DNS weighted footprints for web browsing analytics

José Luis García-Dorado, Javier Ramos, Miguel Rodríguez, and Javier Aracil

High Performance Computing and Networking Research Group,
Universidad Autónoma de Madrid, Spain.

Abstract

The monetization of the large amount of data that ISPs have of their users is still in early stages. Specifically, the knowledge of the websites that specific users or aggregates of users visit opens new opportunities of business, after the convenient sanitization. However, the construction of accurate DNS-based web-user profiles on large networks is a challenge not only because the requirements that capturing traffic entails, but also given the use of DNS caches, the proliferation of botnets and the complexity of current websites (i.e., when a user visit a website a set of self-triggered DNS queries for banners, from both same company and third parties services, as well for some preloaded and prefetching contents are in place). In this way, we propose to count the intentional visits users make to websites by means of DNS weighted footprints. Such novel approach consists of considering that a website was actively visited if an empirical-estimated fraction of the DNS queries of both the own website and the set of self-triggered websites are found. This approach has been coded in a final system named DNSPRINTS. After its parameterization (i.e., balancing the importance of a website in a footprint with respect to the total set of footprints), we have measured that our proposal is able to identify visits and their durations with false and true positives rates between 2-9% and over 90%, respectively, at throughputs between 800,000 and 1.4 million DNS packets per second in diverse scenarios, thus proving both its refinement and applicability. **Index Terms:** Browsing analytics; Data monetization; DNS footprint; web-user profiles; DNSPRINTS.

1 Introduction

The commercial exploitation of web browsing analytics is currently a fact, whereby the collaboration between both marketing and Internet communities has progressively strengthened after each business success. The information about users' preferences that the different actors in the Internet arena gather can be used not only as inputs for tailored online-ads but for improved and novel mechanisms to monetize the rich set of available data [1, 2], even more with the advent of the Big Data era. In paying attention to web access profiles, it becomes apparent that the identification of a competitor attracting the interest among the customers of a given company is of paramount interest to react accordingly. Similarly, the interest in terms of visits of the population of a given geographical area is a useful input to plan the expansion for many retailers or to assess the impact of an advertising campaign. A real-world example of this is Google AdWords service [3], such that Google exploits keywords, statistics of keywords, and search results of its users [4] and customizes ads according to their profiles.

Nonetheless, the capacity of creating users' web-profiles is not the exclusive preserve of web search engines as of today, but also others, such as ISPs, have the opportunities to exploit such business model. Actually, in a richer fashion given that search engines are only aware of the few first accesses users make to a website. After that, typically, web names are retained in the browsing history (or bookmarked), or simply memorized by the users. This way, search engines are no longer used as gateways to such sites.

Alternatively to client intrusive software and add-ons installed on clients' web browsers, such as toolbars [5] or cookies [6], the ISP approach to this issue is the apparently simple task of capturing and inspecting HTTP traffic. However, two concerns arise. The unstoppable increase of HTTPS and the implicit difficulty of capturing (needless to say, inspecting) large volumes of HTTP traffic potentially distributed in different geographical areas. While the latter remains as an open research area [7], the authors in [8, 9] pointed out that DNS can play a major role to reveal the traffic behind a HTTPS flow. Further, the authors in [10, 11] successfully correlated traffic flows [12] and temporally-close DNS queries, thus identifying IP addresses of users and the name of the accessed hosts. Unfortunately, such approaches still require to collect all HTTP/HTTPS and DNS traffic to construct the flows to which relate the DNS protocol, and, importantly, other collateral issues emerge:

- DNS Cache: Most of DNS resolvers and clients implement a cache for DNS traffic where the associated IP of a given domain name is temporarily stored. As Time To Live (TTL) for the cache entry can be long [13], chances are that a point of presence monitoring traffic cannot see clients' DNS queries although they are effectively visiting a web.

- Automated clients, botnets and worms [14, 15, 16]: Often, DNS queries are not generated by actual users from a web browser, but from scripts that poll DNS servers with single queries searching for updates, information about the service or mail server addresses as well as to perform Distributed Denial of Service (DDoS) attacks.

- The tangled web [10]: Almost any web includes a set of external contents (e.g., resources of other webs of the same company or, simply, banners or ads) whose domain names have to be resolved in order to fully load the requested web itself. This causes that additional DNS traffic is generated without being explicitly requested by users and makes users' web-profiles imprecise, especially for business exploitation.

- Websites' embedded content preload: HTML5[1] recommendation defines special websites' attributes such as `preload` that calls for an early load of resources not being part of either the main website loading or the rendering process. Such resources generate additional DNS queries.

- Link prefetching: Popular browsers such as Google Chrome, Mozilla Firefox or Internet Explorer perform link prefetching which generates non-explicitly requested extra traffic. Prefetching can be performed at different levels. Nowadays almost any browser pre-resolves domains present in every link of a website. Moreover, some browsers may not only resolve link addresses but prefetch cacheable contents from websites referred by such links, if possible. Even more, some browsers, such as Internet Explorer, perform image pre-rendering. Furthermore, websites may also indicate browsers to prefetch certain links, as defined formally in the HTML5 recommendations by means of special attributes of the `link` tag. In sum, all these approaches generate both HTTP/HTTPS and DNS extra traffic that impedes generating precise users' browsing profiles.

In this light, we search for a mechanism that relying on DNS traffic to detect every time a user accesses a website is able to circumvent the problem

[1] https://www.w3.org/TR/html5/

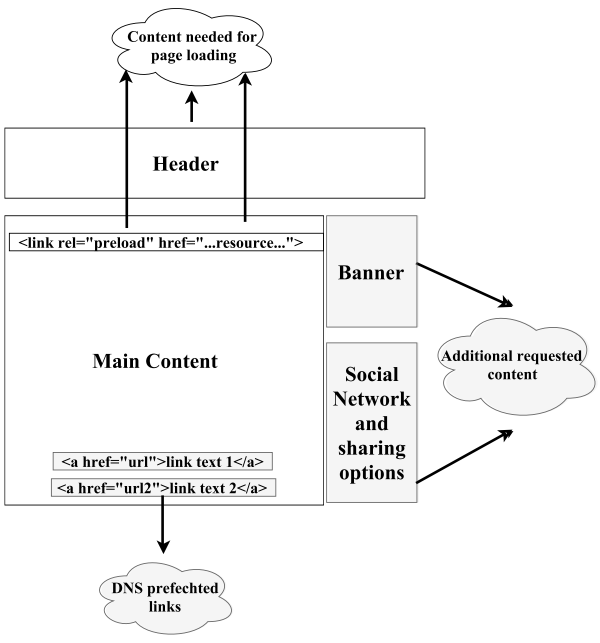

Content needed for page loading
Header
<link rel="preload" href="...resource...">
Banner
Main Content
Social Network and sharing options
Additional requested content
<a href="url">link text 1</a>
<a href="url2">link text 2</a>
DNS prefechted links

Figure 1: Sample website layout, with content that generates self-triggered DNS queries embedded on several banners and external services

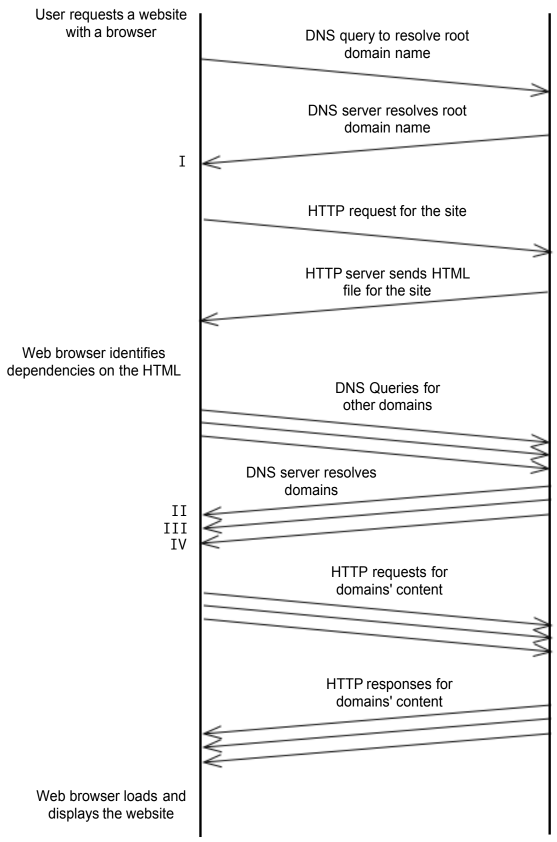

User requests a website with a browser
DNS query to resolve root domain name
DNS server resolves root domain name
I
HTTP request for the site
HTTP server sends HTML file for the site
Web browser identifies dependencies on the HTML
DNS Queries for other domains
DNS server resolves domains
II
III
IV
HTTP requests for domains' content
HTTP responses for domains' content
Web browser loads and displays the website

Figure 2: Flow graph describing the packet flow of DNS and HTTP traffic when a site is requested using a web browser

that some queries are hidden by caches, and, importantly, with the capacity to distinguish between requested website visits and unintentional ones (let us refer to the latter as self-triggered webs or self-triggered traffic). To do so, we propose to explicitly exploit the tangled characteristic of websites and DNS preload/prefetch capabilities of current websites/browsers.

Figure 1 features a sample layout for a website with several embedded ads, banners and links to both external and same company services. After the main website is requested, while users wait for loading, a set of DNS queries/responses are generated. Such behavior is illustrated in Figure 2, where multiple resolution processes are in place. The set of different resolved domains that a given website triggers, defines the DNS footprint of such a website. This way, given a list of root domain websites, i.e., those websites of interest for business purposes, we construct their associated DNS footprints that will be used to search in traffic aggregates to determine the visits per user, potentially sanitized, or per aggregation in market segments and geographical areas.

In fact, being less restrictive with the footprint (i.e., accept a match although not all the websites of a footprint are found) circumvents the cache problem while being more restrictive rules out self-triggered visits, in other words "false" resolutions with respect to the actual website requested by the user. Additionally, we remark that not all websites present in a footprint must be considered at same level. If a website appears regularly in many footprints, its capacity of discrimination is limited, and the opposite for uncommon websites. Consequently, we propose to weight the importance of a website for footprints construction by its number of occurrences in the full set of constructed footprints. How to assign weights is studied along this paper.

The proposed solution has been implemented as a final system called DNSPRINTS. We have applied it to a significant set of websites and diverse cross traffic during different periods of time, as well as to diverse scenarios of operating systems, browsers and devices. The accuracy evaluation of the methodology has shown false positive rates between 2-9% and true positive rates between 90-95% depending on the scenario, thus proving the potential of DNSPRINTS.

