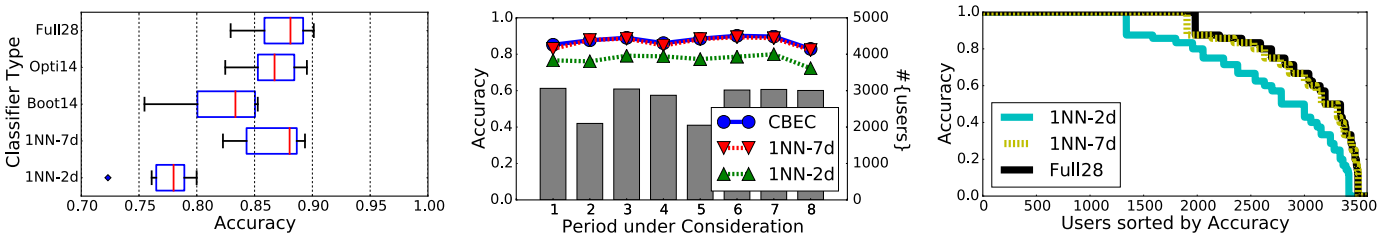

Fig. 1. (a) Aggregated accuracy of tracking techniques, averaged over 8 PuCs consisting of $D = 7$ days each. (b) Break-down of accuracy for individual PuCs over time indicates no dependence on number of concurrently active users. (c) Aggregated tracking accuracy per user for all users over all PuCs.

from the student housing network on the resolvers. In total, 431,210,371 queries for 5,010,507 distinct domains issued by 3,862 different users have been recorded between April 30, 2010 and June 29, 2010 (61 days). The log indicates date and time, a statically assigned pseudonymous user ID, and the domain of each recorded DNS query. As the dataset contains the ground truth for the mapping between users and queries, it can be used to evaluate the accuracy of our tracking technique.

We simulate daily changing IP addresses and store the vectorized sessions in our data matrix. To reduce the computation effort, we perform a rudimentary **feature selection** by removing all domains that have been queried in *fewer than six sessions*. We observed that this step has a negligible influence on tracking accuracy. We also apply a commonly used **sublinear function**, $f(x) = \log(1 + x)$, to every cell, which limits the effect of extremely large outliers [13].

B. Experimental Setup

We partition the dataset into eight non-overlapping PuCs. Each PuC consists of seven consecutive days ($D = 7$), i. e., a training period of six days followed by the evaluation day. For every PuC we simulate an observer that tries to link all sessions observed on the evaluation day. To this end, we iterate over these sessions and ask the classifier for each session $(t_i = D, u_i, x_i)$ to find the most similar session with $t < D$, i. e., a session that supposedly belongs to the same user u_i.

In order to determine the **tracking accuracy** of a classifier, we differentiate between correct and incorrect mappings. A correct mapping is scored if and only if the classifier assigns a session $(t_e = D, u_e, x_e)$ from evaluation day to a session $(t_t < D, u_t, x_t)$ from the training period and $u_e = u_t$. The tracking accuracy within a PuC is then calculated as the fraction of users on the evaluation day with a correct mapping. If two sessions $(t = D, u_1, *)$ and $(t = D, u_2, *)$ are both linked with the same session $(t < D, u_1, *)$, then u_1 scores a correct mapping, while u_2 scores an incorrect mapping.

We have performed a number of experiments in order to compare our semi-supervised CBEC technique with a supervised technique studied in previous work. We re-implemented the 1-nearest-neighbor classifier proposed in [3] that can be considered the state of the art in behavior-based tracking. Our implementation takes into account that this approach, referred to as **1NN-2d** in the following, is limited to link the sessions of two adjacent days, i. e., it considers only the sessions of the evaluation day $(t = D, *, *)$ and the sessions of the last day of the training period $(t = D - 1, *, *)$.

For CBEC we consider two ensembles that consist of 14 classifiers each. Every ensemble is initialized on one of the seven days of the PuC using either the soft or the strict restriction. The **Boot14** ensemble employs bootstrap sampling, while the **Opti14** ensemble employs optimistic sampling. In addition, we consider a large ensemble **(Full28)** comprised of all classifiers from the two smaller ensembles.

