Public Review for

Exploring Domain Name Based Features on the Effectiveness of DNS Caching

Shuai Hao, Haining Wang

This paper focuses on the impact of ”one-time-only” names in DNS queries. The intuition is that records of hostnames accessed only one time will just age in the DNS resolver cache until they expire, therefore wasting the DNS cache resource. If names that are likely to be accessed only one time could be easily identified, the associated resource records could be either discarded faster, or not even been inserted in the cache. Authors propose thus an approach to identify hostnames that are likely to be one-time-only. They use a machine learning approach based on syntactical features extracted from the hostname string to train a classifier. They then test the DNS caching performance on actual data collected from a large campus network, and show that a caching policy that does not cache one-time-only names improve the performance by 10% wrt standard caching policies like LRU/FIFO.

The paper is well written and tackles an actual problem. The idea to classify hostnames based on their syntactical features is novel, and the results are very good. Trace driven simulations confirm the benefit of the idea, and provide a solid and thorough performance comparison that can improve the DNS resolver caching by a significant margin. Authors also commit to release both the dataset and the code to extract the data, thus helping the community to proceed further in this direction.

Public review written by
Marco Mellia
Politecnico di Torino

Exploring Domain Name-Based Features on the Effectiveness of DNS Caching

Shuai Hao[*†] Haining Wang[*]
[*]University of Delaware [†]College of William and Mary
{haos, hnw}@udel.edu

ABSTRACT

DNS cache plays a critical role in domain name resolution, providing (1) high scalability at Root and Top-level-domain (TLD) name servers with reduced workloads and (2) low response latency to clients when the resource records of the queried domains are cached. However, the pervasive misuses of domain names, e.g., the domains of "one-time-use" pattern, have a negative impact on the effectiveness of DNS caching because the cache has been filled with entries that are highly unlikely to be retrieved. In this paper, we investigate such misuse and identify domain name-based features to characterize those one-time domains. By leveraging the features that are explicitly available from the domain name itself, we build a classifier to combine these features, propose simple policy modifications on caching resolvers for improving DNS cache performance, and validate their efficacy using real traces.

CCS Concepts

•Networks → Network measurement; *Network protocols;*

1. INTRODUCTION

As one of the most important components of the Internet, the Domain Name System (DNS) provides vital mapping service for Internet users by translating domain names to IP addresses. Since DNS is a globally distributed database system, caching has been widely adopted in DNS infrastructures, where the acquired mapping results (i.e., the DNS resource records, RRs) will be cached locally to answer the following queries in a specific duration. DNS cache significantly reduces the resolution traffic along referral chains to interact with multiple name servers, resulting in a much shorter client-perceived delay and high scalability of DNS.

Due to its fundamental role for accessing Internet services, DNS traffic is the least blocked [21], and provides both attackers and developers with an attractive channel to transmit information. Thus, the misuse of domain names (either malicious or non-malicious) is widely observed on the Internet. On the other hand, since the cached objects in DNS resolvers are typically small, in some instances caches are not size-limited [17] and memory usage is relatively stable as the expired entries are being evicted. However, when serving a large group of users with a heavy load (e.g., in ISP or CDN/cloud providers), although modern DNS resolvers manage the memory well, they will still quickly consume the memory bytes and go into swap. Meanwhile, this may also cause performance problems on CPU if *cleaning-interval* is enabled to periodically check the stale records. To this effect, a fixed memory allocation is a common configuration [12], and the typical replacement policies (e.g., LRU and LFU) are employed to manage the cache usage [13, 14]. Therefore, it is critical to ensure that the cached RRs would be likely to be accessed again. Unfortunately, the pervasiveness of misused domains, e.g., *disposable domains* [12], causes the ineffectiveness of caching on resolvers since the cache is filled by records with very low or almost zero cache hit rates.

In this paper, we attempt to mitigate the negative effect on DNS caching caused by the domain name misuses, especially the "one-time-use" domains. Different from previous approaches, we do not pursue an accurate detection of domain misuses by employing the deep inspection techniques, such as behavioral features [10], alphanumeric characters-based metric [26], or entropy-based computing [21]. Instead, our key insight is that since most misused domains, either malicious or benign, tend to transmit information over DNS query names, the domain name itself may have distinct features that are explicitly available from individual queries and can be readily exploited for improving DNS cache performance.

Based on DNS trace logs captured in the resolvers of campus networks, we extract the *re-used* and *once-used* RRs. The reused RRs indicate that the queried domains are retrieved for multiple times, and the once-used RRs only appeared once in one trace. By analyzing a large amount of once-used entries, we observe that several explicit domain name-based features are capable of characterizing the reusability of domains. As such, we propose modified caching behavior to enhance the effectiveness of DNS caching by preliminarily excluding unreusable RRs. In order to validate their capability, we quantify the statistical properties of each feature and build a classifier that combines those features. The classification results demonstrate that the proposed modifications are able to prevent approximately 85% of once-used RRs from being cached while only less than 1% of reusable RRs are mistakenly kept out of the cache.