Significant results were also found on the performance evaluation of the system implementation, which is able to handle between 800,000 and 1.4 million DNS packets per second in realistic scenarios including traces from a large National Operator, thus proving the applicability of DNSPRINTS in the most challenging networks.

The following section describes the visit-detection methodology used by DNSPRINTS system. Next, the parameterization and accuracy of the methodology receive attention in Section 3, while the final system architecture, implementation as well as performance are detailed in Section 4. In Section 5, previous works are analyzed pointing out the main difference with our proposal. Finally, some conclusions and future work lines are given in Section 6.

2 DNSPRINTS' methodology

This section describes our method to identify user browsing based on DNS traffic. The typical use case is that of a marketing department willing to analyze customers' behavior in a given market segment, for instance car manufacturers. To this end, the department provides a set of websites of interest for the analysis, namely those of the relevant car manufacturers. Our purpose is detecting when a user has requested a certain website using a browser, out of such set of websites, only by inspecting the DNS resolutions generated.

2.1 DNS footprints

Let $D = \{d_1, d_2 \dots d_N\}$ be the set of websites (typically, root domains) to monitor. As we stated before, when a user requests a website d_i, several resolutions are self-triggered in addition to the domain name explicitly requested. All these resolved domain names (M) make up a set of domains that define the DNS footprint F_i for that website d_i, being $F_i = \{d_i^1, d_i^2 \dots d_i^{M_i}\}$, $i = 1, 2, \dots N$. After a domain name d_i^j has been resolved, clients usually store that information on a DNS cache for a certain time T_i^j suggested by the DNS server.

The time each domain is stored on the DNS cache is called TTL and varies significantly depending on the domain. Typically, root domains present a long TTL (e.g., TTL>1000 seconds) and many of the subsequent domain resolutions have shorter ones (e.g., TTL<30 seconds) [17]. As a typical behavior, Figure 3 shows the different values that T_i^j takes in the set of domains resolved while we visited a popular news website. Such values present a very wide range, from a few seconds to several hours, with the root domain DNS cache entry lasting for more than an hour. Accordingly, if a user reads the news every few minutes, a standard DNS monitoring tool, inspecting root-domain requests, would only detect one request to the root domain every several hours.

Alternatively, while not all the domains from F_i will be resolved every time the root domain d_i is visited, we observe that most of them do it. Note that domains with a low value for T_i^j will be resolved more frequently as they expire sooner from the cache and must be updated regularly. That is, after a DNS server receives a query for a root domain d_i^j, it is likely that it will not be queried again by this user until its DNS cache entry expires but a significant fraction of the footprint will. This way, the accuracy does not depend on the TTL value that a DNS response announces for the requested root domain but on the lowest TTL values of all

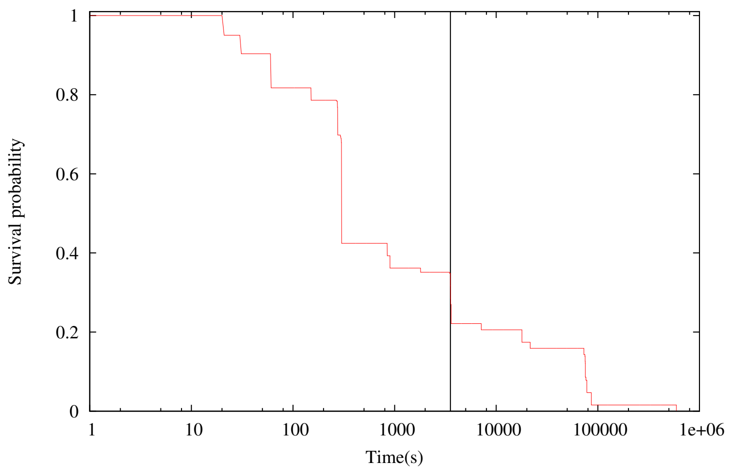


Figure 3: Distribution of the TTLs on DNS responses generated by visiting a popular news website. The vertical line represents the TTL of the root domain

the domains making up footprint F_i. This allows us to improve the detection resolution from several hours to a few minutes or even seconds in the best case.

2.2 Weighted footprints

After visual inspection of the footprints, it became apparent that not every domain name d_i^j is equally representative. For example, www.google.com or www.facebook.com will be present in the footprint of many websites due to ads or social media banners. In other words, resolving www.google.com does not provide much information about which website is being visited and, therefore, its relevance should be low. On the other hand, resolving www.cnn.com is not so common and it will probably appear in fewer footprints, hence giving much more information that the former.

This way, for every domain name d_i^j presented in a footprint a weight w_i^j is assigned. Such weight is inversely proportional to the number of footprints that contain such domain name and then normalized in a way that the sum of all weights is equal to 1. The inversely proportional weighting process is performed to give more importance to the most relevant entries in the footprint and depreciate the importance of domains present in multiple footprints. More formally, the following equation shows how the weight is calculated:

$$\tilde{w}_i^j = \frac{1}{\left|\{x : d_i^j \in F_x\}\right|} \qquad w_i^j = \frac{\tilde{w}_i^j}{\sum\limits_{k=1}^{N} \tilde{w}_i^k} \tag{1}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, M_i$.

2.3 How to consider a visit occurs

To determine whether a user is visiting a certain monitored website d_i at a certain moment, all the weights w_i^j of the domains in F_i that are currently in this user's cache are added. Ideally, this sum would be one per each requested website but it will not be due to the use of caches. However, intuitively, the higher the current sum of the weights the most likely it is that the user is currently browsing such a website. When the user stops browsing the website, the corresponding DNS cache entries will start expiring and therefore the amount of domains cached and the sum of their weights will drop. Using this mechanism we can define the time series $S_i(t), t > 0$ for each website d_i and user that measures the likelihood that a user being currently visiting the website d_i. $S_i(t)$ is calculated at a given point in time t as shown in the following equation:

$$S_i(t) = \sum_{d_i^j \textit{ is cached at time } t} w_i^j$$

Then, we define a certain threshold α that $S_i(t)$ has to exceed in order to mark a website as visited. Whenever $S_i(t)$ surpasses the given threshold α, a visit for that website is counted. In other words, α represents the fraction of weight footprint captured to consider that a visit took place although some domains of the footprint were not present. This is the key of our method as precisely the root domain, because of its high TTL, is the most likely unresolved domain. On the other hand, we consider the finalization of the visit when $S_i(t)$ drops below α.

The accuracy of the method is directly related to the value of α. A low value for α increases both the number of false positives (a website is considered as visited but the user did not request it) and the time

$$\begin{array}{c|cccc|c}
 & d_1 & d_2 & \dots & d_N & \text{TTL} \\
d_1^1 & 0.35 & \dots & \dots & \dots & 178 \\
d_1^2 & 0.15 & \dots & \dots & \dots & 60 \\
d_1^3 & 0.2 & \dots & \dots & \dots & 65 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
d_1^{M_1} & 0.1 & \dots & \dots & \dots & 6000 \\
d_2^1 & 0 & \dots & \dots & \dots & 300 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
d_N^{M_*} & \dots & \dots & \dots & \omega_N^{M_*} & TTL_N^{M_*} \\
\text{sum} & 1 & 1 & \dots & 1 &
\end{array}$$

Figure 4: DNS footprints matrix example

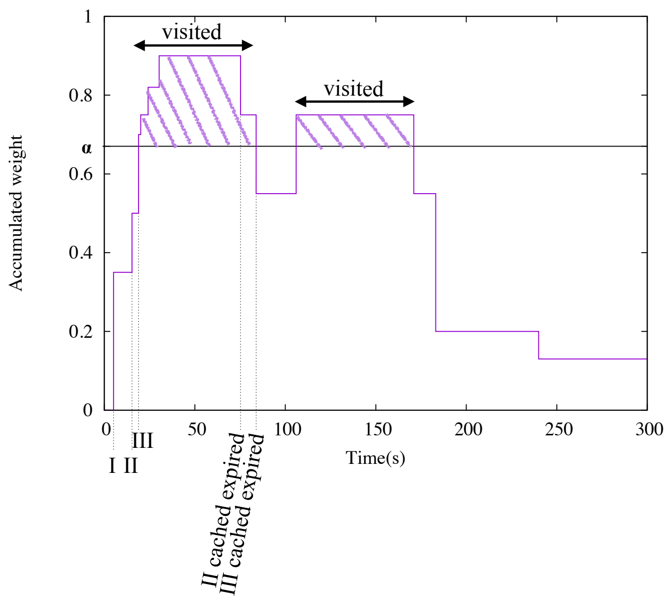


Figure 5: Temporal evolution in the detection of visits to the website d_1

resolution of the method (whenever a website is visited, it is necessary to wait until the weights sum of such website's footprint drops below α). Whereas, a higher value for α will improve the time resolution but also the probability of missing an actual website request from a user.

2.4 Numerical examples

Let us illustrate the operation with examples. Figure 4 represents, in a matrix form, a footprint set where each cell represents a weight and each column corresponds to a root domain (d_1 to d_N). The rows correspond to the union set of all resolved domains (all the different domains observed after resolving the root domains). Then, Figure 5 illustrates the temporal evolution of the detection of visits to the website corresponding to d_1 column of Figure 4.

We note that when accessing to website d_1, the root domain d_1^1 (I in the latter figure and also in Figure 2) presents a weight of 0.35 and a TTL of 178 seconds. At this moment the accumulated weight is 0.35. After resolving the root domain, a domain (II in both figures) is resolved. Such domain presents a weight of 0.15 and TTL of 60 seconds. At this moment, the accumulated weight is 0.50. Next, a second domain (III in the figures) is resolved with a weight of 0.2 and a TTL of 65. After such resolution the accumulated weight is 0.7, which surpasses the predefined α and d_1 is marked as visited. Finally, TTLs of domains II and III expire and the visit is considered as finished.

Certainly, the website d_1 can be visited again after the first visit. If this occurs after domains II and III are expired from cache, they are resolved again and the accumulated weight will surpass again α marking a second visit for website d_1 as the figure shows. If this occurs before entries expire, the second visit will be incorrectly considered as a part of a longer first visit. Note, again, that if we considered the duration of a visit to be the TTL of the root domain this issue would occur frequently (root domains' TTL are typically larger than the time users stay in a website). Interestingly, we limit the cache influence to the lowest TTLs of the website's footprint (typically, shorter than the user session) so making this issue far less likely.

