Users’ Fingerprinting Techniques from TCP Traffic

Original
Users’ Fingerprinting Techniques from TCP Traffic / Vassio, Luca; Giordano, Danilo; Trevisan, Martino; Mellia, Marco; Couto da Silva, Ana Paula. - ELETTRONICO. - (2017). ([Intervento presentato al convegno ACM SIGCOMM Workshop on Big Data Analytics and Machine Learning for Data Communication Networks tenutosi a Los Angeles, California, USA nel August 21 - 25, 2017.

Availability:
This version is available at: 11583/2674705 since: 2017-06-16T12:51:10Z

Publisher:
ACM

Published
DOI:10.1145/3098593.3098602

Terms of use:
openAccess
This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)
Users’ Fingerprinting Techniques from TCP Traffic

Luca Vassio
Politecnico di Torino
luca.vassio@polito.it

Danilo Giordano
danilo.giordano@polito.it
Politecnico di Torino

Martino Trevisan
martino.trevisan@polito.it
Politecnico di Torino

Marco Mellia
Politecnico di Torino
marco.mellia@polito.it

Ana Paula Couto da Silva
Universidade Federal de Minas Gerais
ana.coutosilva@dcc.ufmg.br

ABSTRACT

Encryption at the application layer is often promoted to protect privacy, i.e., to prevent someone in the network from observing users’ communications. In this work we explore how to build a profile for a target user by observing only the names of the services contacted during browsing, names that are still not encrypted and easily accessible from passive probes. Would it be possible to uniquely identify a target user from a large population that accesses the same network?

Aiming at verifying if and how this is possible, we propose and compare three methodologies to compute similarities between users’ profiles. We use real data collected in networks, evaluate and discuss performance and the impact of quality of data being used. To this end, we propose a machine learning methodology to extract the services intentionally requested by users, which turn out to be important for the profiling purpose. Results show that the classification problem can be solved with good accuracy (up to 94%), provided some ingenuity is used to build the model.

CCS CONCEPTS

• Networks → Network privacy and anonymity; Network measurement; • Security and privacy → Privacy protections;

KEYWORDS

Passive Measurements, Machine Learning, Privacy, User Fingerprint, User Profiling.

1 INTRODUCTION

Privacy and user tracking are hot topics that impact everyone who uses the Web. When online, we offer information about our interests, habits, system configuration, etc., and someone able to eavesdrop the traffic that our devices exchange with the network could invade our privacy. End-to-end encryption – HTTPS – limits access to exchanged information, thus mitigating the problem. Yet, some traffic is transmitted without encryption, such as network and transport layer data.

In this work we explore different techniques for profiling and fingerprinting users leveraging the set of domains\(^1\), contacted during web navigation. Indeed, domain names are typically exchanged in clear text, i.e., when resolving a domain via DNS queries because DNS traffic is not encrypted (even DNSSEC does not guarantee confidentiality). Would it be possible to then build a user profile by simply considering the set of domains she visits during her browsing session? And would it be possible to re-identify her in a future time, e.g., when she is connected in a different network? Real-case scenarios include applications for tracking users in different networks, e.g., tracking users from both mobile and house traffic from a certain area, in which we may want to associate the two datasets. Or, when users change their IPs due to dynamical assignment.

Armed with two large datasets containing traffic summaries of \(\approx 7,500\) anonymized users during 4 weeks in 2017, we answer the previous questions. A big data approach must be considered when creating meaningful fingerprints. We investigate the use of three metrics, considering simple Jaccard index, an information theory Maximum Likelihood approach [9], and a text mining methodology based on TFIDF (Term Frequency - Inverse Document Frequency [8]). We evaluate their performance, highlighting their strengths, weaknesses and trade-offs. Results unveil that TFIDF offers overall the best performance, identifying a given user in different scenarios with up to 94% of accuracy. The rationale of this surprising result is the fact that among the hundreds of domains visited during few days, many are persistent in time and create a peculiar and unique mix of traffic.