The remainder of this paper is organized as follows. We introduce the background of DNS caching and disposable domains in §2. We present the proposed domain name-based features on DNS caching in §3. We analyze collected datasets and build a classifier to validate the features in §4, and conduct a trace-driven evaluation in §5. We survey related work in §6 and conclude the paper in §7.

2. BACKGROUND

2.1 DNS Caching

Recursive DNS resolvers retrieve the name resolution results for clients and cache the received responses to answer the following queries. The duration that the cached records would be valid is specified by a *time-to-live* (TTL) value.

In standard TTL-based caching, the TTL value is set and handed out by the administrator of *authoritative* DNS record, and the cached entries are expunged after their TTLs expire. The duration for caching a negative response (e.g., NXDOMAIN, NODATA, *etc.*) is given by the TTL value of SOA record [4]. While TTL-aging-based behavior is legible, the violation of TTL is observed pervasively both in modern web browsers and DNS infrastructures [11]. For instance, many browsers and resolvers assign a minimum amount of seconds for holding an RR, and many resolvers trim the large TTL values to a default maximum value.

2.2 Disposable Domains

The use of DNS in various ways for which it was not originally designed has been observed for many years. For example, DNS is exploited as an effective covert channel for surreptitious communications [21, 25]. Moreover, Chen *et al.* [12] studied *disposable domains*, a more generic class of domain misuse where the query names are adopted to convey the "one-time signals." These domains are not necessarily malicious and are observed pervasively from various types of service providers, including popular search engines, social networks, CDNs, and security companies, and have being increased to a significant portion of queried domains on the Internet. Due to the "one-time-use" pattern and the increasing use of such domains, the DNS cache would be filled with entries with *near-zero* hit rates. Our work is mainly built on the analysis of these entries for disposable domains.

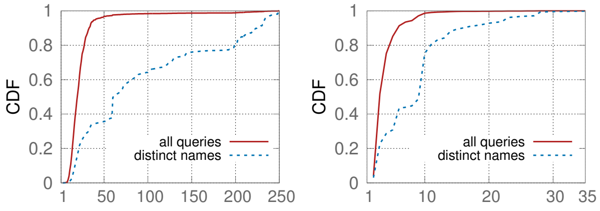
3. DOMAIN NAME-BASED FEATURES

Domain names are human-readable and easy-to-remember character sets. However, the once-used domains exploit the query names as a communication channel. One of our insights is that such misused domains are encoded automatically to convey formatted information and should have significantly different patterns on their domain names.

Therefore, we consider the possibility of characterizing the once-used domains, and then exploit the derived features to filter out the disposable domains. The removal of such once-used domains from the DNS cache will improve its performance because the pervasive use of misused domains causes the DNS cache to be occupied by those entries that are highly unlikely to be reused. Figure 1 presents the preliminary examples that motivate our feature selection. We simply plot the distributions of the query name length and the subdomain depth (i.e., the number of subdomains), respectively, for *all* observed query names and *distinct* domain names from one of our trace logs (§4.1). It is evident that (1) most repeatedly appeared domains have a short name and limited subdomain depth, and (2) a significant portion of domains have a long query name and a large number of subdomains. This implies that, under a limited memory space, discarding those entries with long names or deep subdomains would save more space in the cache to store the entries with a higher possibility of being reused, and thus effectively improving the caching effectiveness.

Based on the analysis of large amounts of once-used domains, we identified that the domain name-based features, such as the two features above, are able to characterize the caching behavior of domains without the help of sophisticated features used in the detection of malicious domains (e.g., the behavioral features in EXPOSURE [10] and statistical features in Notos [6]). As a result, we propose the following features and explain why they may affect caching effectiveness. All but the first one have not yet been characterized in the analysis of DNS, and none of them have been studied in the context of caching performance.

- `F1`: **Length of query name**: Since most of the once-used domains tend to send messages over DNS queries, those domains naturally have (much) longer query names to pack as much information as possible and are hardly reusable.
- `F2`: **Length of the longest subdomain name**: Similar to the query name, the individual lower-level name (i.e., the string representing a subdomain) could also be larger than a legitimate subdomain name that tends to be "easy-to-remember."



(a) Query Name Length (b) Subdomain Depth

Figure 1: Example of distribution for lengths and depths.