The previous examples have shown how a single website is monitored. In a real production scenario, DNSPRINTS monitors several websites in parallel. Figure 6 shows the values of $S_i(t)$ over time for a sample a user session. Each line represents the sum of the weights w_i^j of the cached domains of a websites d_i during the session. Whenever a domain is visited, the sum of the weights starts increasing. But after a given time, namely the TTL values, the sum drops as the DNS cache entries start expiring. According to when each S_i exceeds and, then, falls

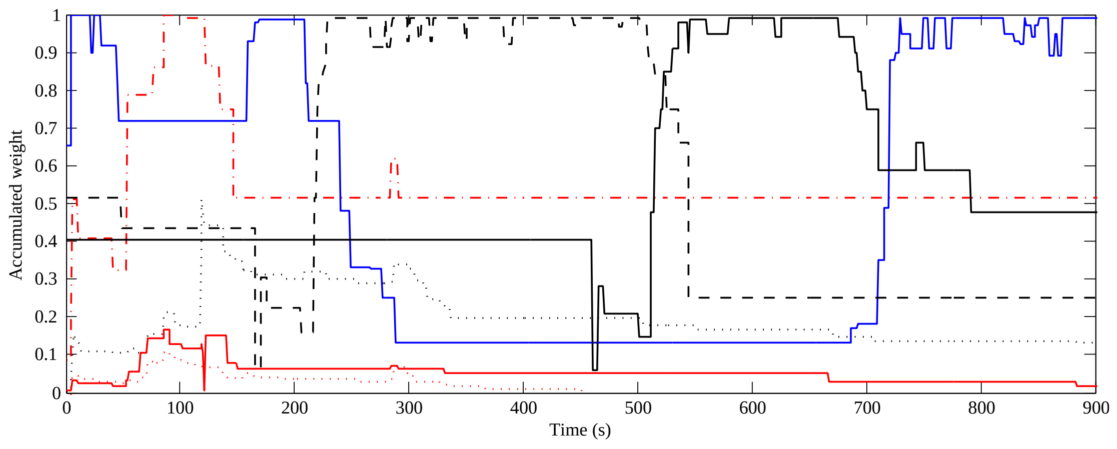


Figure 6: Different footprints likelihood ($S_i(t)$) during a user browsing session each represented with different colors and line patterns

below the threshold α, visits will be recorded.

2.5 Environment diversity

Another important aspect to consider is that self-triggered resolutions for a given domain vary with time as websites layouts do and that, potentially, such resolutions can also differ between browsers or, even operating systems. Similarly, most websites present a different layout, or even a different domain name (after a redirection), when accessed from a mobile device. Let us refer to such characteristics that can impact on websites' footprints as factors.

One option to deal with this diversity is to construct a different footprint per factor or, even, set of factors. Another option is to construct final footprints as a partial intersection from two or more partial footprints (i.e., footprints constructed for a specific browser and operating system). In other words, having n partial footprints of a website, a domain is added to the final footprint if it appears in at least $n \cdot \beta$ of them.

On the one hand, the β parameter should be tuned to maximize the robustness of the footprint. Having $\beta \sim 0$ would turn the partial intersection into a union, and the final footprint would not be resilient to minor changes or variations in the website. On the other hand, having $\beta \sim 1$ would turn the partial intersection into a strict intersection, greatly shrinking footprints size, and the footprint would become even empty if partial footprints are significantly different.

The use of different footprints or partial intersection for any of the factors previously mentioned is configurable in DNSPRINTS. The next section will provide empirical insights and illustrative figures for such parameters.

3 Footprint inspection, design and methodology accuracy

We first inspect the impact that several external factors exert in the construction of footprints. With the resulting conclusions, we perform the evaluation of DNSPRINTS methodology in terms of accuracy. To do both so, we first construct a controlled dataset in terms of factors and visited websites as ground truth. That is, a set of traces labeled with the actual intentional visits on a diverse set of scenarios for significant periods of time.

3.1 Ground truth dataset

We developed a robot for simulating web browsing based on Selenium framework [18] configured to use Google's public DNS server. Such robot simulates a user session where, given a set of websites, it randomly visits the websites of such set (with certain bias towards previously visited ones, i.e., revisitation factor), clicking updates and staying as much time in each visit as parameterized. Then, the robot resets itself and simulates another user session. It gathers all the generated and received DNS traffic as a PCAP [19] trace and outputs textual records of each intentionally visited website. The number of websites a user visits in a session, clicks, and the duration of each visit are exponentially distributed. In the generated dataset, the robot was specifically parameterized with averages of 15 visits per session with revisitation factor of 0.5 [20], 5 clicks per website and durations of 120 seconds [21] per visualized web. For the sake of completeness and comparison, the robot's sources are publicly available at [22].

To construct the dataset the closest possible to a real user freely browsing the Internet, we have chosen the list of websites as the union of sets of potential clients of DNSprints system, and the most popular websites both internationally and nationwide. In more detail, according to Alexa rankings [23], the list comprises:

1. Top 10/20 nationwide in the following categories, as candidates to clients for DNSprints system: radio stations, television channels, newspapers, magazines, mobile providers, retail establishment/supermarkets, universities, banks, car manufactures and airlines. We believe these sorts of companies are likely to benefit from browsing analytics. Besides, the former set is nationwide as we believe the study is especially relevant at this level. For example, comparing users' dynamics of Ford UK and Toyota UK websites makes more sense than comparing one of them with a website of a car manufacturer in Japan.

2. Websites to add diversity to user browsing:
   (a) The top 50 visited websites worldwide.
   (b) Other websites to complete the nationwide top 250.

In summary, the set of websites to monitor, previously named as D, are those comprising the first subset. The other websites cover the typical user behavior and, eventually, help when measuring DNSprints noise tolerance. In this latter regard, note that some of the considered worldwide popular websites can be unpopular at a specific nationwide scope (e.g., some Chinese or Russian websites can be popular in aggregate terms in the world but be marginally visited in Spain).

The robot was executed during two months using different combinations of operating systems (Windows 10 and Ubuntu 16.04), web browsers (Mozilla Firefox and Google Chrome) and desktop/mobile environments (standard PC and Nexus 5). This way, we constructed a set of ground truth records $G = \{G_i = (u_i, d_i, t_{0i}, t_{1i}, o, b, e)\}$, $i = 1, 2, \dots M_G$, which is made up by M_G tuples (number of websites actually visited along the experiment) that contain the user identifier (u_i) (in this case, "robot"), website visited (d_i), initial time (t_{0i}) and finishing time (t_{1i}) registering the operating system (o), web browser (b) and environment (e) used. In addition to this, a different PCAP, P_{obe} trace, is gathered for each combination.

3.2 Impact of factors on footprints

Let us first elaborate on the resulting weighted footprints after applying DNSprints methodology on the ground truth dataset. Our first concern is if footprints are different, and to what extent, between different environments and over time, or, conversely, if we can ignore this and construct a homogeneous set of footprints which, intuitively, is simpler.

To quantify how a given factor impacts on footprints' construction, we entrust the task of summarizing the differences between two footprints to Jaccard distance [24]. Formally, Jaccard distance measures dissimilarity between finite sample sets as the complementary of the size of the intersection divided by the size of the union of the sample sets:

$$Jaccard(S1, S2) = 1 - \frac{(S1 \cap S2)}{(S1 \cup S2)}$$

where S1 and S2 are two sets.

We translate this definition to our context [25], where the members of each set are the domains that make up each footprint and where such members belong to the set, not binarily, but proportional to their weights (in the range $[0, 1]$):

$$Jaccard(F1, F2) = 1 - \frac{\sum_{x \in F_1 \cap F_2} min(w_1^x, w_2^x)}{\sum_{x \in F_1 \cup F_2} max(w_1^x, w_2^x)}$$

where two footprints are considered, $F1$ and $F2$, and w_i^x represent the weight of the domain x in the footprint i ($i = 1, 2$), regardless the order in which x appears in such footprint.

This provides a distance bounded in the $[0, 1]$ interval for each pair of footprints and allow us to ascertain if given two footprints the set of domains of each one are similar or not to the other. Intuitively, the distance gives a hint of how domain weights differ. As an example, a distance of 0.1 implies that the weights of the domains only differ 10% in average, conversely 0.9 means that such weights differ 90% in average.

Next, let us calculate the distance between footprint outputs per each factor under study.

3.2.1 Factor time

Websites update their contents periodically so it is expected that their corresponding DNS footprints suffer changes. This would suggest short sampling periods to capture the variability, but at same time, such short sampling period would make footprints lack generality.

To measure such periods, we construct different sets of footprints varying the sampling period to one day, one week and one month, holding constant other factors. For example, we construct footprints considering the traffic of one day, and then another set of footprints for the traffic of the following day, and so on for all days in the dataset. Finally, we compare such sets.

The distribution of the distances for the set of footprints under study after comparing pairs of consecutive days, pairs of consecutive weeks and the two months we measured are depicted in Figure 7 as histograms. The results show that variations in the intervals of day and week are limited whereas the opposite holds for one month intervals. Specifically, in the cases of one day and week periods more than 85% and 75% of the footprints depict a distance below 25%, respectively, while the same distance is only achieved by less than 50% of the footprints from one month to the following.

This reflects that footprints constructed in one month may differ significantly from the following month, so the footprint constructions process should be repeated with higher frequency. On the other hand, we find no significant differences between those constructed in one week and the following, and importantly, such differences are similar to the differences showed when compared footprints daily. These results point to construct footprints on week-interval aggregates as a good trade-off between variability and generality.

3.2.2 Factors operating system, browser and Mobile/Desktop

Figure 8(a) shows that the variation of the footprint considering different operating system is marginal, whereas the difference between browsers is noticeable (Figure 8(b)). That is, while in the former only 20% of the domains weights vary more than 10%, this amount increases to almost 50% in the latter. Although the variation is limited, note that only 20% of the footprints show difference over 50%. This suggests that the operating system is not a significant factor while the browser is a moderate one.

