Can We Learn What People Are Doing from Raw DNS Queries?

Jianfeng Li[1], Xiaobo Ma[1*], Li Guodong[1], Xiapu Luo[2†], Junjie Zhang[3], Wei Li[1], Xiaohong Guan[1]
[1]MOE KLINNS Lab, Xi'an Jiaotong University, Xi'an, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[3]Department of Computer Science and Engineering, Wright State University, Dayton, USA
Email: jfli@sei.xjtu.edu.cn, {xma.cs, lgdli}@xjtu.edu.cn, csxluo@comp.polyu.edu.hk, junjie.zhang@wright.edu, liw@xanet.edu.cn, xhguan@sei.xjtu.edu.cn

***Abstract*—Domain Name System (DNS) is one of the pillars of today's Internet. Due to its appealing properties such as low data volume, wide-ranging applications and encryption free, DNS traffic has been extensively utilized for network monitoring. Most existing studies of DNS traffic, however, focus on domain name reputation. Little attention has been paid to understanding and profiling what people are doing from DNS traffic, a fundamental problem in the areas including Internet demographics and network behavior analysis. Consequently, simple questions like "How to determine whether a DNS query for `www.google.com` means searching or any other behaviors?" cannot be answered by existing studies. In this paper, we take the *first* step to identify user activities from raw DNS queries. We advance a multi-scale hierarchical framework to tackle two practical challenges, i.e., behavior ambiguity and behavior polymorphism. Under this framework, a series of novel methods, such as pattern upward mapping and multi-scale random forest classifier, are proposed to characterize and identify user activities of interest. Evaluation using both synthetic and real-world DNS traces demonstrates the effectiveness of our method.**

I. Introduction

Domain name system (DNS) is indispensable to nearly all Internet services because of its efficiency in mapping human-friendly domain names into machine-readable IP addresses. It has also been widely exploited by content delivery networks (CDNs) [1,2] and cloud services to accelerate network performance [3]. Due to its prevalence, DNS has been commonly utilized as a vantage point for network monitoring [4]–[7].

Since most existing studies of DNS traffic focus on domain name reputation (i.e., the likelihood to be benign or malicious) [4,8,9], little attention has been paid to understanding and profiling the purpose-driven user activities behind a sequence of raw DNS queries. In such an activity, a user behaves with a semantic purpose that is human-understandable, meaningful and scenario-specific, e.g., visiting a certain website and using a specific mobile app. In this paper, we aim at *identifying user activities from raw DNS queries*. Although seemingly not difficult to achieve such a goal by inspecting application-specific traffic (e.g., HTTP), it becomes challenging when only raw DNS queries are used, which will be detailed in the next paragraph. The benefits of studying this problem, as compared to inspecting application-specific traffic, naturally inherit all advantages of DNS traffic analysis, such as extremely low data volume, wide-ranging applications, and encryption free. More importantly, successfully solving the problem could endue DNS with a new paradigm in behavioral profiling, enriching people's traditional view on DNS traffic analysis.

Despite the benefits, the problem involves the following technical challenges that are not tackled in the literature.

Behavior Ambiguity. Domain names queried in one user activity may also be queried in other user activities, termed as domain multiplexing, leading to the ambiguity in user activity identification if one observes DNS queries separately. For example, in almost all user activities related to Google (e.g., Google play, Gmail, and Google plus), the domain name `www.google.com` will be queried. Through observing individual DNS queries, one *cannot* tell which user activity a query for `www.google.com` comes from. Intuitively, we can correlate multiple DNS queries during a period of time and analyze the DNS query pattern therein to disambiguate behaviors. However, the duration of the observation period, referred to as *time scale*, is difficult to determine, since DNS query patterns are often scale-sensitive. A small time scale may lose the contextual information of different domain names because just a few DNS queries fall into the time bin, while a large time scale would introduce substantial noises (i.e., DNS queries for irrelevant domain names). To make things more complicated, the time scales appropriate for observing different user activities may vary in a large range.

Behavior Polymorphism. To identify a user activity of interest, one needs to collect DNS query samples for training and extract the underlying DNS query patterns. However, it is extremely difficult, if not entirely impossible, to collect DNS query samples for training with sufficient sample diversity as a result of user behavior polymorphism. We use polymorphism to represent two practical phenomena. First, the underlying patterns of a certain user activity, say $\mathbb{A}$, may vary across different end users. For example, when visiting a website, different end users may browse different webpages, resulting in different DNS query patterns. Even in the case of the same webpage, the patterns of different end users may not be completely the same due to the interferences of local DNS caches. Second, the user activities that reuse domains names of $\mathbb{A}$ can hardly be fully enumerated. Compounded by the fact that we can only collect DNS query samples for training from a limited number of end users, the former (resp. latter) phenomenon causes the lack of *positive* (resp. *negative*) DNS query samples, thus likely leading to false negatives (resp. positives) in DNS query samples for testing. The samples are positive if induced by $\mathbb{A}$; otherwise negative.

To address *behavior ambiguity*, we propose a novel multi-scale hierarchical characterization method to represent a se-

*Jianfeng Li and Xiaobo Ma contributed equally to this work.
† Corresponding author.

quence of DNS queries, in favor of retaining contextual behavioral information at various increasing time scales in a bottom-up manner. The basic idea is that, at a certain time scale, the characteristics of DNS queries within each time bin (e.g., a feature vector describing the frequency of each underlying DNS query pattern) are recursively aggregated from those within multiple (successive) child time bins at a smaller time scale, wherein the characteristics at the smallest time scale within each time bin (e.g., a feature vector describing the frequency of each domain name is queried) are derived from raw DNS queries. A key technique before performing the aggregation is to map all feature vectors at the smaller time scale into underlying DNS query patterns, and then represent these patterns using a compressed representation.

To deal with *behavior polymorphism*, we perform DNS query sample recognition and expansion. On one hand, we propose a semi-supervised method to recognize unknown positive samples by exploring their co-occurrence relation with labeled samples. The proposed method could effectively reduce false negatives caused by the lack of positive samples. On the other hand, we generate (artificial) negative samples that encircle (i.e., close to but not within) regions of positive samples. In this way, the outliers, which are away from these regions in feature space, tend to be labeled as negative samples, hence expanding the diversity of negative samples and meanwhile mitigating the risk of false positives.

To the best of our knowledge, our work constitutes the *first* effort towards identifying user activities from raw DNS queries. We mainly make the following contributions:

• We propose a novel multi-scale hierarchical characterization method for DNS queries to recursively describing DNS query behavioral information at various increasing time scales in a bottom-up manner. With increased time scales, the characterization can gradually extract different DNS query patterns involving contextual behavior information with different visibility and granularities. The proposed characterization can also be used in other similar problems.

• To identify user activities based on the characterization, we devise a multi-scale random forest (MRF) classifier, which identifies user activities in a top-down manner (i.e., from large time scales to small ones) with increased granularities. MRF can not only accurately identify user activities, but also recognize unknown DNS query patterns of user activities with the help of a semi-supervised method.

• We build a working system to identify user activities from DNS traces. Experiments using both synthetic and real-world data demonstrate that our system can effectively and accurately identify user activities. We also demonstrate how our methods can be applied in user dynamics surveillance.

Roadmap. Section II describes the problem. Section III presents the multi-scale hierarchical characterization of DNS queries, and Section IV details multi-scale identification of user activities. We report experimental results in Section V, survey related work in Section VI, and conclude in Section VII.

II. Problem Description

Our goal is to identify user activities of one's interest and locate the time period when they occur. Fig. 1 shows our two-stage procedure with the example of identifying the user activity of "visiting Amazon website".