- `F3-a`: **Subdomain depth**: To report information over DNS, the implications of the domain names must be easily recognized by the receivers. In doing so, the domain separator, i.e., the period "**.**", would naturally be employed to format the domains to give the name strings meaningful information, resulting in a deep subdomain level, i.e., a large number of subdomains in one query name.
- `F3-b`: **Number of format fields**: Like the period specifying the subdomain level, we also observed that the hyphen "**-**" is widely used as field separator to format messages in one subdomain name.

Due to the function similarity, in this paper we treat both periods (`F3-a`) and hyphens (`F3-b`) as equal format separators and employ the same term, "number of format fields" (`F3`), to represent the number of strings separated by either "**.**" or "**-**".

- `F4`: **Number of fields with unusual lengths**: To represent various pre-defined types of information inserted in specified positions, the length of format strings would vary widely, and many fields would be either unusually long or unusually short. We consider such domains quite hard to reuse. Thus, we define a metric of the sum of the number of long-format-fields (L-FF) and short-format-fields (S-FF) within one query name to identify such a feature.

Figure 2 shows the sample of domains with the explicit features. Note that these features, e.g., the length of query names, do not necessarily indicate malicious purposes [21]; in fact, most of them are benign. However, they indeed indicate the usage of "signaling" and thus imply the high possibility of a "one-time-use" pattern. Accordingly, there would be a strong correlation between each feature and once-used RRs in the cache. By exploiting these features, we can revise the caching policies to proactively prevent those RRs that are less likely to be reused from being cached, such that the effectiveness of DNS caching could be significantly improved.

Methodology. For the proposed features, we characterize the properties of re-used and once-used domains, train a classifier to classify the entries, and conduct a trace-driven simulation to validate their efficacy in caching. In the feature validation (§4.2) and classification (§4.3), the analysis simply relies on domain names and assumes implicitly that both the cache size and TTL values are unlimited. This assumption simply creates an ideal scenario for caching RRs, and cache hits are not limited by the cache size and TTL values. The simulation (§5.2) runs within a resolver program that caches the entries according to the classification results and common practices on modern DNS resolvers.

4. MEASUREMENT ANALYSIS

4.1 Dataset

The datasets used in the study are the trace logs of outgoing DNS queries captured at local DNS servers at the College of William and

	F1	F2	F3	F4
`0.28e.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.27ebb55310eb3785446c4874fefabd756df2ff2361 ↪ 2f512d4ddc7c9a3ba70d2.b.f.01.s.sophosxl.net`	●	▲	★	▼
`f6a9fc3efdffd146663f44de1c071c0f548c5653.p.00.s.sophosxl.net`		▲		
`p4-hnpieeqf4ghwk-ab6awvsfyjixzdw7-629649-i1-v6exp3-v4.metric.gstatic.com`		▲		▼
`0107c2e22801.t-1436279541.i45381316.04e6d5fe76b6755a1edd7532a34ec877-27718-htm.fp.bl. ↪ barracudabrts.com`	●	▲	★	▼
`f6a556b42904cfd118c0-553848320f40ae46bf95fbb566795773.r63.cf2.rackcdn.com`	●	▲		
`ada1a1b36908553d09b507630a84f2606.profile.lhr5.cloudfront.net`		▲		
`e4cbe5a2594fa1dd8306275ef1e7e4df.azr.msnetworkanalytics.testanalytics.net`	●	▲		
`b-0.19-a3000008.8011081.1644.981.3ea3.410.0.q3j4p1csa3z1seadcmamni9295.avts.mcafee.com`	●		★	▼
`0.0.0.0.1.0.0.4E.c7eijlj4gwumadva92s62m52ri.avqs.mcafee.com`			★	▼
`i1-j1-18-15-4-114-3425533130-i.init.cedexis-radar.net`		▲	★	▼

Figure 2: Sample of domains with the domain name-based features.

Table 1: Summary of datasets

Dataset	Dates	Size (per day)	Features[†]	# Records	# Distinct name	# Re-used	# Once-used
`WM`	6/25/2015 - 7/8/2015	1.1 - 2.3GB	N, S, K, T	192,251,799	4,470,732	619,045	3,851,687
`UD`	12/8/2015 - 12/23/2015	2.4 - 3.9GB	S[‡], T	1,011,877,341	13,179,395	2,852,021	10,327,374

[†] Features: Query Name (**N**), Structure of Domain Name (**S**), Query Type (**K**), and Query Timestamp (**T**)
[‡] In `UD` dataset, the actual query names are anonymized but remain distinguishable to identify reappearance. Meanwhile, the structure of domains is given by the length of each format field (e.g., `a.b.c-d-e.f.g`, where each letter represents the length of each field).

Mary (`WM`) and the University of Delaware (`UD`) over a period of two weeks. We summarize the datasets in Table 1.