In order to analyze the differences between 1NN-2d and CBEC, we have implemented a semi-supervised variant of the 1NN-2d classifier proposed in [3], which has access to the unlabeled sessions of *all* days of the PuC (referred to as **1NN-7d** for $D = 7$). 1NN-7d links each session from the evaluation day to its most similar neighbor found in the training period. This allows us to assess to what degree CBEC benefits from having access to all (unlabeled) sessions in the training period.

C. Results

As shown in Fig. 1a, Boot14 (avg. accuracy over all PuCs: 82 %) performs rather poorly, sometimes worse than 1NN-2d (77 %). As different classifiers tend to err on different sessions, they still improve the robustness of Full28, which correctly classifies 87 % of the sessions on average. Full28 outperforms 1NN-2d in all PuCs by a considerable margin (cf. Fig. 1b). Surprisingly, 1NN-7d achieves the same accuracy (87 %). CBEC correctly tracks a majority of users (55 %) in *all* PuCs in which they are online (1NN-2d: 37 %; 1NN-7d: 53 %). 79 % (1NN-2d: 63 %; 1NN-7d: 77 %) of the users are correctly tracked in at least 75 % of their PuCs (cf. Fig. 1c).

Given the results for $D = 7$, we explored to what extent the tracking techniques benefit from **increasing the PuC length**. In a series of experiments we constructed PuCs with increasing lengths $D = 2, \ldots, 20$, ensuring that all PuCs ended *on the same date* τ_0 to ensure comparability. The influence of D is shown in Fig. 2 for 1NN as well as a single instance of k-means (soft restriction, optimistic sampling, no ensembling, therefore referred to as **CBC** in the following). For $D = 2$, CBC achieves a tracking accuracy of 77 % (1NN-2d: 78 %). Accuracy levels off at 91 % (1NN-9d: 90 %) for $D \geq 9$ and we did not observe any significant increases for higher values ($D = 20$: 90 %; 1NN-20d: 92 %).

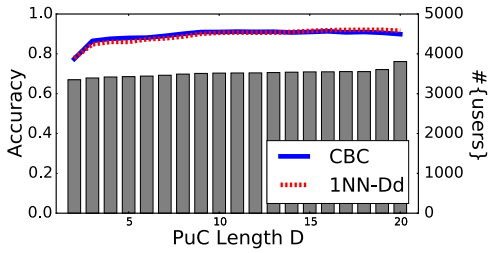

Fig. 2. The semi-supervised approach benefits from increasing D (up to $D = 9$) although the number of concurrently active users increases as well.

D. Discussion, Limitations, and Future Work

Although we have strived for a realistic evaluation, our assumptions (cf. Sect. III) may be invalid in some scenarios. For instance, multiple users in a household share the same IPv4 address today, i. e., the observer cannot differentiate between their traffic. Therefore, behavior-based tracking via DNS queries is not a substitute for third-party tracking cookies (yet). However, observers may employ it to improve the robustness of their existing tracking efforts.

CBEC outperforms 1NN-2d, which was proposed in [3]. The poor accuracy of 1NN-2d stems from its limitation to only consider sessions of adjacent days. Thus, sessions of users who are inactive on a whole day cannot be linked with each other. The CBEC technique overcomes this limitation by building a classifier with the help of the unsupervised k-means technique to utilize multiple unlabeled sessions per user.

Surprisingly, an observer can achieve almost the same accuracy with less effort by using 1NN-7d, a semi-supervised learning technique inspired by the 1NN classifier. However, our approach leaves much room for improvement for both CBEC and 1NN-7d. Future work could analyze the security of the framework proposed in [14] and extend the methodology to capture multiple data types [15] as well as different clustering objectives [16]. Further, some observers may be able to leverage context knowledge (e. g., the presence of HTTP tracking cookies within a session or the geographic location of users) to partition large user groups into smaller ones.

Our results for varying D indicate that characteristic patterns are not visible on every day, but are subject to weekly patterns, presumably due to different behavior on weekdays and weekends. Considering periods with a length of one to two weeks appears to be sufficient.

The tracking technique presented in this paper is focused on maximizing the fraction of correctly linked sessions on the evaluation day (*precision*). However, some observers may strive for maximum *recall*, i. e., linking *all* sessions of a user within a PuC. While this objective is natively supported by CBEC due to its clustering stage, it is an open question whether 1NN-7d can be adapted for this purpose and how both techniques perform. In another line of work, we are currently studying unsupervised learning techniques [17].