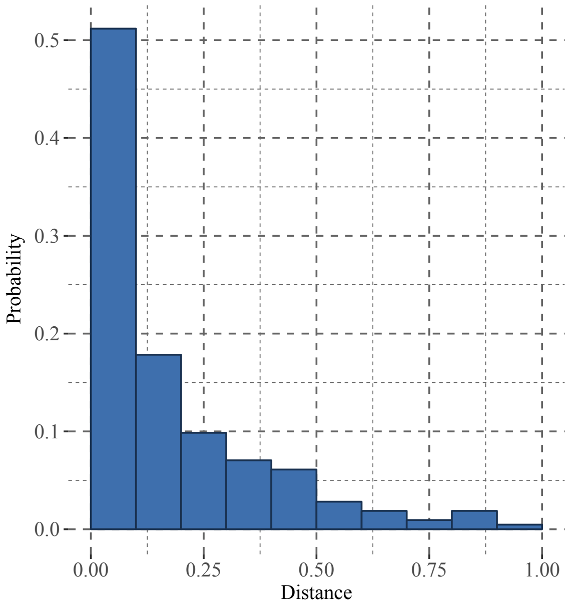
Finally, Figure 8(c) shows the comparison between footprints constructed by the robot in either a desktop PC or a mobile phone. The significant differences between footprints become apparent in all the histogram. For example, only 15% of the footprints can be considered equal, less than a half similar ($\leq$25% variation in weights) and 10% of the footprints only appears in one of the sets (i.e., distance equal to 1). Consequently, we consider that mobile and desktop factor is significant.

3.2.3 Design implications

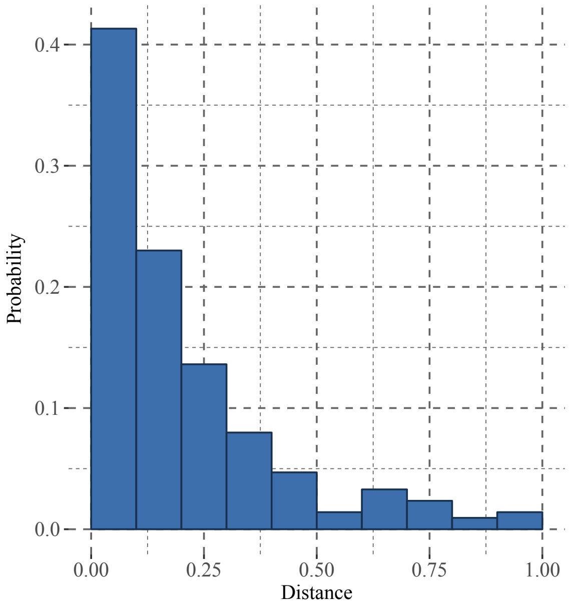
As a conclusion, in order to construct footprints, we recommend not to consider the factor operating system beyond this point. Due to the moderate impact of the factor browser, we will apply a soft intersection between footprints of different browsers, using β parameter (Section 2.5).

Finally, desktop and mobile environments have proven to be so different that we consider constructing one different footprint per website and environment.

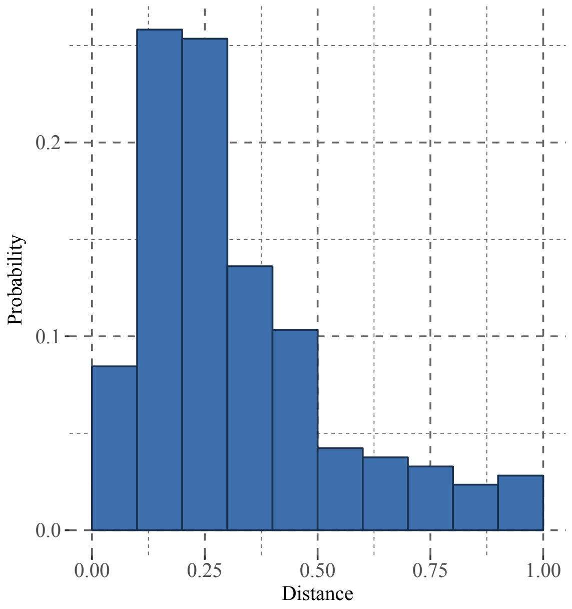
Anyway, the following section that describes DNSPRINTS in terms of architecture will show how these decisions can be made by network managers according to their experience, although we believe that the above experimental results can guide them in this process.



(a) Day.

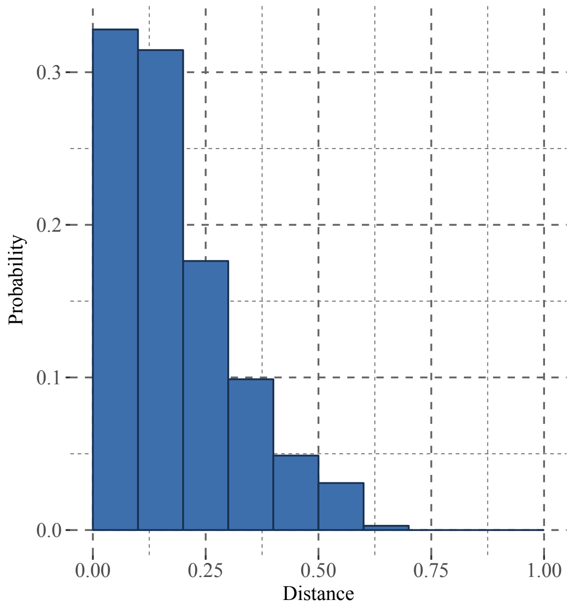


(b) Week.

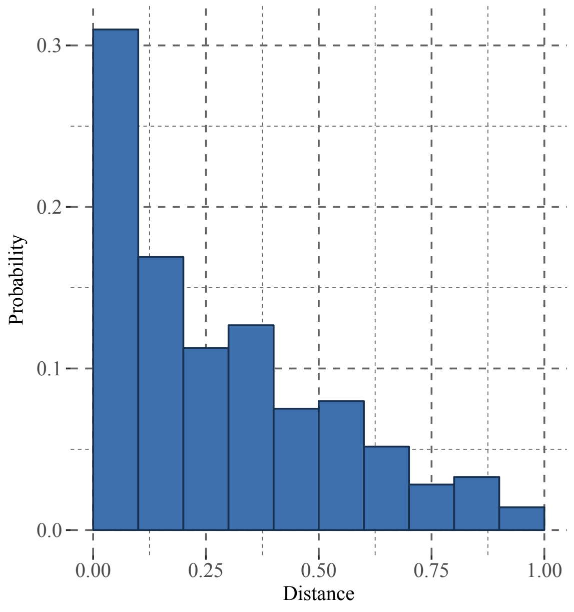


(c) Month.

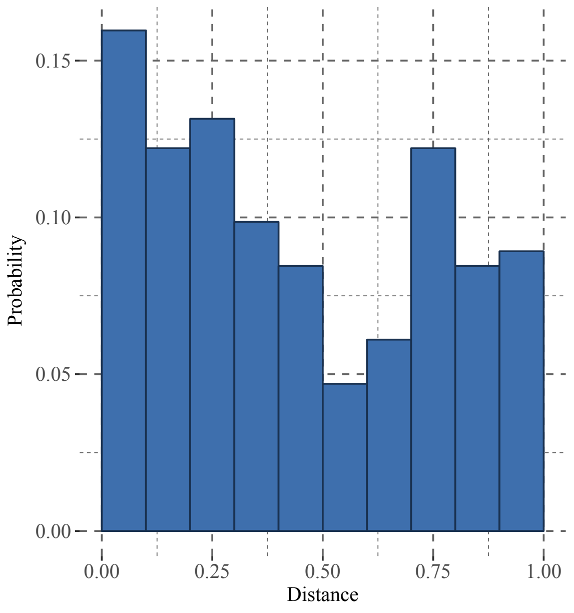
Figure 7: Distribution of Jaccard distance among footprints constructed daily, weekly or monthly



(a) Operating System.



(b) Browsers.



(c) Mobile/Desktop.

Figure 8: Distribution of Jaccard distance for footprints constructed in different operating systems, browsers and mobile/desktop, respectively

3.3 Accuracy evaluation

Now, we compare the ground truth (i.e., robot records) and the classification that DNSPRINTS makes once footprints are constructed according to the above design principles. To sum up, an aggregated duration of one week, β parameter will span the diversity that browsers account, and we will have two footprint versions, mobile and desktop, for websites.

3.3.1 How to evaluate

We applied DNSPRINTS over the full set of captured traffic (i.e., traces P_{obe}) for all different values of α and β in range [0-1] at 0.01 steps. Then, we constructed a set of estimated visits $H^{\alpha,\beta}$ analogous to the previous ground truth set (G) per combination of α and β, i.e., $H^{\alpha,\beta} = \{H_i = (u_i, d_i, t_{0i}, t_{1i})\}$, $i = 1, 2, \ldots M_H$, with M_H being the number of estimated visits.

For each of these sets, the ground truth and the estimations depending on α and β, we constructed time series per user session and website. Such time series are vectors in which each position of the vector states if the website was visited or not by the user. Moreover, we inspect the width of each vector cell, i.e. how much time it represents. Note that small scales (or steps), such as 1 second, allow us to evaluate the accuracy for future analysis that require low-scale precision (e.g., the impact of a TV advertisement). Other larger scale, for example one day, may also provide interesting results (e.g., popularity of a website as unique daily users) but information about how many and how long the visits lasted is lost. Specifically, we have considered steps of 1 second, 5 minutes, 1 hour, 6 hours and 1 day.

In more detail, given a user session and a website, we construct a vector where each position will be marked (otherwise, remains unmarked) if such website was being visited in the time interval the position represents according to the initial time and final time that G and $H^{\alpha,\beta}$ states. Let us define vectors $\overrightarrow{G}_{u,i}^{step}$ and $\overrightarrow{H}_{s,i}^{\alpha,\beta,step}$, with u identifying each user session, i the website, and *step* the particular scale.

Figure 9 illustrates how both vectors are constructed. Specifically, at the top the figure the actual duration of some visits to website example.com are shown, in the example two visits occurred marked by arrows. Then, $\overrightarrow{G}_{U_1,example.com}^{step}$ is constructed for steps equal to 1 day and 1 hour for illustrative purposes. Essentially, the cells that totally or partially overlap to actual visits are marked. In the middle of the figure, the process is analogous but for the estimations. Again, the durations (in this case estimated by DNSPRINTS) are used to construct visit vectors.

Then, to evaluate the accuracy, we calculated the True Positive Rate (TPR) or sensitivity/recall for a website and session as $TPR = TP/(TP + FN)$, where TP is the number of true positives and FN the number of false negatives. In other words, the ratio between the number of cells that $\overrightarrow{G}_{u,i}^{step}$ and $\overrightarrow{H}_{u,i}^{\alpha,\beta,step}$ have both marked and the total number of cells marked in $\overrightarrow{G}_{u,i}^{step}$.

At the bottom of Figure 9, these values are visually shown for both day and hour scales, for the visits to example.com. As an example, we will have a TPR for the estimation of 7/10 in the case of 1-hour scale, and 1 for 1-day scale.