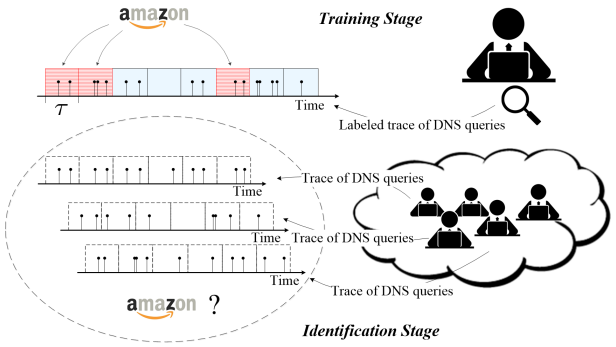


Fig. 1: A motivating example of user activity identification.

Stage 1: We collect DNS traces of end users on the border of the network (or at the DNS server). For each end user whose behavior is known, we label their traces of DNS queries. Specifically, along a series of τ-sized time bins, we assign each time bin a binary label that indicates the presence (red shaded) or the absence (blue shaded) of "visiting Amazon website". Labels can be decided using information from various sources, such as deep packet inspection and browser plug-ins that monitor web activities, depending on the specific user activities of interest. Using the labeled traces, we derive the multi-scale hierarchical characterization and train a multi-scale random forest classifier in Sections III and IV respectively.

Stage 2: For each of the remaining end users with unlabeled traces, leveraging the models trained in *stage 1*, we identify the presence/absence of the user activity "visiting Amazon website" in all its τ-sized time bins.

III. Multi-Scale Hierarchical Characterization of DNS Queries

To identify user behavior from DNS queries, an immediate demand is to formally characterize the queries, facilitating the representation of all possible underlying behavioral patterns. To this end, we propose the multi-scale hierarchical characterization to represent the queries, in favor of retaining contextual information at various time scales.

A. Representing DNS Query Dynamics at Multiple Time Scales

Fig. 2 exemplifies a trace of DNS queries from an end-user, where each bar represents a DNS query. Let $\mathbb{A}$ be the user activity of interest occurring between t_1 and t_2 (red shaded), and $\mathbf{D} = (a, b, \ldots, i)$ be the list of domain names. The entire period is divided into a series of τ-size time bins. Assume that a, b, c, d are domain names queried in $\mathbb{A}$. However, these domain names are also queried when $\mathbb{A}$ is absent (e.g., t_3 to t_4 and t_5 to t_6). Other domain names (e.g., e, f, g) that are not queried by $\mathbb{A}$ but DNS queries for them are interleaved during the same period when $\mathbb{A}$ occurs. DNS query dynamics in the time bin W_t is denoted by a vector $\mathbf{x}_t$, where $\mathbf{x}_t(j)$ counts the number of DNS queries that fall into W_t for the jth domain name in $\mathbf{D}$. For example, DNS query dynamics in W_1 is denoted by $\mathbf{x}_1 = (1, 1, 0, 0, 1, 0, 0, 0, 0)$.

Note that a small value of τ achieves fine-grained characterization but renders the loss of contextual information of different domain names. An extreme case is completely losing

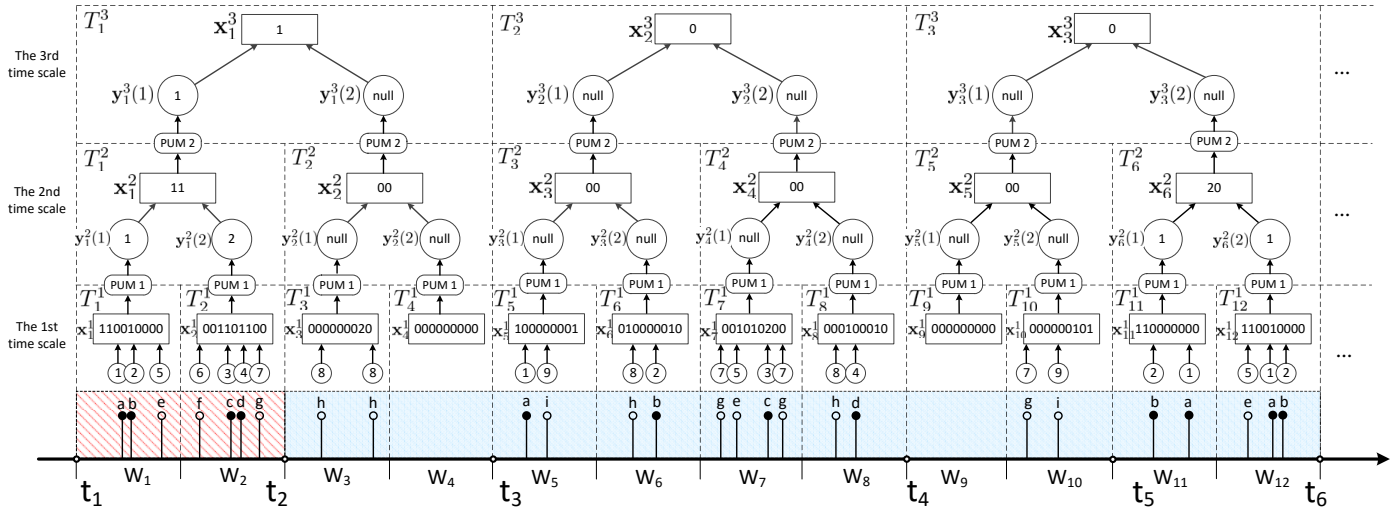


Fig. 2: An illustration of the multi-scale hierarchical characterization of DNS queries.

the contextual information if τ is less than the minimal time interval between DNS queries, because at most one query falls into each time bin. However, a large value of τ may ignore the inner fine-grained information within each time bin. For example, if we increase τ such that the interval from t_1 to t_2 falls in the 1st time bin and that from t_3 to t_4 falls into the 2nd time bin, DNS query dynamics between t_1 and t_2 cannot be distinguished from that between t_3 and t_4 by observing domain name combination a, b, c, d. Moreover, a large value of τ may introduce substantial noises (i.e., DNS queries for irrelevant domain names). Last, characterizing DNS query dynamics at a fixed time scale cannot effectively distinguish different user activities. For example, in Fig. 2, $\mathbf{x}_1$ is the DNS query dynamics of $\mathbb{A}$, whereas $\mathbf{x}_{12}$ is not. However, we have $\mathbf{x}_1 = \mathbf{x}_{12} = (1, 1, 0, 0, 1, 0, 0, 0, 0)$.

We propose a multi-scale hierarchical characterization for DNS query dynamics based on decision tree. Our characterization improves the distinguishability by capturing the structural differences between user activities at different time scales. To this end, we construct hierarchical time bins as shown in Fig. 2 and recursively characterize DNS query dynamics in these time bins. Let n be the number of time scales ($n = 3$ in Fig. 2). For the kth time scale, the tth time bin is denoted by T_t^k. If $k = 1$, we have $T_t^1 = W_t$; otherwise, T_t^k ranges from $(t-1)\tau \prod_{j=1}^{k-1} \alpha_j$ to $t\tau \prod_{j=1}^{k-1} \alpha_j$, where $\alpha_j \in \mathbb{N}$ is the time scale *inflation ratio* from the jth time scale to the $(j+1)$th time scale. We refer to time bins at the jth time scale ($j < k$) as the child time bins of T_t^k if they fall into the range of T_t^k, and T_t^k as the parent time bin of these child time bins accordingly. For example, in Fig. 2, we have $\alpha_1 = \alpha_2 \cdots = \alpha_k = 2$. Note that all inflation ratios do not need to be equal. To characterize DNS query dynamics in hierarchical time bins, we define *query pattern abstraction* at each time scale.