The trace logs from campus vantage points have two limitations. First, the traces may not reflect the dynamics of domain names observed at ISP's DNS servers. However, due to similar patterns of disposable domains from a large-scale ISP dataset reported in [12], we believe that the proposed policies would also be effective in ISP's DNS cache, especially due to the heavy load on their resolvers. Also, we removed the RR entries that retrieve local domains. Second, the length of our trace logs is limited. We believe, however, they are still capable of demonstrating the typical cache usage patterns of local resolvers because the origin TTL values of A/AAAA records are typically shorter than one day [15].

4.2 Feature Validation

Given the heuristics presented in §3, we validate our speculation that these explicit domain name-based features will help to improve the effectiveness of DNS caching. From Table 1, we can see that 86.15% and 78.63% of queried domains appeared only once in each dataset, which is similar to the results of identified disposable domains [12]. In our study, however, we do not attempt to achieve high detection accuracy in identifying such a class of domains. Instead, we focus on exploring the efficacy of proposed features on improving the *overall* cache performance by avoiding a large number of once-used records being filled in the cache.

To validate each proposed feature based on its factual caching effect, we *tentatively* derive a threshold to measure the fractions of excluded domains. We then leverage a learning module to train our datasets and build a classifier that combines the proposed features.

`F1`: **Length of Query Name.** Figure 3(a) shows the distributions of the query name length in which the two classes of domains are clearly distinguishable from one another. The majority of once-used domains apparently have much longer name lengths than re-used domains.

To exclude the useless RRs, we start to consider a tentative threshold at 50 bytes of query name length. In the `WM` dataset, a threshold of 50 bytes would exclude 81.10% of once-used domains and 1.61% of re-used domain, resulting in 69.87% of overall entries being rejected from the cache. By gradually raising the threshold, at the length of 100 bytes we observe that 0.027% of reusable domains are mistakenly dropped while 44.99% of once-used domains (38.77% of RRs in total) are discarded. Similarly in the `UD` set, with the length of 100 bytes, 0.019% of re-used entries are mistakenly kept out of the cache while 32.58% of once-used domains (25.53% of RRs in total) are dropped. Such results indicate that rejecting domains with large names would significantly reduce the waste of cache space but keep the cache hit ratio at the same level.

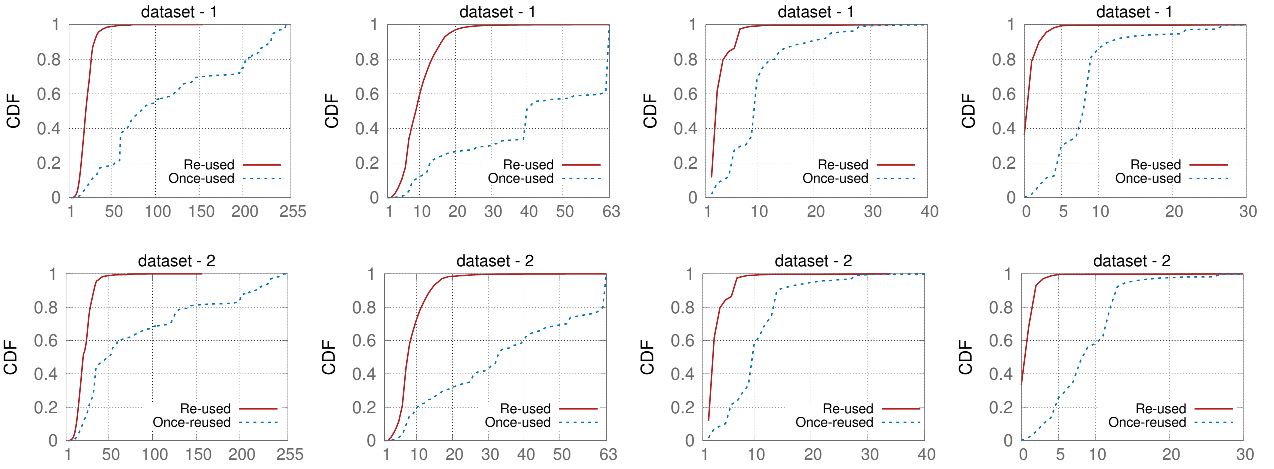
`F2`: **Length of the Longest Subdomain Name.** Figure 3(b) demonstrates that the large strings are widely adopted in once-used domains. Specifically, with each dataset, we identify 3.03% and 1.51% of re-used entries, as well as 73.40% and 68.05% of once-used entries, include one subdomain with the length of more than 20 bytes, respectively. If we increase the threshold to 30 bytes, the fractions of re-used domains will decline to 0.39% and 0.30% while the fractions of once-used domains remain at 69.86% and 57.22%, respectively. Thus, a subdomain name length greater than 30 bytes would strongly indicate that the domain is once-used, with little chance of being a useful entry if cached.