Similarly, the False Positive Rate (FPR), or 1-specificity, is defined as $FP/(FP + TN)$, or intuitively as the ratio between the number of cells that are marked in $\overrightarrow{H}_{s,u}^{\alpha,\beta,step}$ but are not marked in $\overrightarrow{G}_{s,u}^{step}$, and the total cells not marked in $\overrightarrow{G}_{s,u}^{\alpha,\beta,step}$. Turning to the example of Figure 9, FPR is 3/14 and 1 for 1-hour and 1-day scale, respectively.

Using these rates, we build a ROC curve for each combination of the parameters α and β to illustrate the accuracy of the method.

3.3.2 Parameters α and β

Let us first pay attention only to the impact of β in the accuracy, and focus in the 5-minute scale. Figure 10 shows as a ROC curve where each point represents the TPR and FPR for a pair of α and β values (zoomed on FPRs below 0.2 and TPRs over 0.8). Specifically, we show β for 0.1, 0.3, 0.5, 0.7, 0.9 and 0.95, as the legend indicates, while α ranges between 0.01 and 1 at 0.01 steps. The results indicate that high values of β provide the closest points to the top left corner, while values of 0.5 and lower provide worse results. However, we note that the difference between values is limited. For high β values (namely, between 0.7 and 0.95) the differ-

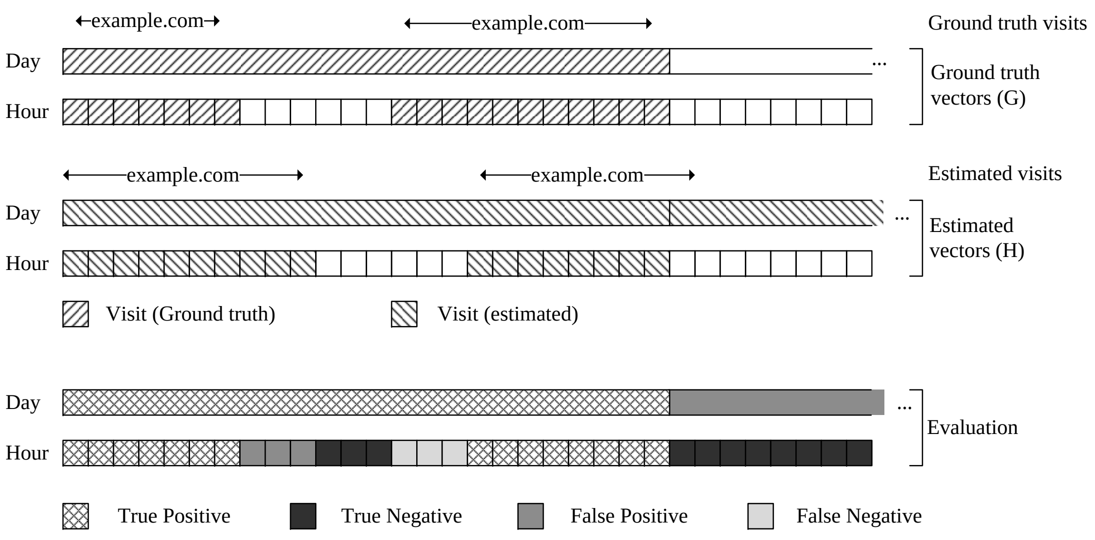
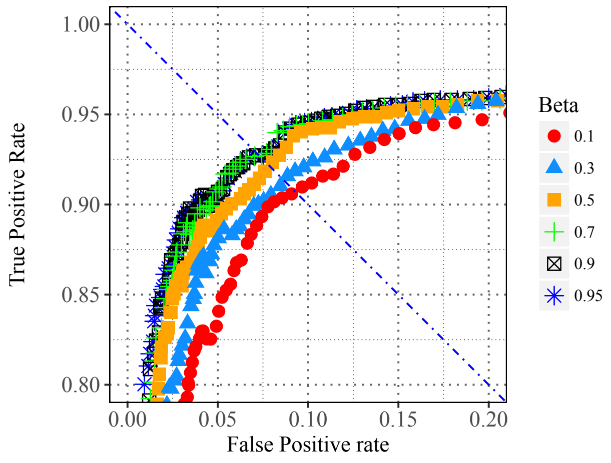


Figure 9: Example for the classification rates used on accuracy evaluation for a given user session and website example.com



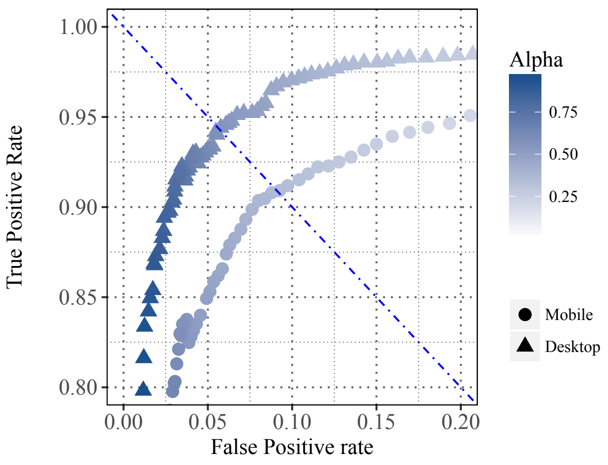


Figure 10: Analysis of β for 5-minute granularity for an extensive range of α values

Figure 11: Analysis of α for 5-minute granularity and $beta$=0.9

ence is marginal, and only show some significance compared to the lowest β values (0.1 or 0.3). As a conclusion, the sensitivity to β is low, roughly $\leq 5\%$, but it becomes apparent that by using high values the accuracy peaks. Let us assume in what follows a β of 0.9 which provides a TPR of 0.93 and FPR of 0.07 for a certain value of α, which receives now our attention.

Figure 11 displays the accuracy for the range of α values, by means of a color scale, separated by desktop and mobile environments. Three observations arise: First, we note that the ideal value (top left corner) for alpha depends on the environment considered. The value of α to choose in the case of desktop visits is around 0.75 while it is around 0.5 in the case of mobile visits. As α works as a threshold to define the fraction of a footprint detected to mark a visit, this shows that DNSPRINTS have to be less demanding for mobile footprints.

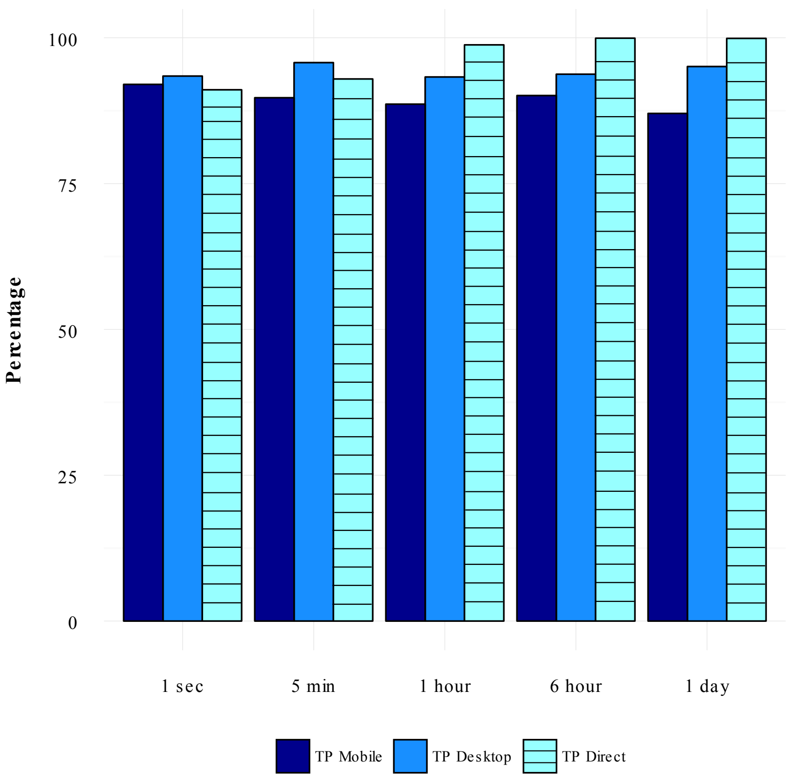
Second, the accuracy is worse in the mobile environment. We believe that the rationale behind these two results are a less rich set of self-triggered traffic in mobile websites, which precisely DNSPRINTS methodology exploits.

Finally, we can conclude that the methodology of DNSPRINTS achieves, for 5-minutes steps, rates of TPR=0.95 and FPR=0.05 in desktop traffic. For the sake of completeness, these metrics translate into a precision of 0.92, an accuracy of 0.94 and, finally, a F-score of 0.94. Regarding mobile traffic, rates of TPR$\geq$0.9 and FPR$\leq$0.1 are achieved, while the rest of the metrics take on values of 0.88, 0.92 and 0.9, for the precision, accuracy and F-score respectively.

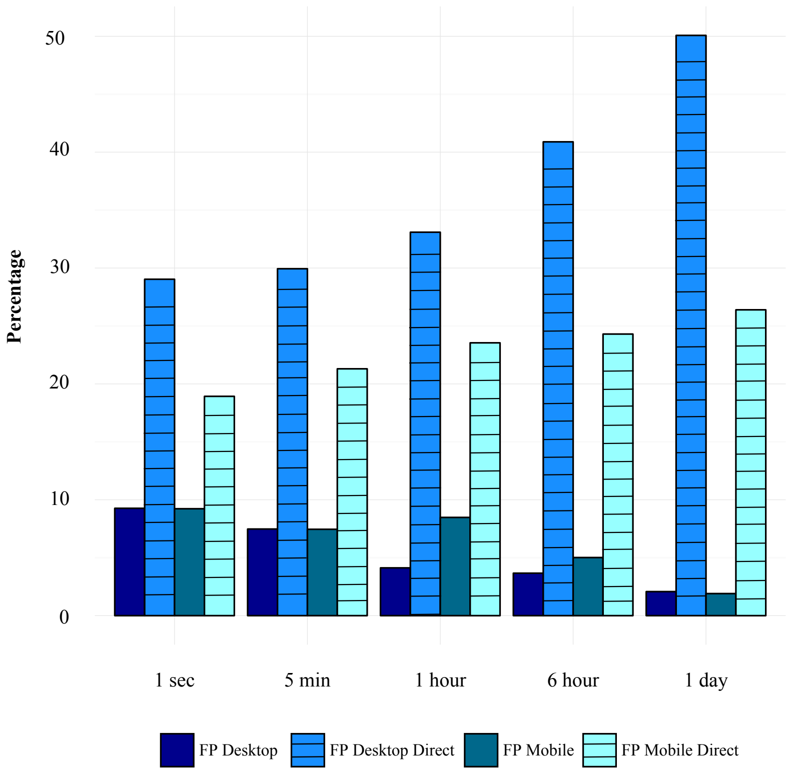
3.3.3 Accuracy

We show the accuracy of DNSPRINTS methodology for different steps assuming parameters β set to 0.9 and α equal to 0.75 and 0.5 for desktop and mobile footprints, respectively. Alternatively, for comparison purposes, we have considered a simpler mechanism to assess that a visit occurs. Let us refer to it as Direct method which consists of considering that a website was visited if the request for its root-domain is found in the DNS traffic. Then, the duration of the visit is considered to be the TTL of the DNS response of such root-domain.