Definition 1. *A query pattern abstraction (QPA) at the kth time scale is an elementary entity to characterize the DNS query dynamics in time bins at the kth time scale.*

A QPA at the 1st time scale is a DNS query. It is denoted by an integer, i.e., the index of domain name in $\mathbf{D}$. At the kth time scale ($k > 1$), a QPA is a compressed representation of DNS query dynamics in the corresponding child time bin at the $(k-1)$th time scale. We characterize the DNS query dynamics in each time bin by counting the numbers of different QPAs in this time bin. QPAs in each time bin are therefore aggregated into a vector dubbed *QPA-vector*. We construct multi-scale characterization by recursively aggregating QPAs into QPA-vector and mapping QPA-vector to the QPA at a larger time scale. Formally, we denote the ith QPA in T_t^k by $\mathbf{y}_t^k(i)$. Aggregating all QPAs in T_t^k yields the QPA-vector in T_t^k, denoted as $\mathbf{x}_t^k$. Specifically, QPAs and QPA-vectors are derived recursively as follows. At the 1st time scale, we have $\mathbf{x}_t^1 = \mathbf{x}_t$. At the kth time scale ($k > 1$), we have

$$\mathbf{y}_t^k(i) = \text{PUM}_{k-1}(\mathbf{x}_{t'}^{k-1}), \tag{1}$$

where $1 \le i \le \alpha_{k-1}$ and $t' = \alpha_{k-1}t - \alpha_{k-1} + i$. Assume that there are m_k different QPAs at the kth time scale. $\mathbf{x}_t^k$ is an integer vector consisting of m_k elements. We have

$$\mathbf{x}_t^k(j) = \sum_{i=1}^{\alpha_{k-1}} \mathbf{1}\{\mathbf{y}_t^k(i) = j\}, \tag{2}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. Take an example from Fig. 2. The QPA-vector $\mathbf{x}_1^1 = (1, 1, 0, 0, 1, 0, 0, 0, 0)$ is mapped into the QPA $\mathbf{y}_1^2(1) = 1$ via PUM_1. Aggregating QPAs $\mathbf{y}_1^2(1) = 1$ and $\mathbf{y}_1^2(2) = 2$ yields the QPA-vector in T_1^2, i.e., $\mathbf{x}_1^2 = (1, 1)$. In (1), $\text{PUM}_{k-1}(\cdot)$ is the pattern upward mapper at the $(k-1)$th time scale, which is constructed based on decision tree (see Section III-B).

By leveraging QPA-vectors at different time scales, we can distinguish DNS query dynamics of $\mathbb{A}$ from that of other user activities at some proper time scales. For example, DNS query dynamics of $\mathbb{A}$ (between t_1 and t_2) can be distinguished from that between t_3 and t_4 at all time scales. It can also be distinguished from DNS query dynamics between t_5 and t_6 at the 2nd and 3rd time scales. In addition, our characterization is noise-tolerate since it is immune to the impact of irrelevant domain names.

B. Mapping QPA-Vector into QPA at A Larger Time Scale

To map the QPA-vector at the kth time scale into the QPA at the $(k+1)$th time scale, we construct the pattern upward mapper $\text{PUM}_k(\cdot)$ in a supervised manner. We collect training traces of DNS queries from end users. These traces are labeled

to indicate the presence/absence of $\mathbb{A}$. Formally, a labeled trace of DNS queries can be expressed by $\mathcal{S} = \{s_t\}_{t=1}^{M}$, where $s_t = \langle \mathbf{x}_t, a_t \rangle$ is the sample extracted from the time bin W_t and $a_t = 1$ (resp. $a_t = 0$) indicates the presence (resp. absence) of $\mathbb{A}$ in W_t. We derive training data of PUM$_k$ (for $k = 1, 2, \ldots, n$) from each labeled trace. Specifically, training data induced by a labeled trace comprises a series of samples, where the tth sample, denoted by $b_t^k = \langle \mathbf{x}_t^k, e_t^k \rangle$, consists of a feature vector $\mathbf{x}_t^k$ and a label e_t^k. The feature vector $\mathbf{x}_t^k$ is the QPA-vector in T_t^k. Obviously, $\mathbf{x}_t^k$ depends on PUM$_{k-1}$. Therefore, we recursively construct training data of pattern upward mappers from small time scales to large ones. The label e_t^k is derived by $e_t^k = \bigvee_{j=t_s}^{t_e} a_j$, where $t_s = 1+(t-1)\prod_{i=1}^{k} \alpha_i$ and $t_e = t\prod_{i=1}^{k} \alpha_i$. Note that $e_t^k = 1$ (resp. $e_t^k = 0$) indicates the presence (resp. absence) of $\mathbb{A}$ in the parent time bin of T_t^k. Integrating samples extracted from all labeled traces yields the training set of PUM$_k$, denoted by $\mathcal{B}_k = \{b_i^k\}_{i=1}^{M_k}$.

Note that the QPA-vector $\mathbf{x}_t^k$ is an integer vector, thus can be represented in tree-based structure. Fig. 3 illustrates a tree-based representation of QPA-vector at the 1st time scale. Let $\mathscr{T}$ be the set of QPA-vectors. $\mathscr{T}$ is divided into different partitions with the splitting of tree. Each partition of QPA-vectors corresponds to a unique path from root to leaf. All these paths constitute the set $\mathcal{U}^\beta$, where β is the depth of tree. The information gain caused in the splitting of tree is

$$IG(\mathscr{T},\mathcal{U}^\beta) = \mathrm{H}(\mathscr{T}) - \sum_{u^j \in \mathcal{U}^\beta} \frac{|\mathcal{L}(u^j)|}{|\mathscr{T}|}\mathrm{H}(\mathcal{L}(u^j)), \tag{3}$$

where $\mathcal{L}(u^j)$ returns the subset of $\mathscr{T}$ corresponding to the path u^j and $H(\cdot)$ is the information entropy. We rewrite (3) as

$$\begin{aligned} IG(\mathscr{T},\mathcal{U}^\beta) = \mathrm{H}(\mathscr{T}) &- \sum_{u^j \in \mathcal{U}^\beta} \frac{|\mathcal{L}(u^j)|}{|\mathscr{T}|} g\Big(\frac{\sum_{v^i_{1:\beta}=u^j} |\mathcal{L}(v^i)|p_i}{|\mathcal{L}(u^j)|}\Big) \\ &- \sum_{u^j \in \mathcal{U}^\beta} \frac{|\mathcal{L}(u^j)|}{|\mathscr{T}|} g\Big(\frac{\sum_{v^i_{1:\beta}=u^j} |\mathcal{L}(v^i)|(1-p_i)}{|\mathcal{L}(u^j)|}\Big), \end{aligned} \tag{4}$$

where $v^i \in \mathcal{U}^{\beta+c}$, $g(x) = x\log x$, and p_i is the probability that a QPA-vector in $\mathcal{L}(v^i)$, say $\mathbf{x}_t^k$, has the label $e_t^k = 1$. It is worth noting that $g(\cdot)$ is a convex function. According to Jensen's inequality, we have

$$\begin{aligned} IG(\mathscr{T},\mathcal{U}^\beta) \le \mathrm{H}(\mathscr{T}) &- \sum_{u^j \in \mathcal{U}^\beta} \frac{\sum_{v^i_{1:\beta}=u^j} |\mathcal{L}(v^i)|g(p_i)}{|\mathscr{T}|} \\ &- \sum_{u^j \in \mathcal{U}^\beta} \frac{\sum_{v^i_{1:\beta}=u^j} |\mathcal{L}(v^i)|g(1-p_i)}{|\mathscr{T}|} = IG(\mathscr{T},\mathcal{U}^{\beta+c}). \end{aligned} \tag{5}$$

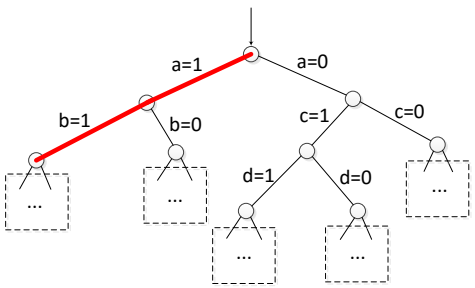


Fig. 3: Representing QPA-vector in tree-based structure.