The creation of fingerprints to profile users is a problem widely studied in literature under different perspectives. Here we briefly list those works where a limited set of features are considered. Authors in [6] collected volunteers’ web browsing histories (i.e., full URLs), discovering that for 97% of the users, as little as 4 web pages uniquely distinguish them. Authors in [4] used the DNS traffic to build user fingerprints, reaching accuracy up to 74% in recognition of users in real world traces. Authors in [5] tackle the user tracking and recognition problem exploiting HTTP, HTTPS and SSH protocols, achieving a 50% accuracy. Differently from previous works, we rely on large flow level datasets in which we only consider the name of the contacted server as feature, comparing different methodologies. This work extends our previous work [3] where we evaluated the usage of the DNS requests to create a fingerprint and track the users over time, discovering that with the simple Jaccard

\(^1\)We use the term domain informally throughout the paper, meaning Fully Qualified Domain Name (FQDN)

similarity method and one day of traffic only 60% of users can be correctly identified in the future.

To get more insights, we investigate which domains are more useful for such purpose, in particular considering those intentionally visited by the users, that we call Core domains, or those contacted by the browser to fetch objects that compose a web page or by other background applications, that we call Support domains. To automatically identify them, we propose a methodology based on machine learning, and, more specifically, on decision trees. Results show that the intentionally visited web-services prove to better characterize the user than Support domains; however users are better re-identified when all the traffic is taken into account, suggesting that even Support domains help in characterizing users.

Our study, although preliminary, shows on the one hand how complicated is to protect our privacy when online; on the other hand, the potentiality of good similarity metrics and machine learning applications linked to, e.g., forensic. To foster new studies and permit results reproducibility, we contribute our dataset to the community.²

2 METRICS FOR SIMILARITY

In this section we provide a description of the three different methodologies upon which we base our profiling and identification techniques. Consider a user \( u \) in our collection of users \( U \). In a time period \( \Delta T_1 \) she visits a set of domains \( D_u, \Delta T_1 \). Let us call \( \hat{D} \) the set of all the domains seen by the population at time \( \Delta T_1 \), i.e., \( \hat{D} := \cup_{u \in U} D_u, \Delta T_1 \). Given the profiles \( D_u, \Delta T_1 \) for all \( u \in U \) in a certain \( \Delta T_1 \), we suppose to have the set \( D_v, \Delta T_2 \) in a future \( \Delta T_2 \) only for user \( v \), but without knowing the identity of user \( v \) in \( \Delta T_2 \). Our goal is to correctly identify the user \( v \) among the profiles of users in \( U \) built in the past. The underlying hypothesis is that there is a positive-correlation among the domains retrieved by a user in different time windows.

2.1 Jaccard Index

The Jaccard index measures similarity between two finite sets. It is defined as the size of the intersection divided by the size of the union of the two sample sets:

\[
Jac(D_u, \Delta T_1, D_v, \Delta T_2) := \frac{|D_u, \Delta T_1 \cap D_v, \Delta T_2|}{|D_u, \Delta T_1 \cup D_v, \Delta T_2|}
\] (1)

Jaccard index is in \([0, 1]\), and it is equal to 0 when there is no common element between the two sets, while if the two sets contain same elements, the index equals 1.

2.2 Maximum Likelihood Estimation

For this method, we assume a simple behavioral model for the visited domains in which a user’s likelihood of visiting a certain domain is governed by the domain overall popularity and whether this domain already appeared in her previous domains set. Then, for each user \( u \in U \), we compute her likelihood of generating the domain set \( D_u, \Delta T_1 \) under the imposed model. This method, with few modifications, has been already proposed in [9], where the proof can be found.