`F3`: **Number of Format Fields.** Figure 3(c) presents the distribution of the total number of format fields. It is easy to identify that a threshold of 10 is capable of distinguishing the reusability for each class of domains. Using this threshold, we can exclude 0.59% and 0.79% of re-used entries, and 31.17% and 42.39% of once-used domains, respectively, from each dataset; overall, around 25% and 37% of entries would be discarded from the cache, respectively.

`F4`: **Total number of L-FF and S-FF.** In order to profile the total number of the *long-format-fields* (L-FF) and the *short-format-fields* (S-FF), we first empirically determine the specific values of the length to define the L-FF and S-FF. Since most of *TLD*s include one or two fields with two or three characters, we define the S-FF as the fields with characters less than or equal to three.[1] Also, we investigate the distributions of `F4` by varying the length of L-FF, and observe that a length of 10 is sufficient to demonstrate the distinct statistical properties for this feature.

Figure 3(d) shows the distribution of the sum of L-FF ($\geqslant$10 bytes) and S-FF ($\leqslant$3 bytes). With a clear threshold observed at five, we identify that 0.61% of re-used and 70.03% of once-used domains in `WM`'s dataset (60.33% of RRs in total), and 0.40% of re-used and 74.55% of once-used domains in `UD`'s dataset (63.97% of RRs in total), would be discarded. As a result, this would exclude

[1] A more accurate approach may need to exclude the TLDs (e.g., `.com` and `.co.uk`) and SLDs (second-level domains, such as `msn` and `cnn`) since they may be regarded as the S-FF. However, we observe that checking the entire domain names has already produced effective results, so we decide to use a simple way to avoid introducing additional steps to identify the SLDs.



(a) DNS Query-Name Length (b) Longest-Subdomain Length (c) Number of Format Field (d) Total Number of L-FF and S-FF

Figure 3: Distribution of domain name-based features for `re-used` and `once-used` domains.

Table 2: Percentage of mis-classified instances

	Decision Tree			Random Forest `(ntree=5)`		
	D	R	O	D	R	O
`WM`	16.07%	0.26%	13.88%	12.93%	0.19%	11.16%
`UD`	13.91%	0.98%	14.13%	11.89%	0.34%	12.18%

`D`: Disposable, `R`: Re-used, `O`: Overall

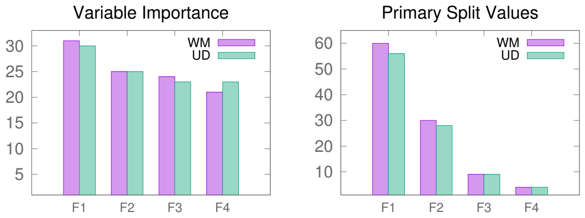


Figure 4: Training Results (with Decision Tree).

the majority of the useless entries but have little negative impact on caching reusable domains.

4.3 The Classifier

To validate the efficacy of the combination of proposed features, we train the datasets with both *decision tree* and *random forest* models [5] by using the `rpart` and `randomForest` packages in R, respectively. Note that we leveraged the *class-weights* (i.e., the `parms` parameter in `rpart` and the `classwt` parameter in `randomForest`) to handle unbalanced class sizes in our datasets.

Ground truth. Since it is (almost) impossible to have a ground truth that identifies the "disposable domains," we label the once-appeared domains extracted from the datasets as disposable domains. As such, our labels correspond to those assigned by an oracle with perfect knowledge. Although the domain unpopularity may cause mis-labeling (i.e., some unpopular domains may be mislabeled as disposable), our labels are the acceptable approximation to the ground truth in practice, especially given more than thousands of users from each campus network. Moreover, mis-labelling a rarely re-used domain as disposable would have a marginal impact on practical caching performance, being it likely to be evicted before reappearing.

Evaluation of the classifiers. Each dataset is divided into mutually exclusive training and testing partitions, where 66% of the dataset is used for training and the rest is used for testing. With the random forest model, we observe that the benefit cannot be achieved with the number of constructed trees higher than five. Table 2 lists the percentage of incorrectly classified instances using the combination of all features.[2] Note that we aim to improve the caching effectiveness in two folds: (1) effectively reject the useless entries, and (2) minimize the negative impact on the reusable ones. The results in Table 2 demonstrate that we can achieve both goals in the classification processes. They also indicate that, although a simple decision tree tends to overfit the training set, it is capable of producing accurate results when applied to the classifier constructed by the combination of proposed features. More specifically, 85% to 88% of once-used RRs are correctly labeled and expelled from the cache, while only 0.2% to 1% of re-used RRs are incorrectly classified in the `WM` and `UD` datasets, respectively.