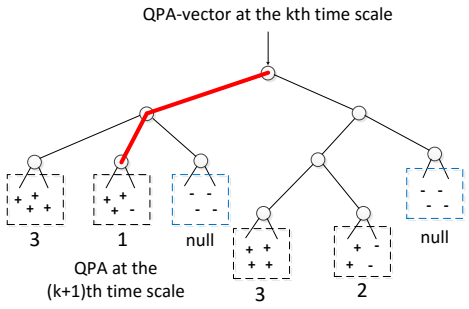
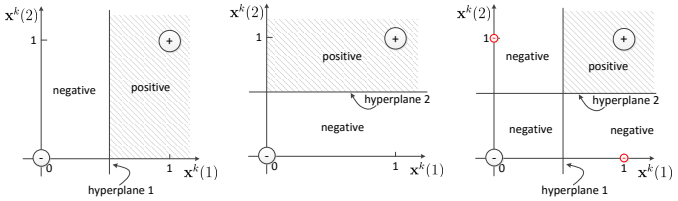


Fig. 4: Decision-tree-based pattern upward mapping.



(a) Splitting with hyperplane 1. (b) Splitting with hyperplane 2. (c) Splitting with hyperplane 1 and hyperplane 2.

Fig. 5: Basic idea of introducing artificial negative samples.

To map a QPA-vector at the kth time scale to a QPA at the $(k+1)$th time scale, we need to analyze whether and to what extent this QPA-vector can indicate the presence/absence of $\mathbb{A}$ at the $(k+1)$th time scale. According to (5), the splitting of tree results in reduction in entropy, or equivalently the uncertainty of telling whether $\mathbb{A}$ occurs in the parent time bin at the $(k+1)$th time scale given a QPA-vector at the kth time scale.

A path from root to leaf distinguishes a (sort of) QPA-vector(s) from those corresponding to other paths. For example, in Fig. 3, the path in red distinguishes QPA-vectors in T_1^1, T_{11}^1, and T_{12}^1 from those in other time bins in Fig. 2. This observation inspires us to employ the path as the compressed representation of QPA-vectors. Increasing β enhances the discrimination of a QPA-vector, while rendering an exponential growth of the number of unique QPA-vectors. Even through a complete splitting, i.e., $\beta = \dim(\mathbf{x}_t^k)$, some QPA-vectors belonging to different user activities (e.g., QPA-vectors in T_1^1 and T_{12}^1) are still indistinguishable. We refer to such QPA-vectors as the *ambiguous* QPA-vectors, which will be handled in two steps. First, we seek the optimal compressed representation of QPA-vector that maximizes the information gain using a decision tree trained with $\mathcal{B}_k$. Second, we take advantage of the *contextual* behavior (i.e., other QPAs in the parent time bin) to distinguish ambiguous QPA-vectors. This process essentially further splits these ambiguous QPA-vectors, leading to further entropy reduction according to (5).

Fig. 4 illustrates pattern upward mapping from the QPA-vector at the kth time scale to the QPA at the $(k+1)$th time scale. By applying the decision tree, QPA-vectors of samples in $\mathcal{B}_k$ are divided into partitions associated to different leaf nodes. Let $\Phi_1, \Phi_2, \ldots, \Phi_N$ be these partitions, which are arranged in the decreasing order of information entropy. If the entropy of a partition is positive, this partition is called an *ambiguous* partition. Otherwise, if a partition only consists of positive samples (resp. negative samples), it is called a *positive* partition (resp. *negative* partition). QPA-vector in positive partitions can independently indicates the presence of $\mathbb{A}$. It means that the

contextual characteristics is not needed on the presence of such a QPA-vector. To reduce the feature dimension (i.e., the number of unique QPAs) at the $(k+1)$th time scale, we merge positive partitions as one (denoting them by the same index). That is, we map QPA-vectors in positive partitions into the same QPA. Likewise, QPA-vectors in negative partitions can independently indicate the absence of $\mathbb{A}$. All of them can be mapped into the same QPA. To further reduce the feature dimension, we leave out this QPA because it can be naturally represented by the absence of all other QPAs. Therefore, we leave out negative partitions (denoted by "null"). Consequently, partitions can be expressed by $\Phi_1, \Phi_2, \ldots, \Phi_{N'}$, where N' is the final number of partitions. We conduct pattern upward mapping using these partitions. Given a QPA-vector at the kth time scale, say $\mathbf{x}_t^k$, if it falls into Φ_i, we obtain $\text{PUM}_k(\mathbf{x}_t^k) = i$. In Fig. 4, the resulting QPA at the $(k+1)$th time scale is 1, since the QPA-vector at the kth time scale falls into Φ_1.

C. Enhancing The Discrimination of Pattern Upward Mapping

In practice, the user activities that reuse domain names of $\mathbb{A}$ can hardly be fully enumerated, hence leading to the lack of diversity of negative samples. Insufficiency of negative samples may degrade the discrimination of pattern upward mapping. Fig. 5 shows the distribution of training data in two-dimensional feature space. Positive samples are located in $\mathbf{x}^k = (1,1)$, whereas negative samples are all concentrated in $\mathbf{x}^k = (0,0)$. Since decision tree maximizes the information gain via the splitting of samples, the hyperplane to split positive and negative samples can be either hyperplane 1 in Fig. 5(a) or hyperplane 2 in Fig. 5(b). Further splitting will not happen since no information gain can be obtained from any further splitting. Thus, $\mathbf{x}^k = (1,1)$ cannot be distinguished from $\mathbf{x}^k = (1,0)$ or $\mathbf{x}^k = (0,1)$. This is because of the over-generalization of decision tree. Consequently, a QPA-vector without labels may be erroneously treated as a positive sample if domain names queried in this QPA-vector is a subset of domain names queried in the QPA-vectors that have been labeled as positive samples in training set. Such a disadvantage degrades the discrimination of pattern upward mapping, potentially leading to the increase of false positives.

To tackle this problem, we construct artificial negative samples in training data to encircle the regions of positive samples. Any outlier away from regions of positive samples tend to be viewed as negative samples, thereby overcoming the insufficiency of negative samples. Fig. 5(c) illustrates how artificial negative samples (in red) improve the discrimination of pattern upward mapping. By leveraging these artificial negative samples, feature space is partitioned to maximize the information gain. In the partitioned feature space, feature vector $\mathbf{x}^k = (1,1)$ can be distinguished from $\mathbf{x}^k = (1,0)$ and $\mathbf{x}^k = (0,1)$. Let Ψ_k be a set comprising QPA-vector of all positive samples in $\mathcal{B}_k$. Artificial negative samples are constructed by modifying QPA-vectors in Ψ_k, as is elaborated in Algorithm 1. Let $L_k = \eta \cdot |\Psi^k|$ be the number of artificial negative samples to construct. We set $\eta = 0.1$. The output of Algorithm 1 is the set of artificial negative samples $\mathcal{B}'_k$.

Algorithm 1 Generating artificial negative samples.