The results are depicted in Figure 12 both TPR (a) and FPR (b) where the plain-color bars rep-



(a) TPR



(b) FPR

Figure 12: FP and TP rates for desktop and mobile environments for both DNSPRINTS methodology and direct method according to different granularities

resent DNSPRINTS results. Note that that mobile/desktop environments are shown separately when differences are significant (i.e., TPRs for direct method were indistinguishable). The values that are shown for all the cases are in the intersection of ROC curve to the top left bisector (straight line [0,1]-[1,0]).

There are four main sources of imprecision in both approaches:

- Overestimate visit durations
- Underestimate visit durations
- Marking visits that did not occur
- Failing to mark visits that actually occurred

True positive rates will be high in classification methods that do not underestimate duration and that do not fail to mark visit. Note that this can happen to an extent that a mechanism that would claim that each website of the list of websites to monitor was constantly visited will present a TPR of 1, as no visit escapes its detection. However, this would be at the expense of overestimating visit duration and considering visits that did not occur, and so high FPRs.

On the other hand, false positive rates will be low in systems that do not overestimate durations, that do not mark as visited websites that were not, and that do not consider unintentional visits.

In this light, with respect to DNSPRINTS and Direct approaches, we note that:

- Direct approach will often overestimate the duration of the visit as it considers that visits last as much the root-domain TTL states, and this is very unlikely to happen (Section 2.1 explained that root-domains TTLs tend to be very long). Conversely, DNSPRINTS rarely overestimates as the duration is not based on TTL of root domain, but in the shortest ones that make up a footprint.
- Regarding underestimation of durations, both approaches may underestimate some time for a website if, while a user is visiting a website, the root-domain TTL finishes (Direct) or the sum of domain weights goes below α threshold (DNSPRINTS). However, the frequency with which this occurs is different between approaches. For this to happen in Direct method, sessions must be longer than root-domain TTL which is again very unlikely. But in DNSPRINTS, underestimation may not be so uncommon. For example, a footprint with many short TTL domains (e.g., 10 seconds) may make the sum of weights fall below α before a user finishes visiting the website (e.g., 1 minute). However, this TTL distribution is not the case for the majority of the footprints and, in any case, the imprecision is limited to some seconds whereas the imprecision of the overestimation of Direct method is typically in the range of minutes or even hours.
- DNSPRINTS may mark a visit that did not occur if two or more footprints of the list of websites to monitor are essentially the same. However, Direct method will suffer from all problems pointed out in the introduction: tangled web, preload and prefetching of contents, etc.
- DNSPRINTS may fail to mark a visit if α threshold is higher than it should be for a certain website. Direct method does not miss visits as if a session goes further its TTL, a new DNS query will be triggered and the session duration extended.

As a conclusion, from a qualitatively standpoint, DNSPRINTS tends to underestimate durations and some website visits may escape, and Direct method tends to overestimate and mark websites that were not visited.

From Figure 12, it stands out that while quantitatively both methods give good TPR rates, i.e. DNSPRINTS barely underestimates and few visits escape, Direct approach shows very poor FPR (to the extent that rates typically are found over 20%), i.e. its overestimation and the number of websites wrongly marked are very significant. Actually, note that DNSPRINTS results in terms of TPR are fairly close to those from Direct. In the case of the smallest granularities, 1 second and 5 minutes, DNSPRINTS achieves equivalent results and, in other scales, the difference is only noticeable by a few percentage points.

Moreover, note that the FPR of Direct method is especially high for desktop version compared to mobile. The rationale behind this is that mobile websites have less external links and preload/prefetching is less common. Next, Direct approach increases FPR as the scale increases, this is because overestimating, in large scales, can make that a session of only some minutes, but incorrectly estimated in hours, extends over the entire following day. However, DNSPRINTS tends to capture length durations more tightly, even underestimate, and then this effect is the opposite.

To conclude, after paying attention to specific digits, the relevance of DNSPRINTS methodology becomes apparent. FPRs are bounded for the worst case of 9% and 1% in the best case, and TPR are over 90% in all cases and up to 95% in the best case. Regarding other metrics, precision ranges between 0.92 and 0.87, accuracy between 0.95 and 0.91 with F-scores between 0.94 and 0.9 for the best and worst cases, respectively.

4 System architecture and performance assessment

While we believe the previous results have highlighted the potential of the DNSPRINTS methodology, let us now describe its translation into a final system and its processing performance.

4.1 Architecture and implementation

DNSPRINTS system may be co-located with a DNS server itself, or in a traffic probe that captures the traffic by means of a Switched Port Analyzer (SPAN), splitter, or traffic collector [26]. In addition, the system has as inputs a list or lists of websites to monitor, a list of other websites to add diversity in the training phase and, optionally, a list of user identifiers to monitor, typically IP addresses (or further details), otherwise all IP addresses would be considered as users to study.

DNSPRINTS architecture is divided into three main modules as Figure 13 shows, let us detail them.

4.1.1 Robot and footprint construction module

This module is an extension to the previously introduced robot (Section 3.1) that has been enhanced to automatically visit websites in different operating systems, browsers and environments thanks to virtualization.

In this way, a diverse set of virtual machines generate visits to the sets of websites to monitor and diversity websites, recording each intentioned visit as well as its duration. With these ground truth records and the DNS traffic the module estimates on-line the ideal values for α (desktop and mobile) and β parameters (as stated before, intersection of bisector and ROC curve). With these parameters, the weighted footprints are constructed as explained in Section 2.2, the sources are available upon request at [27]. Finally, such set of footprints in addition to α parameters will be used by the web-profile monitor module to detect visits.

4.1.2 Online web-profile monitoring module

This module is in charge of applying the weighted footprint to the traffic, and preparing the output for the analysis module. The outputs are records (formally, H in Section 3.3) that state start and end times for each estimated visit per user and website, and, optionally per other characteristics. This way, the subsequent analysis can be performed, initially, per user and per website or both, and, if additional information is provided (e.g., clients' location or contract with the company) in more detailed aggregates. Let us detail its implementation.

DNSPRINTS needs to emulate the DNS cache of each user but note that only those domains included in the set of footprints. To do so, the system uses four data structures:

- CA: Per user, a structure set to track the domains (all those included in the footprints) considered cached at a given time.
- $S_i^{user}(currently)$: Per user and per website in the monitor list, the current value of the sum of the weights of the domains cached included in the footprint of the website.
- PQ: A unique heap-based priority queue (priority according to timestamps) that tracks when the domains considered cached per user at a given time must be expired.
- HT: A structure to contain the DNS footprints illustrated as a matrix in Figure 4 but

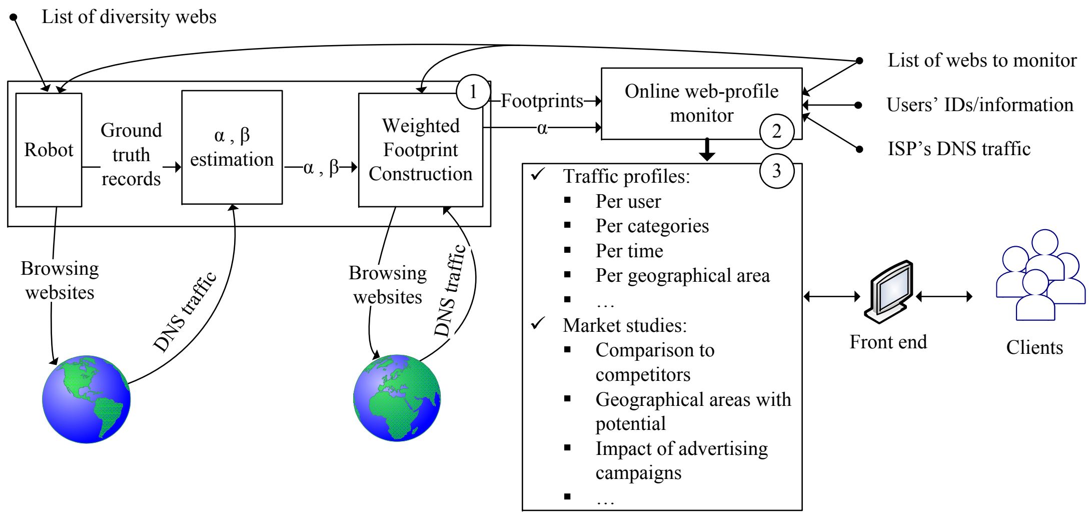


Figure 13: DNSPRINTS system's architecture

implemented as a hash table. Such hash table is indexed by domain and allows extracting the footprints (columns of Figure 4) where the domain was included and its relevance, i.e., the weight in the footprint (a cell in Figure 4).

Then, the program flow is:

- Every time a domain response for a user, $user$, is observed, d_i^j, such domain is searched in the set of cached domains (CA):
  - If it is not found:
    - For each those footprints including d_i^j (through HT), $S_i^{user}(currently)$ is increased by w_i^j (which is the corresponding weight associated to such domain name, and again, available at HT), and d_i^j is added to the user's list of cached domains (CA). If, then, S_i currently surpasses α threshold a visit for that website is recorded.
    - An event is added to the priority queue, PQ, which will subtract the domain's weight from S_i^{user} when time T (TTL of the domain's response) passes.
  - If it is found, the priority queue is updated. Actually, such update consists of adding an event to cancel the old expiration time, and another stating the new one. Nevertheless, it is worth remarking that the fact of receiving a response of a cached domain implies that a unexpected issue is occurring as cached domains do not need to be resolved. However, this characteristic is implemented because there are browsers that may violate the times indicated by DNS servers at their own discretion (e.g., TOR [28]).
- Synchronously, PQ launches events which remove domains from user cache (CA) according to its expiration time. When this happens, if S_i drops below α, the visit that previously was counted is now done and a record with the information is exported (e.g., H or G sets records).