²Anonymized sets of domains are available to the public at http://bigdata.polito.it/content/domains-web-users

Let us suppose that each domain \( d \in \hat{D} \) has a certain likelihood \( p(d) \) of being picked in \( \Delta T_2 \), independently from the user, given its popularity in \( \Delta T_1 \):

\[
p(d) := \frac{|u : d \in D_u, \Delta T_1|}{|U|}
\]

We suppose a user is more likely to visit a domain she already visited in the past. For any \( u \in U \) and parameter \( r \geq 0 \) we define a random variable \( H(u, r) \) s.t.:

\[
Pr(H(u, r) = d) := \left\{ \begin{array}{ll} r \cdot p(d)/z & \text{if } d \in D_u, \Delta T_1 \\ p(d)/z & \text{otherwise} \end{array} \right.
\] (2)

where \( z \) is a normalizing factor. Then, given a number equal to \( |D_u, \Delta T_2| \) of i.i.d. draws of \( H(u, r) \), the Maximum Likelihood Estimation \( \hat{u}, \hat{r} \) of the underlying parameters for producing the set \( D_u, \Delta T_2 \) are:

\[
\hat{u} = \arg\max_{u \in U} \left\{ q_u \log \frac{q_u}{s_u} + (1 - q_u) \log \frac{1 - q_u}{1 - s_u} \right\}
\]

\[
\hat{r} = \left( \frac{q_{\hat{u}}}{1 - q_{\hat{u}}} \right) / \left( \frac{s_{\hat{u}}}{1 - s_{\hat{u}}} \right)
\]

where \( q_u = |D_u, \Delta T_1 \cap D_v, \Delta T_2|/|D_v, \Delta T_2 \cap \hat{D}| \) and \( s_u = \sum_{d \in D_u, \Delta T_1} p(d) \). \( q_u \) is the fraction of domains of \( D_u, \Delta T_1 \) that are in a previous set \( D_u, \Delta T_1 \). \( s_u \) is the generalized size of \( D_u, \Delta T_1 \), where it accounts both for the total number of domains in \( D_u, \Delta T_1 \) and the popularity of those items. Intuitively, \( \hat{u} \) is a user for which \( q_{\hat{u}} \) is large and \( s_{\hat{u}} \) is small; that is, \( D_u, \Delta T_1 \) is not too big, but contains many of the domains in the observed history \( D_v, \Delta T_2 \). The model allows for \( r < 1 \), in which case \( D_{\hat{u}, \Delta T_2} \) is an anti-recommendation set. However, in our case we consider here only the cases where \( r > 1 \). If such \( r \) does not exists, we consider \( \hat{u} \) for which \( r \) is bigger.

2.3 Cosine similarity based on TFIDF

TFIDF is the product of two statistics, Term Frequency and Inverse Document Frequency, and it is widely used in information retrieval. TFIDF reflects how important a domain \( d \) is for a user \( u \) in \( U \), with respect to the set of all users \( U \).

Term Frequency (TF) measures the importance of domains for user \( u \). Differently from the classic version of TF, since we are using just sets of domains it will be independent of a particular domain:

\[
TF(u) := \frac{1}{|D_u, \Delta T \cap \hat{D}|}
\]

IDF measures how important a domain is in the whole collection of users, in \( \Delta T_1 \). While computing TF, all domains are considered equally important. However certain domains may be very popular and therefore have little importance. Thus, we weight less the frequent domains while scale up the rare ones.

Notice that for a user in \( \Delta T_2 \), we are removing the domains never seen by the population at time \( \Delta T_1 \), because such new domains will not have an Inverse Document Frequency (IDF) – remind that \( \hat{D} \) is built at \( \Delta T_1 \).

IDF is defined as:

\[
IDF(d) := \log \frac{|U|}{|u : d \in D_u, \Delta T_1|}
\]
When a browser visits a Web page, it first downloads the main images, scripts, advertisements, etc.). These are often hosted on external servers (e.g., CDNs) having different domains. We call Core domain a domain originally contacted to download the main HTML document of a page. Core domains are important since they are visited intentionally by users, like www.facebook.com and en.wikipedia.org. Instead, we call Support domains all the remaining ones, i.e., the ones automatically triggered by a visit to a website, or contacted by background applications, like static.xx.fbcdn.net and dl-client.dropbox.com. Support domains do not contain useful information about user intention. When analyzing network traffic, having a list of all possible Core domains is important to make user behavior emerge. Are Core domains important also for user tracking and identification?