Input: Ψ_k, L_k
Output: $\mathcal{B}'_k$
1: $\mathcal{B}'_k \leftarrow \emptyset$
2: **for** $l = 0, 1, \ldots, L_k$ **do**
3: Pick a random QPA-vector $\mathbf{x}^k \in \Psi_k$
4: Pick a random index $i \in [1, \dim(\mathbf{x}^k)]$ with the probability

$$q_i = \begin{cases} \dfrac{1}{\sum_j \mathbf{1}\{\mathbf{x}^k(j) > 0\}}, & \mathbf{x}^k(i) > 0, \\ 0, & \mathbf{x}^k(i) = 0. \end{cases}$$

5: Generate a feature vector $\tilde{\mathbf{x}}^k$ as

$$\tilde{\mathbf{x}}^k(j) = \begin{cases} 0, & j = i, \\ \mathbf{x}^k(j), & j \neq i. \end{cases}$$

6: $\tilde{e}^k \leftarrow 0$
7: $\mathcal{B}'_k \leftarrow \mathcal{B}'_k \cup \{\langle \tilde{\mathbf{x}}^k, \tilde{e}^k \rangle\}$
8: **end for**

IV. Multi-Scale User Activity Identification

Using the multi-scale characterization, we devise the multi-scale random forest (MRF) classifier to recursively identify whether $\mathbb{A}$ occurs in each time bin at different times. We also propose a semi-supervised method to recognize unknown DNS query patterns of $\mathbb{A}$ to facilitate user activity identification.

A. Identifying User Activity at Different Time Scales

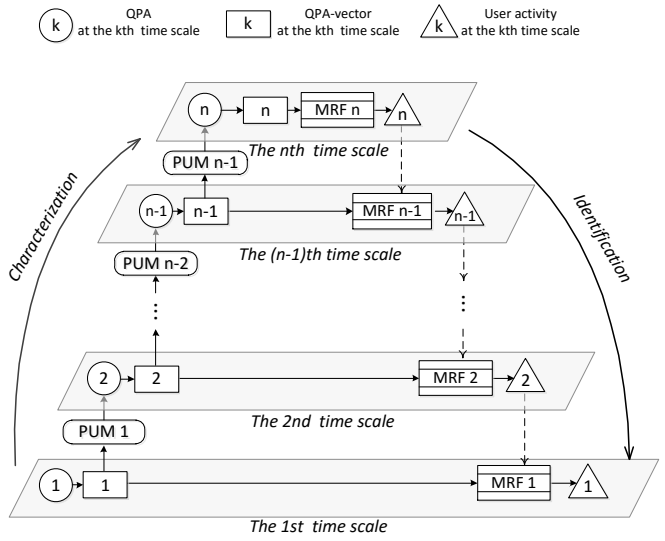


Fig. 6: The framework of multi-scale random forest classifier.

As shown in Fig. 6, the characterization of DNS query dynamics is hierarchical from small time scales to the large ones. In contrast to characterization, we identify user activities in a recursive fashion from large scales to the small ones, leading to gradually fine-grained identification. We conduct multi-scale identification of user activities by leveraging the MRF, where we construct a random forest (RF) classifier at each time scale. The RF for the kth time scale is denoted by MRF_k. Because the predicted result of MRF_k depends on that of MRF_{k+1}, we refer to MRF_k for $k = 1, 2, \ldots, n-1$ as the *conditional* RF.

To train MRF_k, we extract training data from labeled DNS traces. Let $\mathcal{S}_a = \{s_t\}_{t=1}^M$ be a labeled trace of DNS queries, where $s_t = \langle \mathbf{x}_t, a_t \rangle$ is an original sample extracted in W_t. Let $\mathcal{G}_a^k = \{g_t^k\}_{t=1}^{N_a^k}$ be a set consisting of candidate samples of MRF_k induced by $\mathcal{S}_a$, where $g_t^k = \langle \mathbf{x}_t^k, z_t^k \rangle$ comprises a

feature vector $\mathbf{x}_t^k$ and a label z_t^k. For the feature vector, $\mathbf{x}_t^k$ is recursively derived from (2). For the label, we have $z_t^1 = a_t$ and $z_t^k = \vee_{j=t_s}^{t_e} a_j$ for $k > 0$, where $t_s = 1 + (t-1)\prod_{i=1}^{k-1} \alpha_i$ and $t_e = t\prod_{i=1}^{k-1} \alpha_i$. Let $\mathcal{R}_a^k$ be the training set of MRF_k induced by $\mathcal{S}_a$. At the largest time scale (i.e., $k = n$), $\mathcal{R}_a^n$ contains all samples in $\mathcal{G}_a^n$, i.e., $\mathcal{R}_a^n = \mathcal{G}_a^n$. At the kth time scale ($k < n$), $\mathcal{R}_a^k$ contains a portion of samples in $\mathcal{G}_a^k$. Specifically, we have $\mathcal{R}_a^k = \{\langle \mathbf{x}_t^k, z_t^k \rangle \in \mathcal{G}_a^k | \vee_{j=t_s}^{t_e} z_j^k = 1\}$ for $k < n$, where $t_s = \lfloor t/\alpha_k \rfloor \alpha_k + 1$ and $t_e = \lceil t/\alpha_k \rceil \alpha_k$. Integrating samples extracted from all labeled traces yields the training set of MRF_k, denoted by $\mathcal{R}^k = \{r_i^k\}_{i=1}^{N_k}$.

Once the MRF has been trained, we use it to identify user activities. Given the DNS query trace of an end user, we first obtain their multi-scale hierarchical characterization using $\text{PUM}_1, \text{PUM}_2, \ldots, \text{PUM}_{n-1}$. At the kth time scale, we obtain QPA-vectors $\hat{\mathbf{x}}_1^k, \hat{\mathbf{x}}_2^k, \ldots$ for the time bins $T_1^k, T_2^k, \ldots$, respectively. Then, we identify the presence/absence of $\mathbb{A}$ from large time scales to the small. Specifically, the predicted label $\hat{z}_t^k = 1$ (resp. $\hat{z}_t^k = 0$) indicates the presence (resp. absence) of $\mathbb{A}$ in the time bin T_t^k. When $k = n$, we have

$$\hat{z}_t^n = \text{MRF}_n(\hat{\mathbf{x}}_t^n); \tag{6}$$

when $k < n$, we have

$$\hat{z}_t^k = \begin{cases} \text{MRF}_k(\hat{\mathbf{x}}_t^k), & \text{if } \hat{z}_{\lceil t/\alpha_k \rceil}^{k+1} = 1, \\ 0, & \text{if } \hat{z}_{\lceil t/\alpha_k \rceil}^{k+1} = 0. \end{cases} \tag{7}$$

In (6) and (7), $\text{MRF}_k(\hat{\mathbf{x}}_t^k)$ returns the predicted label of $\hat{\mathbf{x}}_t^k$ by applying MRF_k.

B. Recognizing Unknown DNS Query Patterns of $\mathbb{A}$

In practice, training data extracted from a limited number of end users may not cover all DNS query patterns of $\mathbb{A}$. That is, positive samples may be insufficient in reflecting the diversity of $\mathbb{A}$'s DNS query patterns. It potentially increases the risk of false negatives in user activity identification. To overcome this challenge, we propose a semi-supervised method to further recognize unknown DNS query patterns of $\mathbb{A}$, i.e., DNS query patterns belonging to $\mathbb{A}$ but not involved in the training data.