These ideas have been translated into Pseudocode 1.

4.1.3 Analysis module

Finally, the monitor feeds the analysis module that prepares data to be accessed by clients. Currently, this module estimates visits per user, visits per websites and per category (e.g., car manufacturers) over time and grouped by geographical areas showing this information visually, as tables and maps.

Pseudocode 1 DNSPRINTS main program flow

```
HASH TABLE HT = empty
Initialize HT = Load Footprints-Weights
PRIORITY QUEUE PQ = empty
SET CA (per user) = empty
VARIABLE S (per user and footprint) = 0
for all DNS response (user,$d_i^j$, $T$) do
  for all Footprint ($d_i$,$w_i^j$) including $d_i^j$ at HT do
    $w_i^j$ = get weight in HT($d_i^j$)
    if (user, $d_i^j$) is not in $CA$ then
      Add (user, $d_i^j$) to $CA$
      Add (user, $w_i^j$) to $S$
      if $S > \alpha$ then
        Record-visit (currentTime, user, $d_i$)
      end if
      Insert (currentTime + T, user, EVENT substract and check finished visits (user, $d_i^j$, $w_i^j$)) in PQ
    else
      Insert (currentTime + remaining TTL, user, EVENT add (user, $d_i^j$, $w_i^j$)) in PQ
      Insert (currentTime + T, user, EVENT substract and check finished visits (user, $d_i^j$, $w_i^j$)) in PQ
    end if
  end for
end for
```

This module also provides market-oriented conclusions. It is prepared to compare profiles between clients and other competitor companies, as well as remarking those areas where clients are losing/gaining relevance. Also, it permits to relate the impact of an advertising campaign as Internet relevance or the popularity of mobile phone apps.

Finally, this module may exploit monitor outputs as clients require to and can be easily enhanced if companies may provide DNSPRINTS with information about users. As an example, given a phone operator, DNSPRINTS could focus on users that are inspecting alternatives in the market and whose contract is ending.

4.2 Performance

Once the architecture and operation of DNSPRINTS have been considered, we turn our attention to the evaluation of the speed and requirements of the online monitoring module, as footprint generation can be performed off-line and without demanding requirements.

The evaluation is performed on a desktop PC equipped with 16 GB of DDR3 memory and a 4 core Intel i5-3570K processor running at 3.40 GHz on an ASUS P8H77-M LE motherboard.

4.2.1 Throughput rate

It is relevant to study DNSPRINTS performance on different scenarios, as the throughput rate can vary depending on the popularity of monitored websites. Whenever a DNS packet is received, DNSPRINTS needs to perform two independent tasks: decode the DNS response packet and, should the decoded domains correspond to any DNS footprint, apply DNSPRINTS method to that packet. Therefore, for each packet a time T_{dec} is required to parse the DNS packet and, in some packets, a time T_{alg} is also required to apply the method. Consequently, the more traffic corresponds to monitored websites and the domains in its footprints, the lower throughput rate will be obtained.

To evaluate this phenomenon, DNSPRINTS has been executed over different proportions of traffic of websites to monitor. Using the browsing robot introduced in previous sections, the proportion of traffic that corresponds to websites to monitor over the total resolved domains has been progressively modified to span all the range [0-100%]. This allows calculating the throughput rate as a function of the proportion of monitored websites, as shown in Figure 14. It can be observed that throughput rate depends linearly on this proportion ranging between some 800,000 and 1 million DNS packets per second. Interestingly, 1 million queries per second is the workload of Cloudflare, one of the largest authoritative DNS providers in the world [29].

With a similar illustrative purpose, the DNSPRINTS's analysis has been executed over a DNS trace from a National Operator containing more than 3.6 million DNS packets from 20,000 different users. In this case, we propose to touch up operation by ruling out some of the most popular domains worldwide. We refer to domains such as Google, Apple, Facebook and Microsoft, for example. Intuitively, these companies are not the most likely clients of a ISP using DNSPRINTS as, actually, they have already their own mechanisms to create users analytics and exploit them.

Note that this approach has a negligible impact on the analysis' accuracy since the most popular domains have a very low weight in footprints because of inversely proportional weighting, but represents a significant cut in the number of DNS packets that DNSPRINTS has to handle.

Table 1 shows the percentage of queries to the set of ruled out domains found in the operator's trace. According to these figures, by filtering the 60% of DNS traffic corresponding to such top domains, DNSPRINTS is able to handle more than 1.4 million DNS packets per second. To put all this figure into perspective, and assuming that DNS traffic may represent between 1% and 5% of the traffic share [30], this result states the capacity of DNSPRINTS in the range of links of several dozens of Gb/s.

4.2.2 Memory requirements

DNSPRINTS' memory consumption depends on three variables, namely: the number of websites being monitored, the number of domain names in the footprints of such websites and the number of users.

The operator trace has several thousand users, hence DNSPRINTS' needs in terms of memory consumption are well below the capacity of our standard desktop PC (16 GB). To provide further nu-

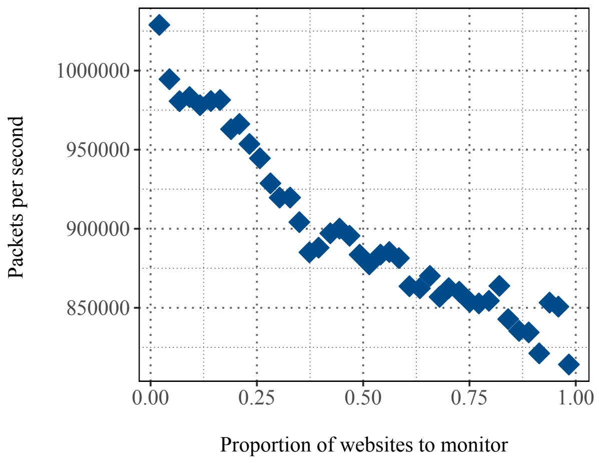


Figure 14: DNSPRINTS performance as a function of the proportion of traffic that corresponds to the list of websites to monitor.

Domains related to	% of queries
Google	20%
Apple	18%
Facebook	8%
Whatsapp	8%
Microsoft	6%
Total	60%

Table 1: Distribution of domain groups in the operator trace

meric examples on more demanding scenarios, let us extract from the dataset of Section 3.1 some empirical figures, and then extrapolate. As it turns out, a list of 300 websites to monitor would give in mean, according to the dataset, roughly 4500 different domains in the resulting set of footprints with an average domain name length of 20 bytes. Besides, the peak over time of the number of cached domains per user (of the domains included in footprints) would be 300 in the worst case.

Assume as well that at busy hours 50% of users are connected (again, as a worst case [31]) and a network with a million users. Then the memory for the footprints table is the addition of the cache table $300 \cdot 4500 \cdot 20$ =27 MB (20 bytes as averaged footprint size), the priority queue would be $500,000 \cdot 300 \cdot 40$ =6 GB (40 bytes per entry) and the S_i weight counters $100,0000 \cdot 300 \cdot 8$ =2.4 GB (8 bytes as a floating point number), totaling approximately 8.5 GB. This is a number that a typical desktop PC can handle.

Even in a challenging scenario such as an ISP's DNS server serving to 50 million users and 300 websites to monitor, memory requirement remains well below 1 TB, available as of today in high-end servers. And all this, in a single machine without resorting to parallelism (i.e., sharing the burden, for example, per sets of IP addresses) among several machines.

5 Related Work

The typical mechanisms for measuring the relative popularity of websites are based on intrusive software and tracker addons installed on clients' web browsers such as toolbars. Between these proposals, Alexa [5] stands out although the number of users is also remarkable in ComScore [32] and NetRatings [33]. Unfortunately, the information this approach provides is much aggregated and, importantly, is only based on volunteers' traffic, which does not guarantee a representative population. Alternatively, Google with its AdWords service [3] analyses the searches users carry out on its interface and exploits this information commercially. This approach is limited by the fact that users tend to search for website only the first times they visit it. After that, tools such as recent-history suggestions or bookmarks are used for subsequent accesses to those websites. In addition, our focus is on the opportunities that ISPs have in this area, instead of the ones from search-engine companies' as they are in fact monetizing their queries since years ago.

The natural approach from ISPs is to sniff HTTP traffic [34, 30] and interpret it. However, this approach comes together with limitations such as the proliferation of traffic encryption and the difficulty of storing, capturing and analyzing traces (especially for HTTP traffic, given its relevance in volume in the current Internet).

The answer from the research community was to turn to DNS protocol. The analysis of both DNS traffic and resolvers has been a research topic for long, whereby its performance, security, authenticity and behavior have been extensively studied [35, 36].

There are several tools for displaying and analyzing DNS traffic such as TreeTop [8] or dnstop [37]. However, none of these tools provide an accurate method for relating web traffic and user behavior. The same applies to the Cisco Umbrella Network [38]. Cisco makes use of DNS traffic to generate a one-million-domain popularity list similar to the one provided by Alexa. The information is obtained from its pool of DNS servers (OpenDNS network) but, unfortunately, apart from the worldwide popularity list itself and available reports, Cisco does not provide details of its processing algorithm nor specific software. Thus, limiting its usefulness for other members of the Internet community with access to DNS traffic.

In this regard, we remark several efforts to link ephemeral IP addresses and users on traffic captures apart from resorting to ISPs' RADIUS records. Specifically, Herrmann et al. [39] developed several DNS traffic analysis techniques for tracking users based on their daily behavior. In particular, using the previous day traffic records, it was possible to re-identify the users the day after by recognizing their browsing habits on a daily-changing dynamic IP addressing environment. We note that the problem we face is almost the opposite. Instead that the users' habits permit us to track them, we pursue to identify the web browsing user behavior.

Rajab et al. [40] related the probability of a host name to be in a DNS server's cache to the way users' access to websites. The result is an estimation of the density of users accessing to certain domains. As positive points, this method does not need access to the DNS traffic (but periodically polling DNS servers, although many servers are not vulnerable to such cache snooping) and users privacy is respected. However, the temporal scale, impossibility of relating popularity and types of users, and precision of the information that this mechanism provides is not enough for most of the commercial analysis we introduced before.

Interestingly, Krishnan and Monrose [41] showed that it was possible to estimate the keywords some users introduce at search engines by analyzing their DNS traffic. The authors observed that in the presence of full prefetching mechanisms, when users receive a list of websites related to the introduced keywords, some of such websites' addresses are preresolved. Those words that appear more frequently as part of resolved domain names can be considered as search terms' candidates. They compared 50 search queries and predictions over a day with results ranged between TPR of 85% and 73%, and FPR of 3% and 15%. In any case, we note that this approach is complementary to ours, while they focus on users' search terms, similarly to how search engines work with the limitations previously remarked, we analyze actual visits to websites.