Finally, we intend to design and analyze unobtrusive defenses. Potential candidates include anonymizing mixes [18], dummy queries [19], and short-lived IP(v6) addresses [20].

V. Conclusion

The contribution of this paper is to improve our understanding of the abilities of observers on the network, which is a critical building block for the development of effective countermeasures. We show that the *unlabeled* user sessions of one week may be sufficient for behavior-based tracking. We obtain an accuracy of 87 % on a dataset with more than 3,800 users. 55 % of the users are correctly re-identified every time. Behavior-based tracking is a threat to privacy, because it cannot be detected on the client side.

Acknowledgment

Marius Kloft acknowledges support from DFG (grant KL 2698/2-1) and BMBF (grants 031L0023A and 031B0187B).

References

[1] Z. Yu, S. Macbeth, K. Modi, and J. M. Pujol, "Tracking the Trackers," in *WWW*, 2016, pp. 121–132.
[2] Y. Yang, "Web user behavioral profiling for user identification," *Decision Support Systems*, vol. 49, pp. 261–271, 2010.
[3] D. Herrmann, C. Banse, and H. Federrath, "Behavior-based Tracking: Exploiting Characteristic Patterns in DNS Traffic," *Computers & Security*, vol. 39A, pp. 17–33, Nov. 2013.
[4] APNIC Labs, "DNSSEC Validation," 2016, http://stats.labs.apnic.net/dnssec, accessed: 2016-10-01.
[5] D. Herrmann, K. Fuchs, and H. Federrath, "Fingerprinting Techniques for Target-oriented Investigations in Network Forensics," in *Sicherheit*, ser. LNI, vol. 228. GI, 2014, pp. 375–390.
[6] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *IMC*. ACM, 2009, pp. 90–102.
[7] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How dynamic are IP addresses?" in *SIGCOMM*. ACM, 2007, pp. 301–312.
[8] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.
[9] M. Kumpošt and V. Matyáš, "User Profiling and Re-identification: Case of University-Wide Network Analysis," in *TrustBus '09*, ser. LNCS, vol. 5695. Springer, 2009, pp. 1–10.
[10] D. W. Kim and J. Zhang, "You are how you query: Deriving behavioral fingerprints from DNS traffic," in *SecureComm*. Springer, 2015, pp. 348–366.
[11] I. S. Dhillon and D. S. Modha, "Concept Decompositions for Large Sparse Text Data Using Clustering," *Machine Learning*, vol. 42, no. 1/2, pp. 143–175, 2001.
[12] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.
[13] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques*. San Francisco: Elsevier, 2005.
[14] M. Kloft and P. Laskov, "Security analysis of online centroid anomaly detection," *JMLR*, vol. 13, no. Dec, pp. 3681–3724, 2012.
[15] C. Cortes, M. Kloft, and M. Mohri, "Learning kernels using local rademacher complexity," in *NIPS*, 2013, pp. 2760–2768.
[16] J. E. Vogt, M. Kloft, S. Stark, S. S. Raman, S. Prabhakaran, V. Roth, and G. Rätsch, "Probabilistic clustering of time-evolving distance data," *Machine Learning*, vol. 100, no. 2-3, pp. 635–654, 2015.
[17] M. Kirchler, D. Herrmann, J. Lindemann, and M. Kloft, "Tracked Without a Trace: Linking Sessions of Users by Unsupervised Learning of Patterns in Their DNS Traffic," in *AISec*. ACM, 2016.
[18] D. Herrmann, K. Fuchs, J. Lindemann, and H. Federrath, "EncDNS: A Lightweight Privacy-Preserving Name Resolution Service," in *ESORICS*, ser. LNCS, vol. 8712. Springer, 2014, pp. 37–55.
[19] D. Herrmann, M. Maaß, and H. Federrath, "Evaluating the Security of a DNS Query Obfuscation Scheme for Private Web Surfing," in *IFIP SEC*, ser. AICT, vol. 428. Springer, 2014, pp. 205–219.
[20] D. Herrmann, C. Arndt, and H. Federrath, "IPv6 Prefix Alteration: An Opportunity to Improve Online Privacy," *CoRR*, vol. abs/1211.4704, 2012, available at http://arxiv.org/abs/1211.4704.