Our basic assumption is that DNS queries occurring together with the user activity $\mathbb{A}$ are probably the DNS queries of $\mathbb{A}$. Thus, recognizing unknown DNS query patterns of $\mathbb{A}$ can be transformed into analyzing the co-occurrence of DNS queries and $\mathbb{A}$. To this end, we capture *unlabeled* traces of DNS queries from large-scale end users. By applying the methods proposed in Section III and Section IV, we obtain the predicted label $\hat{z}_t^k$ (for k=1,2,...,n and $t = 1, 2, 3, \ldots$) for each time bin in these traces. Let $\mathbf{D}'$ be the list of domain names that are involved in these unlabeled traces but not belonging to $\mathbf{D}$. When recognizing unknown DNS query patterns, we consider domain names in $\mathbf{D}'$. Let k_c be the largest time scale for co-occurrence analysis. We refer to a time bin T_t^k (for $1 \le k \le k_c$) as a *relevant* time bin (resp. *irrelevant* time bin) if $\hat{z}_{t'}^{k_c} = 1$ (resp. $\hat{z}_{t'}^{k_c} = 0$), where $t' = \lceil t/\prod_{j=k}^{k_c} \alpha_j \rceil$.

Assume that there are m_k' unique QPAs at the kth time scale. Let γ_i^k be the ith unique QPA and Γ_i^k be a vector recording the presence/absence of γ_i^k in each time bin at the kth time scale. If γ_i^k occurs in T_t^k, we have $\Gamma_i^k(t) = 1$; otherwise we have $\Gamma_i^k(t) = 0$. We quantify the co-occurrence of γ_i^k and $\mathbb{A}$ using Fisher score:

$$\text{FS}(\gamma_i^k) = \frac{n_R(\mu_R - \mu)^2 + n_I(\mu_I - \mu)^2}{\sigma^2}, \tag{8}$$

where n_R (resp. n_I) is the number of relevant time bins (resp. irrelevant time bins), μ_R (resp. μ_I) is the mean value of $\Gamma_i^k(t)$ in relevant time bins (resp. irrelevant time bins), and μ (resp. σ) is the mean value and standard deviation of $\Gamma_i^k(t)$ over all time bins. We define the dominant co-occurrence QPA below.

Definition 2. *A dominant co-occurrence QPA at the kth time scale is a QPA, say γ_i^k, which is significantly concurrent with the user activity of interest such that FS(γ_i^k) $\ge \epsilon^k$. (ϵ^k is the co-occurrence threshold at the kth time scale)*

We refer to the QPA-vector in a relevant time bin as a dominant co-occurrence QPA-vector if there is at least one dominant co-occurrence QPA in this relevant time bin. If the QPA-vector in T_t^k is a dominant co-occurrence QPA-vector, it will be labeled by $c_t^k = 1$; otherwise, it is labeled by $c_t^k = 0$. The QPA-vector $\mathbf{x}_t^k$ and its label c_t^k constitutes a sample $\langle \mathbf{x}_t^k, c_t^k \rangle$. Similar to the method in Section III-B, we recursively train pattern upward mappers $\text{PUM}_1', \text{PUM}_2', \ldots, \text{PUM}_{k_c-1}'$ to characterize DNS query dynamics associated to domain names in $\mathbf{D}'$. Similar to the method in Section IV-A, we train multi-scale random forest classifier $\text{MRF}_1', \text{MRF}_2', \ldots, \text{MRF}_{k_c}'$ to recognize dominant co-occurrence QPA-vectors. Note that pattern upward mappers and multi-scale random forest classifier here are only applied in relevant time bins.

Given a trace of DNS queries from an end user, we first conduct the characterization of DNS queries and user activity identification to obtain the predicted label $\hat{z}_t^k$. According to $\hat{z}_t^{k_c}$, the relevant time bins are thus identified. We conduct dominant co-occurrence QPA-vector recognition in these relevant time bins. When $\hat{c}_t^k = 1$, we recognize that QPA-vector in time bin T_t^k is a dominant co-occurrence QPA-vector. Finally, we conclude that $\mathbb{A}$ occurs in T_t^k if $\hat{z}_t^k = 1$ or $\hat{c}_t^k = 1$.

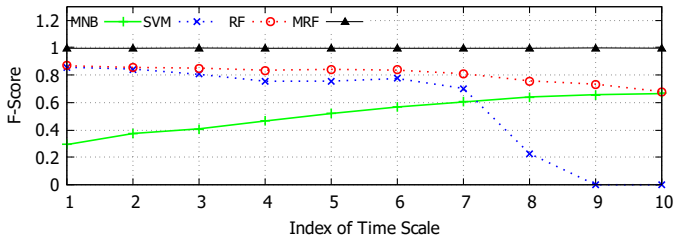
V. Evaluation and Application

We evaluate the effectiveness of our method using both synthetic data and real-world DNS traces, and then demonstrate its application in surveilling different user activities.

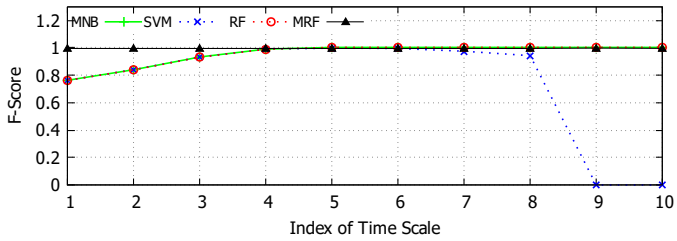
A. Evaluation on Synthetic Data

To examine whether our method can address the challenges presented in Section I, we first perform evaluation using synthetic data in the form of DNS queries, enabling us to deliberately embed challenges within the data for comprehensive evaluation. Consider a user activity (e.g., visiting a website), denoted as $\mathbb{A}$, which consists of two operations $\mathbb{A}_1$ (e.g., visiting one webpage) and $\mathbb{A}_2$ (e.g., visiting another webpage). Suppose $\mathbb{A}_1$ triggers a burst of DNS queries for domain names a and b, and $\mathbb{A}_2$ for domain names c and d. Assume the occurrences of $\mathbb{A}_1$ is a Poisson process, and so is $\mathbb{A}_2$. This is a reasonable assumption commonly used to model end users' DNS querying behavior [10] and other human behaviors [11].

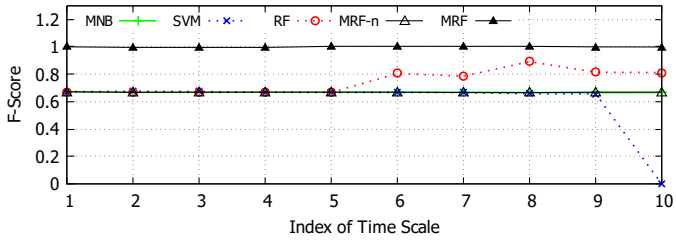
We aim at identifying $\mathbb{A}$ in the following four typical scenarios, wherein scenarios 1 and 2 involve the challenge of behavior ambiguity, and scenarios 3 and 4 involve the challenge of behavior polymorphism.



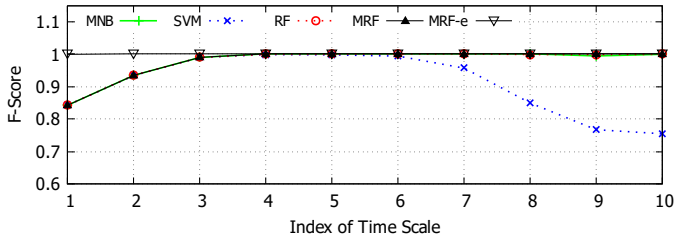
(a) *Scenario 1:* reusing domain names associated with $\mathbb{A}$.



(b) *Scenario 2:* reusing domain names associated with one operation of $\mathbb{A}$.



(c) *Scenario 3:* unknown reusing of domain names associated with $\mathbb{A}$.



(d) *Scenario 4:* unknown domain name queries of $\mathbb{A}$.

Fig. 7: Accuracy of identifying user activities in typical scenarios.

TABLE I: Evaluating the accuracy of identifying user activities of visiting websites. Each number denotes the F-score.