As a further step, the research community turned its attention to tag flow records according to the DNS answers triggered by such flows. This approach, followed by Plonka et al. [8, 9] and Bermudez et al. [10], outputs a set of flows whose destination addresses are labeled by the host name servers derived from the DNS. This approach was further improved by Mori et al. [42] who paid special attention to the local cache problem when a DNS query is triggered by a final client. Essentially, much DNS traffic was missing due to such local caches which impeded flows from being labeled. They introduced a domain name graph representation whereby the relationship between addresses and real host names was stored and shared between different queries over time. Hence, previous resolved queries were used as a replacement for other queries missing. With respect to such previous works, DNSprints only needs access to the DNS traffic instead of HTTP/HTTPS which significantly simplifies and cheapens the regular operation, especially on large and distributed networks.

Moreover, as a key distinguishing feature, DNSprints exploits self-triggered traffic in two ways. First, it is used to ensure that a website was intentionally visited and second, to mitigate cache impact. Note that unintentional visits do not generate the set of resolved domains that websites' footprints require, and that, although caches may prevent the query of root domain from being performed, some of the rest of domains in the footprints (likely, with lower TTLs) will be effectively resolved.

We remark that these characteristics are key for constructing accuracy and marketable web-profiles, otherwise such profiles would lead to erroneous conclusions, e.g., a user is interested in some service in a certain extent when it is not true.

6 Conclusion and future work

We have presented a method for identifying accesses to websites using DNS footprinting of interest for web analytics. The method exploits the fact that current websites contain a set of contents such as ads, banners, other services of the owner of the website and social networks, as well as that some contents are prefetched and preloaded by browsers although not intentionally requested. Such contents must be downloaded in order to fully load the website that the user introduced in the address bar, in other words they are self-triggered.

Our proposal takes advantage of such entanglement on websites to create a DNS footprint that allows identifying an access to a website not by analyzing only the DNS response for such website but also considering a fraction of the generated DNS responses resulting from visiting such website. This way, both cache problems and misidentification of self-triggered visits as intentional ones are avoided. Additionally, as our method only inspects DNS traffic we can surpass the HTTPS monitoring restriction providing user web-profiling even when encryption is present.

The proposed DNSprints method has been evaluated in terms of accuracy with satisfactory results, i.e., obtaining false positive rates between 2 and 9% and true positive rates in between 90% and 95% in diverse scenarios. Then, the method has been implemented in a namesake system whose performance ranges between some 800,000 and 1.4 million packets per second, which translates into multi-Gb/s rates, in a single desktop PC.

To conclude, as future work we plan to measure the impact of ads or social media blockers such as Adblock or Ghostery browser addons may have in the generation of footprints. Although such resources tend to have low TTLs and, given that they are very common in the set of websites under study, their relevance in footprints should be low. In addition, DNSPRINTS has been tested with some ISP's traces from a DNS machine serving some 20,000 users, however its theoretical bound, even in a unique machine, is over a million users, we plan to contrast such figures with other larger ISP traces.

Acknowledgments

The authors would like to acknowledge funding received through TRAFICA (TEC2015-69417-C2-1-R) grant from the Spanish R&D programme. The authors thank Víctor Uceda for his collaboration in the early stages of this work.

References

References

[1] T. S. Raghu, P. K. Kannan, H. R. Rao, A. B. Whinston, Dynamic profiling of consumers for customized offerings over the Internet: a model and analysis, Decision Support Systems 32 (2) (2001) 117–134.

[2] Y. Yang, Web user behavioral profiling for user identification, Decision Support Systems 49 (3) (2010) 261–271.

[3] Google adwords.
URL `https://www.google.com/adwords`

[4] M. Lee, Google ads and the blindspot debate, Media, Culture & Society 33 (3) (2011) 433–447.

[5] About Alexa data.
URL `http://www.alexa.com/about`

[6] A. Juels, M. Jakobsson, T. Jagatic, Cache cookies for browser authentication, in: IEEE Symposium on Security and Privacy, 2006, pp. 300–305.

[7] V. Moreno, J. Ramos, J. L. García-Dorado, I. Gonzalez, F. J. Gomez-Arribas, J. Aracil, Testing the capacity of off-the-shelf systems to store 10Gbe traffic, IEEE Communications Magazine 53 (9) (2015) 118–125.

[8] D. Plonka, P. Barford, Context-aware clustering of DNS query traffic, in: ACM SIGCOMM Conference on Internet Measurement, 2008, pp. 217–230.

[9] D. Plonka, P. Barford, Flexible traffic and host profiling via DNS rendezvous, in: SATIN Workshop on Securing and Trusting Internet Names, 2011, pp. 29–36.

[10] I. Bermudez, M. Mellia, M. M. Munafò, R. Keralapura, A. Nucci, DNS to the rescue: Discerning content and services in a tangled web, in: ACM SIGCOMM Conference on Internet Measurement, 2012, pp. 413–426.

[11] T. Mori, T. Inoue, A. Shimoda, K. Sato, S. Harada, K. Ishibashi, S. Goto, Statistical estimation of the names of HTTPS servers with domain name graphs, Computer Communications 94 (2016) 104–113.

[12] M. Fullmer, S. Romig, The OSU flowtools package and Cisco Netflow logs, in: USENIX LISA Conference, 2000, pp. 291–304.

[13] T. Callahan, M. Allman, M. Rabinovich, On modern DNS behavior and properties, ACM SIGCOMM Computer Communication Review 43 (3) (2013) 7–15.

[14] H. Choi, H. Lee, Identifying botnets by capturing group activities in DNS traffic, Computer Networks 56 (1) (2012) 20–33.

[15] D. Acarali, M. Rajarajan, N. Komninos, I. Herwono, Survey of approaches and features for the identification of HTTP-based botnet traffic, Journal of Network and Computer Applications 76 (2016) 1–15.

[16] K. Ishibashi, T. Toyono, K. Toyama, M. Ishino, H. Ohshima, I. Mizukoshi, Detecting mass-mailing worm infected hosts by mining DNS traffic data, in: ACM SIGCOMM Workshop on Mining Network Data, 2005, pp. 159–164. `doi:10.1145/1080173.1080175`.

[17] J. Jung, E. Sit, H. Balakrishnan, R. Morris, DNS performance and the effectiveness of caching, IEEE/ACM Transactions on Networking 10 (5) (2002) 589–603.

[18] SeleniumHQ Browser Automation.
URL http://www.seleniumhq.org

[19] tcpdump and libpcap.
URL http://www.tcpdump.org

[20] A. Cockburn, B. Mckenzie, What do web users do? An empirical analysis of web use, International Journal of Human-Computer Studies 54 (6) (2001) 903–922.

[21] A. Bianco, G. Mardente, M. Mellia, M. Munafò, L. Muscariello, Web user-session inference by means of clustering techniques, IEEE/ACM Transactions on Networking 17 (2) (2009) 405–416.

[22] Robot for web browsing.
URL https://github.com/jlgarciadorado/robotFWB

[23] Alexa top 500 sites on the web.
URL http://www.alexa.com/topsites

[24] R. Real, J. M. Vargas, The probabilistic basis of Jaccard's index of similarity, Systematic Biology 45 (3) (1996) 380–385.

[25] F. Chierichetti, R. Kumar, S. Pandey, S. Vassilvitskii, Finding the Jaccard median, in: ACM SIAM Symposium on Discrete Algorithms, 2010, pp. 293–311.

[26] V. Moreno, P. M. Santiago del Río, J. Ramos, D. Muelas, J. L. García-Dorado, F. J. Gomez-Arribas, J. Aracil, Multi-granular, multi-purpose and multi-Gb/s monitoring on off-the-shelf systems, International Journal of Network Management 24 (4) (2014) 221–234.

[27] DNSprints.
URL https://repit.ii.uam.es/gitlab/hilo/DNSprints

[28] B. Greschbach, T. Pulls, L. M. Roberts, P. Winter, N. Feamster, The effect of DNS on Tor's anonymity, arXiv.

[29] Cloudflare Inc.
URL https://www.cloudflare.com/dns

[30] J. L. García-Dorado, A. Finamore, M. Mellia, M. Meo, M. Munafò, Characterization of ISP traffic: Trends, user habits, and access technology impact, IEEE Transactions on Network and Service Management 9 (2) (2012) 142–155.

[31] G. Maier, A. Feldmann, V. Paxson, M. Allman, On dominant characteristics of residential broadband Internet traffic, in: ACM SIGCOMM Conference on Internet Measurement, 2009, pp. 90–102.

[32] Comscore Inc.
URL http://www.comscore.com

[33] Netratings Inc.
URL http://www.nielsen-netrating.com

[34] G. Maier, A. Feldmann, V. Paxson, M. Allman, On dominant characteristics of residential broadband Internet traffic, in: ACM SIGCOMM Conference on Internet Measurement, 2009, pp. 90–102.

[35] X. Ma, J. Zhang, Z. Li, J. Li, J. Tao, X. Guan, J. C. Lui, D. Towsley, Accurate DNS query characteristics estimation via active probing, Journal of Network and Computer Applications 47 (2015) 72–84.

[36] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, T. Holz, Going wild: Large-scale classification of open DNS resolvers, in: ACM SIGCOMM Internet Measurement Conference, 2015, pp. 355–368.

[37] The measurement factory, DNS tools.
URL http://dns.measurement-factory.com/tools

[38] Cisco Umbrella 1 million.
URL https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million

[39] D. Herrmann, C. Banse, H. Federrath, Behavior-based tracking: Exploiting characteristic patterns in DNS traffic, Computers & Security 39, Part A (2013) 17–33.

[40] M. Rajab, F. Monrose, A. Terzis, N. Provos, Peeking through the cloud: DNS-based estimation and its applications, in: ACNS Applied Cryptography and Network Security, 2008, pp. 21–38.

[41] S. Krishnan, F. Monrose, DNS prefetching and its privacy implications: When good things go bad, in: USENIX Workshop on Large-scale Exploits and Emergent Threats, 2010, pp. 1–9.

[42] T. Mori, T. Inoue, A. Shimoda, K. Sato, K. Ishibashi, S. Goto, SFMap: Inferring services over encrypted web flows using dynamical domain name graphs, in: IFIP/ACM Traffic Monitoring and Analysis Workshop, 2015, pp. 126–139.