In the literature, previous works tackled the problem of identifying intentionally visited domains, also called user actions. Authors of [12] exploit the referer field in HTTP requests to reconstruct web page structures from HTTP traces, while machine learning techniques are used by authors of [11] to automatically build rules for such purpose. However, an important fraction of traffic is becoming encrypted, making the above approaches ineffective and leaving passive probes with no visibility on HTTP fields. Few efforts have been put in identifying user actions from encrypted traffic; authors of [2] exploit machine learning techniques to identify user actions in Android devices, where most of the traffic is TLS, while those of [7] build traffic fingerprints for mobile applications. All these works aim at identifying user actions from flow level measurements examining traffic at runtime. Here, on the contrary, all we need is to build a pre-computed list of domains that typically contain user actions since they host actual Web services.

Given a domain, deciding if it is a Core or Support domain is a classification problem. Instead of building a custom heuristic to solve the problem, we opt for a machine learning approach by means of a decision tree classifier. First, we need to define the set of features to use: we consider an extensive list guided by domain knowledge, and let the classifier choose the ones that better allow us to separate Core and Support domains. Features include the length and the content type of the main HTML document (if present); the number of objects of the page and domains contacted by the browser to fetch all objects; HTTP response code (e.g., 2xx, 3xx and 4xx); and whether the browser has been redirected to an external domain. To get the set of features, we use active crawling, and visit the home page of each domain by means of Selenium automatic browser to extract page features.\(^3\)

To train the classifier, we build a labeled dataset that we use for training and testing. For this experiment, we consider a list of 500 Core and 500 Support domains. More in detail, we picked the list of domains found in the Campus trace (see next Section for details), sorted by number of visits. Then, we manually visited each home page (if existing) corresponding to the domain name. We label such domain as a Core or a Support domain by looking at the rendered web page. We stop the labeling process when we reach 500 items for each class, we obtain a balanced labeled dataset that we make publicly available.\(^4\) For the decision tree, we opt for the J48 implementation of the C4.5 algorithm offered by Weka.\(^5\)

Interestingly, the final decision tree results in a very simple, efficient, and descriptive model which reads as: a) the main HTML document size must be bigger than 3357B and b) the browser must not be redirected to an external domain. Intuitively, support domains typically lack of real home page, and reply with a raw and simple text message. In some cases, Support domains redirect visitors to the service’s home page (e.g., fbcdn.net redirects on www.facebook.com).

Despite its simplicity, overall accuracy is higher than 96% when tested against 1000 labeled domains, using 10–fold cross validations.

### 4 DATASETS FOR EXPERIMENTS
For our analysis we mainly rely on a dataset collected in our University Campus, where we consider the traffic of approximately 2 500 users. Users are faculty members whose terminals are directly connected to the Internet via wired Ethernet, using fixed IP addresses that we use as identifier of the terminal itself. Hence we assume each IP address is associated with one and only one user.

We rely on Tstat [10] to perform passive measurements. Tstat monitors each TCP connection, exposing information about more than 100 metrics, from IP addresses and port numbers, to fields coming from the DPI module. Here, we are interested in retrieving the domain of the server being contacted. Tstat implements three

\[^3\]http://www.seleniumhq.org/
\[^4\]http://bigdata.polito.it/content/domains-web-users
\[^5\]http://www.cs.waikato.ac.nz/ml/weka/
techniques to get it. For plain HTTP flows, the Host header is extracted from HTTP request. In case of TLS, the DPI module provides the Server Name Indication (SNI) field in Client Hello message. At last, it reports the domain name clients resolved via DNS queries prior to flows [1]. We combine these three sources to label each flow with a domain indication, giving higher priority to Host/SNI fields where more than one is present.

First row of Tab. 1 summarizes the characteristics of Campus dataset. In total, we observed about 404 k unique domains, corresponding to more than 136 k unique second level domains. In total, 691 millions flows have been observed during 4 weeks between January and February 2017; no holidays or festivities occurred during this period. We load and process the logs using Apache Spark in a 20-machine Hadoop cluster, capable of reading from disk and processing the Campus dataset in about 20 minutes. With the same machine, it took about 1 hour for classifying 404 k domains as Core or Support domains. The Internet access speed (1Gbps in our case) was the main bottleneck.