User Activities of Visiting Website	**MNB**	**SVM**	**RF**	**MRF** (our method)				
	1st time scale	1st time scale	1st time scale	1st time scale	3rd time scale	5th time scale	7th time scale	9th time scale
Baidu	0.6555	0.6779	0.6818	0.8635	0.9061	0.9368	0.9500	0.9362
Taobao	0.5939	0.8236	0.8721	0.8969	0.9074	0.9039	0.9320	0.9624
Jingdong	0.9224	0.9160	0.9461	0.9531	0.9532	0.9598	0.9652	0.9630
QQ	0.7995	0.5153	0.8019	0.8206	0.8881	0.9158	0.9555	0.9813
Netease	0.8833	0.9192	0.9066	0.9340	0.9234	0.9070	0.9102	0.8866
Amazon	0.4615	0.8462	0.7143	0.9565	0.9474	0.9333	0.9333	0.9231
Sohu	0.9838	0.9822	0.9830	0.9839	0.9718	0.9462	0.9259	0.9545
Youku	0.7368	0.7931	0.9000	0.9333	0.9130	0.9231	0.9375	0.9565
iQIYI	0.7888	0.8760	0.8800	0.9242	0.9636	0.9896	0.9959	0.9936
ifeng	0.5161	0.6857	0.6857	0.9231	0.9412	0.9231	0.9167	0.9000

Scenario 1: *Reusing domain names associated with* $\mathbb{A}$. All domain names associated with $\mathbb{A}$, (i.e., a, b, c, d) are also queried by another user activity $\mathbb{B}$, which consists of four operations that trigger DNS queries for domain names a, b, c, and d, respectively. Since each operation triggers a query for only one domain name, thereby not inducing a query burst (see DNS queries between t_3 and t_4 in Fig. 2).

Scenario 2: *Reusing domain names associated with one operation of* $\mathbb{A}$. For example, different from scenario 1, user activity $\mathbb{B}$ in this scenario reuses domain names a and b triggered by $\mathbb{A}_1$ (see DNS queries between t_5 and t_6 in Fig. 2).

Scenario 3: *Unknown reusing of domain names associated with* $\mathbb{A}$. We simulate the user activity $\mathbb{B}$ querying for a and c in the testing data, and $\mathbb{B}$ is not present in the training data.

Scenario 4: *Unknown domain name queries of* $\mathbb{A}$. We add new operations $\mathbb{A}_3$ and $\mathbb{A}_4$ for $\mathbb{A}$ in the testing data. $\mathbb{A}_3$ and $\mathbb{A}_4$ trigger DNS queries for domain names e and f, respectively.

To simulate the situation that the above scenarios are in a realistic network, we introduce background DNS queries (irrelevant to $\mathbb{A}$ and $\mathbb{B}$) that follow Poisson arrivals as noises. We also consider the impact of local DNS caches on DNS queries. That is, a domain name will not be queried if it has been cached in a local host and the cache record does not expire. τ is set to be 300 seconds, and inflation ratio is set to be 2 for all time scales.

Given a user activity of interest, identifying its presence/absence can be viewed as a binary classification problem in each time bin. Therefore, we consider binary classifiers that have been extensively used, including multinomial naive Bayes (MNB) classifier, support vector machine (SVM) classifier, and random forest (RF) classifier, as benchmark methods. At the kth time scale, we construct a feature vector by counting numbers of DNS queries for different domain names in $\mathbf{D}$ for each time bin. These feature vectors are the input of the classifiers. We choose F-score (aka. F_1 score) as the evaluation metric, because it can integrally reflect recall and precision in a single statistic. F-score ranges from 0 to 1, where bigger values indicate higher accuracy.

Figs. 7(a) to 7(d) report the experimental results in scenarios 1 to 4, respectively. In scenarios 1 and 2, we answer two questions: i) is our method effective in overcoming behavior ambiguity? ii) can benchmark methods also cope with behavior ambiguity at a proper time scale? In scenarios 3 and 4, we evaluate all methods in the face of unknown DNS query patterns in the testing data.

As shown in Fig. 7(a), MRF outperforms all benchmark methods, since it *consistently* identifies user activities with the F-score closer to 1 across all time scales. On the other hand, benchmark methods not only achieve noticeably *degraded* performance, but also have *inconsistent* performance across different time scales. Specifically, as the time scale increases, SVM and RF have decreased F-score, while MNB has an opposite trend. The former is because of the loss of behavioral information when the time scale is large. The latter is caused by MNB's independence assumption of DNS query arrivals for different domain names.

In Fig. 7(b), MRF is still the best. Note that benchmark methods have lower accuracy at small time scales (e.g., the 1st and 2nd time scales). It is caused by reusing domain names associated with one operation of $\mathbb{A}$, which triggers a burst of DNS queries for different reused domain names. Specifically, benchmark methods cannot distinguish between DNS query behaviors containing a single burst (i.e., at a small time scale), but can distinguish in the case of multiple bursts (i.e., at a large time scale). In addition, the accuracy of SVM decreases drastically as the time scale increases. It is because a larger time scale will introduce substantial noises and SVM is *not* robust against noises compared with other methods. This observation can be observed in *all* scenarios.

Fig. 7(c) demonstrates the effectiveness of introducing artificial negative samples when reusing domain names is not present in training but exists in testing. We can see that MRF successfully recognizes the unknown negative samples in testing and achieves a high accuracy in identifying $\mathbb{A}$. However, MRF without introducing artificial negative samples (i.e., MRF-n) and other benchmark methods fail to accomplish this task. Fig. 7(d) shows the performance in scenario 4. We can learn that MNB, SVM, RF, and MRF have decreased accuracy at small time scales, since they *cannot* recognize DNS query behaviors of new operations of $\mathbb{A}$. Fortunately, the enhanced MRF (i.e., MRF-e) proposed in Section IV-B can effectively handle this situation by recognizing the DNS query behaviors co-occurring with positive samples.

To summarize, our method can distinguish between user activities by exploiting their differences at multiple time scales. On the contrary, a proper time scale for benchmark methods is difficult to determine, since it may differ across user activities and even is time-varying in the same user activity. Compared to all benchmark methods, our method is also much more robust against new patterns of DNS query behaviors in testing.

B. Evaluation on Real-World DNS Traces

Using real-world DNS traces from our campus network, we further evaluate our methods in identifying user activities, where the ground-truth of each user activity is *uniquely* identified by the website that an end user visits.

The dataset contains 10-day network traffic of 159 end users (i.e., IP address without NAT) captured on the network border. We extract DNS queries for each individual end user. Meanwhile, we build the ground truth (i.e., which website an end user is visiting) by analyzing traffic (e.g., URL in HTTP request), and label it for each time bin at the 1st time scale. In our dataset, DNS queries only account for 0.166% of the total amount of network traffic in bytes. Identifying user activities from raw DNS queries provides a light-weight and thus more scalable solution to figure out what online users are doing.

Table I lists ten websites to identify in the first column. These websites are the most popular in the dataset. Following the same benchmark methods, parameter settings and evaluation metric as in Section V-A, we perform evaluation with four-fold cross-validation. Table I summarizes the results. For each user activity associated with a website, F-score values of MNB, SVM, and RF (at the 1st time scale) are presented from the second column to the fourth column, respectively. F-score values from the fifth column to the ninth column reveal the accuracy of the MRF at increased time scales. We observe that MRF consistently outperforms benchmark methods for every user activity. Note that identifying user activities of visiting "Baidu", "Taobao", and "QQ" is slightly less accurate than identifying others. This is probably because domain names of these websites provide many other network services, leading to complicated domain multiplexing.

C. Application

Surveilling user dynamics, such as how the number of users engaging in certain user activities varies within one day and across days, facilitates many problems such as user interest mining and content routing optimization.