The unbalanced (but expected and positive) results can be interpreted by the observation that the re-used entries have more consistent and concentrated distributions of features, while the extracted features from once-used entries exhibit more diffused distributions.

Figure 4 shows the index of variable importance and the primary split values in the decision tree training for each dataset, which illustrates that (1) all the features play important roles in the classification (the importance index varies from 21 to 31), and (2) although the primary split values are more aggressive than the thresholds derived from any single feature (§4.2), we can further lower error rates by using the combination of features.

5. TRACE-DRIVEN SIMULATION

In §4, we demonstrate that the explicit domain name-based features are useful to infer the reusability of RRs. However, in practice, the resolvers behave slightly differently due to the presence of TTLs. The re-used entries may still cause cache misses as the cached RRs are expired. Meanwhile, some mis-classified reusable entries may not affect the caching performance since many of them have a lower possibility to be retrieved again *within* the duration of TTL. In this section, we apply the classifier in §4.3 with the combination of proposed features to conduct a trace-based simulation[3] to evaluate the effectiveness of proposed policies.

[2] We explored different combinations of feature sets and found that using all features in the classification can achieve the minimum error rate.

[3] We only perform the simulation with `WM`'s trace since the actual domains have been anonymized in `UD`'s trace.

Table 3: Summary of RR Types (%)

A	AAAA	TXT	PTR	SRV	SOA	NS	other†
70.80	15.33	5.09	3.96	3.86	0.83	0.08	0.06

† MX, ANY, NAPTR, DS, DNSKEY, CNAME, AFSDB, and AFXR

5.1 Implementation

We implemented the proposed caching policies in a simulated resolver program modified from `djbdns` [2], in which the decisions employ the classification results from §4.3. Our resolver program follows the standard TTL model, i.e., do not assign a default minimum TTL value. The duration of negative caching is subject to the TTL values of SOA records [4]. Moreover, we do not set the *cleaning-interval* to periodically expel the stale records due to the use of sophisticated memory management in modern resolvers [3]. Only when hitting the cache limit will some entries be prematurely evicted from the cache (e.g., using LRU replacement policy).

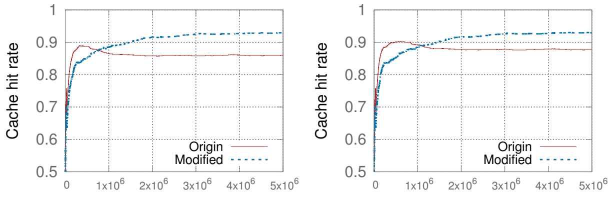
Types of RRs. First we study the caching properties of different types of RRs. In particular, we examine whether they need to be given discriminative considerations when caching in resolvers. Table 3 lists the breakdown of the types of queries. The SOAs are treated the same as A/AAAAs, i.e., caching such RRs according to the proposed policies in §4. Other unspecified types of RRs are not particularly studied because of their small amounts of queries.

- **TXT records**: We identify that only 0.01% of TXT records have been reused, and indeed observe that the TXTs are being used as an information channel. We observed similar distributions of proposed features in TXTs and thus use the proposed policies for caching TXTs.
- **Reverse lookup queries (PTRs)**: We do not apply the modifications on PTRs since it is rarely misused, and we do not observe the studied features taking effect on PTRs.
- **Service records (SRVs)**: We identify that most SRVs (97.46%) are involved with local queries that have been removed in our study (§4.1). Like PTRs, we also observe that the studied features have no impact on SRVs' caching effectiveness either, and thus we will not apply the polices on SRVs.
- **NS records**: Caching NS records can significantly enhance the efficiency of DNS and reduce the load on name severs [17]. Also, the number of NS records is much smaller than the other types of RRs above. Thus, there is no need to apply the policies on NS records. In fact, no NS records would be excluded from caching if the proposed policies were applied.

5.2 Results

We now evaluate the effectiveness of the proposed policies given a fixed cache memory allocation. To simplify the assessment, we define the cache allocation by the number of RRs. We input the RRs to a cache file and then examine the cache hit rate, which is calculated as the ratio between the number of cache hits and the total number of retrieved RRs.

We need to determine how many RRs should be cached to represent a real scenario for the evaluation of our proposed policies. Jung *et al.* [17] identified that the DNS cache hit ratio is between 80% and 87%. Thus, we choose our cache size as the number of cached RRs that can achieve a similar cache hit ratio. To this end, given the moderate size of our dataset and an FIFO replacement policy, i.e., simply remove the oldest entry when the cache runs out of space, we observe that a size of 100,000 entries would have a cache hit rate of about 86%. Therefore, we set the cache size to 100,000 RRs in our simulations. Note that this setting is derived from local campus networks, and in ISPs' DNS servers, a larger cache size is needed to handle the larger number of DNS queries.