Fig. 1 shows the number of unique domains (Core and Support domains) seen in the whole dataset. New domains are continuously discovered, reaching ≈ 404 k after 4 weeks, of which Core domains are ≈ 115 k (left y-axis). Next, we observe the number of active users creating some traffic for each hour; they vary from 400 at night to almost 1 400 at midday. We plot the median over the users of unique domains discovered over time (right y-axis). Ratio between Support and Core domains increases, with a median of 662 Support and 70 Core domains after 4 weeks. Domain popularity is unbalanced: a very small portion of domains are popular and just 634 domains out of the 400 000 are contacted by 10% or more of the users. Core domains are about 32% of the total; however, only 4% of the top 1 000 most popular domains are Core domains.

Finally, in the experiment in Sec. 5.5 we use a second trace. We deployed Tstat during the same period in a PoP of an European ISP where the traffic of ≈ 5 000 ADSL and Fiber households is aggregated. All households have a fixed IP address which identifies a home gateway acting as NAT router, behind which multiple user devices may be connected to the Internet. Therefore a single identifier (i.e., the client IP address) can actually hold more than one user. Second row of Tab. 1 reports the dataset statistics.

4.1 Ethical and privacy implications

Both the data collection process and the collected data have been discussed, reviewed and approved by the ethical board of our University. We took all possible actions to protect leakages of private information from users. In particular, we anonymized the IP addresses of clients using a technique based on irreversible hash functions, only retaining the data that is strictly needed for our study. More in detail, we limit the data to i) the anonymized client

\[ SNI \text{ is a TLS extension by which the client indicates the domain of the server that is trying to contact.} \]

4.2 USER FINGERPRINT AND IDENTIFICATION

In this section, we show different case studies for defining a user fingerprint. Our main task is to identify a user, by tracking her visited domains. In the first phase, we profile the users creating fingerprints, while in a second phase, we try to identify a given user in a later trace. All clients are provided with a public and fixed IP address that we use as identifier to build the ground truth. We consider 1 205 users that were active during the period of time of data collection. The constraints used for choosing the 1 205 users are written in each experiment. When otherwise stated, the 1 205 users are chosen randomly. Performance of each similarity metric is measured by the percentage of users correctly identified. The results can be meaningfully compared since they are always related to 1 205 possible cases.

5.1 Role of number of domains

The first set of results aims at assessing the importance of the number of Core and Support domains for the profiling and identification tasks. We collect the domains visited by users and generate a profile \( D_{u, AT} \) for each \( u \in U \). In the next step, we associate user \( u \) with profile \( D_{v, AT} \) to the most likely user \( u \in U \). If \( u = v \) we have a positive match. Fig. 2 shows the percentage of users that were correctly identified, versus the number of Core and Support domains used in the profile generation phase. For equal user number, the larger is the data, the better is the identification. For low number of both Core and Support domains, it is hard to identify users; in fact, just up to 23.8% were correctly identified with 10 domains.

Core domains are clearly better characterizing and more important for user identification than Support domains, for all the three...
metrics. With only 70 Core domains, the range of correctly identified users is equal to 72-79%. This percentage is higher than the range of correctly identified users when 500 Support domains are used (68-73%). Regarding to the performance of the three similarity metrics, Jaccard performs worst in all the cases; TFIDF has the best results in most of the experiments with Support domains, while MLE performs a slightly better with Core domains.

5.2 Role of observation time

We now focus our attention to study how observation time impacts on profiling and identifying users. The rate for discovering new domains differs from user to user, and we expect that the same number of domains is collected after a different amount of time. For instance, for discovering 50 Support domains it takes, in median, only 11 hours. For Core domains, instead, it takes about 3.5 days. 5 days are needed to obtain 70 Core domains, but only 3 days for 500 Support domains. This is mainly due to the large number of Support domains (see Sec. 4).