We capture one-week DNS traces from more than $15,000$ users, and apply the proposed method in user dynamic surveillance of visiting different websites, including "Baidu", "QQ", "Taobao", and "Youku", in our campus network. To characterize the within-day dynamics and day-to-day dynamics, we divide the surveillance period into a series of time bins with the duration of 300 seconds. We identify user activities and compute the time-varying user number by counting users engaging in different user activities in each time bin. Fig. 8 reports the surveillance results. The numbers of users visiting different websites exhibit a significant diurnal pattern within each day. However, diurnal patterns have obvious "week effect". That is, diurnal patterns between weekdays and weekends tend to be different. In weekdays, user numbers grow rapidly in the morning, fluctuate during daytime, and drop at night. In weekends, the rapid growth of user numbers is put off to the afternoon. We believe this is due to different human schedules between weekdays and weekends. Moreover, diurnal patterns exhibit difference across various websites. For example, the percentage of users visiting "Baidu" at midnight is significantly larger than that of users visiting any other website; the percentage of users visiting "Youku" almost drops to zero at midnight.

VI. Related Work

Most existing studies of DNS traffic essentially focus on domain name reputation, for judging the likelihood of a domain name to be benign or malicious. For example, Antonakakis et al. built the dynamic reputation of DNS to discover the malicious use of DNS [4]; Jiang et al. identified suspicious activities in failure DNS queries [6]; Bilge et al. detected malicious domains by leveraging passive DNS analysis [7].

DNS caching behavior has also been studied, with an application to remote population estimation via active DNS cache probing. The basic idea is that, given a TTL value, a domain name that is more frequently cached in a DNS server tends to have a larger user base, and the cache entries can be

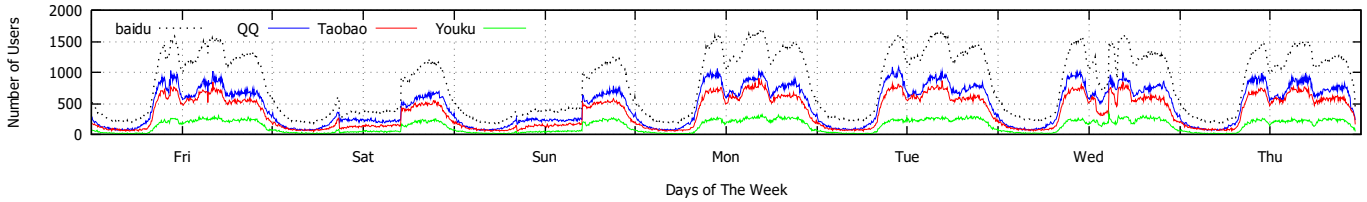


Fig. 8: Surveillance of user dynamics.

remotely probed to infer the popularity of the domain name. For example, Rajab et al. estimated website popularity and botnet population in [12]; Ma et al. found that the exponential distribution assumption in previous work is over-simplified and proposed the H-EXP estimator to improve the accuracy [13].

A few studies focus on DNS-based user behavior mining. Although our work falls into this category, it has different research objectives. For example, Herrmann et al. tracked users who have dynamic IP addresses and may migrate to new IP addresses by exploring DNS characteristics to uniquely identify a user [14]; Wu et al. leveraged probabilistic latent semantic analysis to cluster users that exhibit similar DNS characteristics, based on which they then recommend domain names for users [15]. However, they do not consider what activities a user in a cluster engage in.

A rich literature concerning what people are doing in network traffic analysis exists, such as deep packet inspection [16]–[18], website fingerprinting in anonymity networks [19,20]. Nevertheless, none studies this problem using raw DNS queries. Besides effectively identifying user activities (of visiting websites in our experiments), the benefits of using raw DNS queries are significant. Specifically, DNS has low data volume, thus light-weight and scalable when used in identifying user activities in large networks. Moreover, it has wide-ranging applications and is encryption free in normal applications, thereby promising in facilitating user activity identification of various types. In future work, we will apply our methods to identify more types of user activities.

VII. Conclusion

This paper takes the first step towards identifying user activities from raw DNS queries. We propose a multi-scale hierarchical characterization of DNS queries and a multi-scale identification method of underlying user activities. The proposed methods can effectively identify user activities from end users' DNS queries with a high accuracy, and overcome practical challenges including behavior ambiguity and polymorphism. Evaluations using both synthetic and real-word DNS traces demonstrated the effectiveness of our methods. The application of profiling user dynamics shows that our methods are promising in network monitoring.

Acknowledgement

This work is supported in part by National Natural Science Foundation (61602371, 61221063, 61202396), China Postdoctoral Science Foundation (2015M582663), Natural Science Basic Research Plan in Shaanxi Province (2016JQ6034), the Fundamental Research Funds for the Central Universities, Shaanxi Province Postdoctoral Science Foundation, Hong Kong General Research Fund (PolyU 5389/13E, PolyU 152279/16E), Shenzhen City Science and Technology R&D Fund (JCYJ20150630115257892), of China.

References

[1] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 315–327, 2002.

[2] J. Pan, Y. T. Hou, and B. Li, "An overview of dns-based server selections in content distribution networks," *Computer Networks*, vol. 43, no. 6, pp. 695–711, 2003.

[3] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 231–242, 2010.

[4] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for dns," in *Proc. USENIX Security*, 2010.

[5] J. Jung, A. W. Berger, and H. Balakrishnan, "Modeling ttl-based internet caches," in *Proc. IEEE INFOCOM*, 2003.

[6] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z.-L. Zhang, "Identifying suspicious activities through dns failure graph analysis," in *Proc. IEEE ICNP*, 2010.

[7] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: a passive dns analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 14:1–14:28, 2014.

[8] S. Yadav, A. Reddy, A. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, 2012.

[9] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive dns analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 14:1–14:28, Apr. 2014.

[10] A. Shimoda, K. Ishibashi, K. Sato, M. Tsujino, T. Inoue, M. Shimura, T. Takebe, K. Takahashi, T. Mori, and S. Goto, "Inferring popularity of domain names with dns traffic: Exploiting cache timeout heuristics," in *Proc. IEEE GLOBECOM*, 2015.

[11] C. C. Zou, D. Towsley, and W. Gong, "Email worm modeling and defense," in *Proc. IEEE ICCCN*, 2004.

[12] M. A. Rajab, F. Monrose, and N. Provos, "Peeking through the cloud: Client density estimation via dns cache probing," *ACM Transactions on Internet Technology*, vol. 10, no. 3, pp. 9:1–9:21, 2010.

[13] X. Ma, J. Zhang, Z. Li, J. Li, J. Tao, X. Guan, J. C. Lui, and D. Towsley, "Accurate dns query characteristics estimation via active probing," *Journal of Network and Computer Applications*, vol. 47, pp. 72–84, 2015.

[14] D. Herrmann, C. Banse, and H. Federrath, "Behavior-based tracking: Exploiting characteristic patterns in dns traffic," *Computers & Security*, vol. 39, pp. 17–33, 2013.

[15] J. Wu, X. Li, X. Wang, and B. Yan, "Dns usage mining and its two applications," in *Proc. ICDIM*, 2011.

[16] X. Chen, Z. Yuan, and Y. Xue, "Pbc: A novel method for identifying qq traffic," in *Proc. IEEE ICNC*, 2014.

[17] Q. Huang, P. P. Lee, C. He, J. Qian, and C. He, "Fine-grained dissection of wechat in cellular networks," in *Proc. IEEE IWQoS*, 2015.

[18] P. A. Branch, A. Heyde, and G. J. Armitage, "Rapid identification of skype traffic flows," in *Proc. NOSSDAV*. ACM, 2009.

[19] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at internet scale," in *Proc. NDSS*, 2016.

[20] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proc. USENIX Security*, 2014.