(a) Cache hit rate with FIFO

(b) Cache hit rate with LRU

Figure 5: Distribution of cache hit rates. The X-axis represents the number of entries read into the resolver.

FIFO. We first evaluate proposed policies with the FIFO caching scheme, which is still widely used in popular DNS resolvers such as `djbdns` [2]. Figure 5(a) presents the measured cache hit rates under the FIFO, with and without proposed polices, respectively. We observe that the modified caching policies can improve the cache hit rate by about 8% with a cache size of 100,000 entries.

Pseudo-LRU. We then evaluate the proposed policies with a simplified pseudo-LRU that leverages one bit to store the cache status for each entry (i.e., the MRU-bit). When a cache hit occurs for an entry, its cache bit is set to 1. When the cache is full, the oldest entry with the cache bit 0 is evicted. When the cache bits of all cached entries have been set to 1, all the bits are cleaned to 0 except for the last one. Figure 5(b) shows the measured cache hit rates with the LRU replacement scheme. We observe that the proposed caching modifications improve the cache hit rate by about 7%. Compared with FIFO, LRU without our policies can increase the cache hit rate by about 2%. With the proposed modifications, both FIFO and LRU can increase the cache hit rate to 92% - 93%.

5.3 Discussion

TTL values. The lower TTL values have been observed in both malicious domains [10] and disposable domains [12]. However, the domain owners are free to set the TTLs and have been switching to larger TTL values (most of them have a TTL of 300s [12]), resulting in a larger duration of the useless entries being regarded as valid in the cache. Meanwhile, since modern resolvers have made the cleaning-interval obsolete [3], caching the once-used domains even for a short time still degrades the performance. This is because the cache is filled with such useless entries, and no space is left for caching useful ones. Thus, TTL may not be a reliable indicator of the caching effectiveness, and we do not consider it to be a metric to quantify the caching behavior.

Counteraction. One may be concerned that the domain owners could circumvent our polices by changing the structure of domain names. However, we believe the modifications will not provoke them to seek more sophisticated approaches since those "one-time use" domains have accomplished the communication mission, and the developers using such approaches would not care if their DNS responses are cached.

6. RELATED WORK

DNS Caching and TTL characterization. Pang *et al.* [20] presented a comprehensive study on DNS, and [19] observed that a significant fraction of web clients and LDNSes do not honor DNS TTLs. Callahan *et al.* [11] passively monitored DNS traffic within a residential network to profile the modern DNS behaviors and properties. They also observed that web browsers do not adhere to the given TTLs, and CDNs tend to shape traffic with shorter TTLs.

Jung *et al.* [17] presented a detailed analysis on DNS traces to evaluate the client-perceived performance and the effectiveness of DNS caching. They [18] then presented an analytic method of modeling the cache hit rate given consistent TTLs. Choungmo-Fofack *et al.* [14] studied the DNS cache hierarchy in which the TTLs are overridden by the local values.

Cache modifications. Shang *et al.* [23] proposed an approach to improve the DNS caching by letting the ADNS's response piggyback extra resolution results for future queries predicted by site usage and DNS history. Cohen *et al.* [13] studied the proactive renewal policies in LDNSes, where the expired records are reused to answer the queries and are validated with a concurrent query. Similarly, Ballani *et al.* [9] proposed a minor change in caching behavior of DNS resolvers to mitigate the DNS DoS attacks, where the expired records are stored in a separate "stale cache" and reused to answer the queries unresponded by the authoritative servers.

Malicious domain detection. There have been studies to understand and detect the malicious domains. Hao *et al.* [16] examined the features that may indicate malicious purposes of a domain during its registration. Antonakakis *et al.* [7, 8] and Yadav *et al.* [26] proposed methods to detect dynamically generated malicious domains in DNS traffic.

Bilge *et al.* [10] built a passive analysis system extracting 15 features to detect malicious domains. Similar to our work, two of the features are domain name-based: (1) the percentage of numerical characters and (2) the ratio of the length of LMS (longest-meaningful substring) to the total length of a second-level domain. The most salient difference is that our work studies caching behaviors, regardless of whether a domain is malicious or benign. Also, the features we used are simpler and more straightforward (e.g., the second feature above requires checking the English dictionary).