Here, we maintain constant the time of observations $\Delta T_1$ and $\Delta T_2$. We consider two consecutive weeks of traffic, taking into account only users that visited at least 20 Core domains per week, avoiding those that either disappeared or generated little traffic. We use the first week for profiling, and the second one for identification. Tab. 2 details results for the three similarity metrics considering Core, Support and All domains. The median number of domains per week per user is reported for completeness. Best results are in bold. We repeat the same experiment considering 2 consecutive days, with results in Tab. 3, where the median number of discovered domains decreases to about the half, and performance also drops. In both cases the large quantity of Support domains, about ten times the number of Core domains, helps in user identification.

As expected, performance decreases when scarce information is available, in particular moving from the two weeks experiment to the two days one. However, still 70% of users can be identified when observing them during one day only. Results of the three metrics are consistent to the ones showed in Fig. 2: Jaccard has always the worst performance, MLE seems less affected by the limited size of Core domains and TFIDF offers the overall best performance except when few Core domains are considered.

To give an intuition about the discriminative power of the profiles, we report in Fig. 3 the Cumulative Distribution Function (CDF) of TFIDF metric between the same user (called self-similarity) and between two different users, considering all the domains from the same two consecutive weeks of Table 2. Self-similarity is almost always higher than the similarity with a different user, thus allowing to correctly identify the target user.

5.3 Longer profiling or identification?

Differently from the previous results, here we consider all users in our dataset, allowing real cases in which users may disappear and generate little traffic. We consider two cases. First, we assume to have i-a) fixed and long profiling time, i.e., 2 weeks for each user, and ii-a) variable observation windows for identify them. Second, we assume to have i-b) a fixed amount of 2 weeks for identification, but ii-b) variable time profiling all the users. Figs. 4a and 4b depict the results for both aforementioned cases. Jaccard considers the two sets independently from the rest of the population; therefore such metric is symmetric with respect to the two sets and depends just on their sizes. This implies that Jaccard performs equally in a) and b) cases, reaching good performance only when the training and testing sets of domains are both large. TFIDF and MLE metrics account for the whole population when profiling, hence having large profiling sets for the population is much more important than having a single large identification set. This is why with a 2 weeks profiling time TFIDF and MLE show good performance even with just few hours of identification time. On the contrary, building profiles with few hours of traffic makes the large amount of data for identification much less useful.

In a nutshell, building better profiles for the whole population is much more important than having a lot of data for identification.
We now repeat the experiments of Tab. 2 and 3 using the ISP trace, where users are possibly more heterogeneous. To compute results from timing, or volume of data. We explored techniques for users' fingerprinting and identification using only the domains of visited web-services. Results show that a simple approach like the TFIDF can be used to solve the identification problem in different scenarios, provided users are profiled for enough time. Web-services intentionally requested have proved to better characterize users.

We expect the probability of identification to decrease with respect to the user population size. Our conclusions could be over-optimistic with larger user-base, and we are planning to repeat the same experiments with larger datasets. We are also focusing on extending this approach to include more features e.g., the information coming from timing, or volume of data.

6 CONCLUSIONS

We explored techniques for users’ fingerprinting and identification using only the domains of visited web-services. Results show that a simple approach like the TFIDF can be used to solve the identification problem in different scenarios, provided users are profiled for enough time. Web-services intentionally requested have proved to better characterize users.

We expect the probability of identification to decrease with respect to the user population size. Our conclusions could be over-optimistic with larger user-base, and we are planning to repeat the same experiments with larger datasets. We are also focusing on extending this approach to include more features e.g., the information coming from timing, or volume of data.

ACKNOWLEDGMENTS

This work has been partially funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, BigDAMA.

REFERENCES


Table 4: Users correctly identified in residential dataset.

<table>
<thead>
<tr>
<th></th>
<th>Median</th>
<th>Jac</th>
<th>MLE</th>
<th>TFIDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 day</td>
<td>556</td>
<td>80.4%</td>
<td>84.1%</td>
<td>86.6%</td>
</tr>
<tr>
<td>1 week</td>
<td>1785</td>
<td>93.6%</td>
<td>93.7%</td>
<td>94.9%</td>
</tr>
</tbody>
</table>