7. CONCLUSION

We presented an empirical study on the domain name-based features of DNS queries and exploited these features to improve DNS cache performance. The identified features, including the length of a query name, the length of the longest-subdomain, and the number of subdomains or format fields, are explicitly available from a domain name itself, without involving deep inspections. Whereas the features do not indicate malicious purposes, the majority of domains with such properties are indeed associated with the one-time-use pattern and would be highly unlikely to be reused in a DNS cache. Our analysis and simulation demonstrate that proactively rejecting such domains from the cache can improve the *overall* effectiveness of DNS caching. Finally, we make one of the traces used in this paper publicly available [1], with proper anonymization while being able to perform training and simulation.

8. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and our Area Editor, Marco Mellia, for their detailed and insightful comments, which help to improve the quality of this paper. We would also like to thank Clarke Morledge, Roger O. Clark, Thomas M. Baxter, Ben Miller, and other IT team members at the College of William and Mary and the University of Delaware for the DNS data collection.

9. REFERENCES

[1] Feature Datasets. `https://doi.org/10.5281/zenodo.161198`.

[2] D. J. Bernstein. `djbdns`. `http://cr.yp.to/djbdns.html`.

[3] BIND 9 Configuration Reference. `http://ftp.isc.org/isc/bind9/cur/9.10/doc/arm/Bv9ARM.ch06.html#id2586632`.

[4] RFC 2308: Negative Caching of DNS Queries (DNS NCACHE). `https://tools.ietf.org/html/rfc2308`.

[5] P. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining. *Addison-Wesley*, 2006.

[6] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a Dynamic Reputation System for DNS. In *USENIX Security Symposium*, 2010.

[7] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon. Detecting Malware Domains in the Upper DNS Hierarchy. In *USENIX Security Symposium*, 2011.

[8] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-based Malware. In *USENIX Security Symposium*, 2012.

[9] H. Ballani and P. Francis. Mitigating DNS DoS Aattacks. In *ACM Conference on Computer Communication Security (CCS)*, 2008.

[10] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[11] T. Callahan, M. Allman, and M. Rabinovich. On Modern DNS Behavior and Properties. *ACM SIGCOMM Computer Communication Review (CCR)*, 43(3): 7-15, July 2013.

[12] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee. DNS Noise: Measuring the Pervasiveness of Disposable Domains in Modern DNS Traffic. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2014.

[13] E. Cohen and H. Kaplan. Proactive Caching of DNS Records: Addressing a Performance Bottleneck. *Journal of Computer Networks*, 41(6): 707-726, Apr. 2003, .

[14] N. Choungmo Fofack and S. Alouf. Modeling Modern DNS Caches. In *EAI International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*, 2013.

[15] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An Empirical Reexamination of Global DNS Behavior. In *ACM SIGCOMM*, 2013.

[16] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck. Understanding the Domain Registration Behavior of Spammers. In *ACM Internet Measurement Conference (IMC)*, 2013.

[17] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Transactions on Networking*, 10(5): 589-603, Oct. 2002.

[18] J. Jung, A. W. Berger and H. Balakrishnan. Modeling TTL-based Internet Caches. In *IEEE INFOCOM*, 2003.

[19] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. On the Responsiveness of DNS-based Network Control. In *ACM Internet Measurement Conference (IMC)*, 2004.

[20] J. Pang, J. Hendricks, A. Akella, R. D. Prisco, B. Maggs, and S. Seshan, Availability, Usage, and Deployment Characteristics of the Domain Name System. In *ACM Internet Measurement Conference (IMC)*, 2004.

[21] V. Paxson, M. Christodorescu, M. Javed, J. Rao, R. Sailer, D. L. Schales, M. Stoecklin, K. Thomas, W. Venema, and N. Weaver. Practical Comprehensive Bounds on Surreptitious Communication Over DNS. In *USENIX Security Symposium*, 2013.

[22] P. Porras, H. Saidi, and V. Yegneswaran. A Foray into Conficker's Logic and Rendezvous Points. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.

[23] H. Shang and C. E. Wills. Piggybacking Related Domain Names to Improve DNS Performance. *Journal of Computer Networks*, 50(11):1733-1748, Aug. 2006.

[24] D. Wessels, M. Fomenkov, N. Brownlee, and K. C. Claffy. Measurements and Laboratory Simulations of the Upper DNS Hierarchy. In *Passive and Active Measurement Conference (PAM)*, 2004.

[25] K. Xu, P. Butler, S. Saha, and D. Yao. DNS for Massive-Scale Command and Control. *IEEE Transactions on Dependable and Secure Computing*, 10(3):143-153, 2013.

[26] S. Yadav, A. K. K. Reddy, A.L. Narasimha Reddy, and S. Ranjan. Detecting Algorithmically Generated Malicious Domain Names. In *ACM Internet Measurement Conference (IMC)*, 2010.