Summary Review Documentation for

“DNS to the rescue: Discerning Content and Services in a Tangled Web”

Authors: I. Bermudez, M. Mellia, M. Munafo, R. Keralapura, A. Nucci

Reviewer #1

Summary: In this paper, the authors exploit a simple observation: Typically any request to any server is preceded by a DNS query to resolve the hostname. They exploit this observation by keeping track of DNS queries (in effect replicating client DNS cache) in the middle of the network and then associating server IP addresses in packets with appropriate FQDNs. Once FQDNs are available, the idea is that traffic monitoring becomes easier since they can mine this quite a bit. For example, one can figure out which set of server IP addresses correspond to a given FQDN. And the reverse as well, that is, which FQDNs share the same IP addresses. They can even mine tokens within FQDNs to figure out protocol (e.g., SMTP may have the word smtp in the FQDN smtp.example.com). Using real data collected from ISPs, they show how this information may be useful in measurement and monitoring purposes.

Strengths:

- Idea of using DNS queries for figuring out the FQDNs associated with servers is kind of clever and nice.
- The idea has significant potential in terms of applicability for various types of traffic mining.
- Well written and thorough evaluation.

Weaknesses: Goals are a little unclear. What in the end they are trying to do is a bit open-ended. Several of the insights are kind of well-known.

Comments to authors: The idea of using DNS for identifying the FQDN and subsequently the associated service and resource is kind of trivial in hindsight, but not so obvious. Given all the literature in traffic classification, I have not seen anybody exploit this to build a system around it. So, in that sense, I give credit to the authors for making and exploiting this observation.

I thought that the paper lacked clarity in identifying the problem however. The paper starts of with two compelling application scenarios, which really got me excited:

The first one involves classifying services or applications such as Google Docs, Blogger and Youtube. Unfortunately, the proposed solution of identifying FQDNs is not sufficient. Imagine two parallel tabs open in a browser with one tab playing youtube and the other blogger. I would suspect that since most of the web services are mashups, you would see a temporally-mixed series of FQDN accesses from these two different services. Recovering which set of FQDNs are part of which service is almost impossible using the proposed approach.

The second application scenario is blocking certain services such as Zynga where as allowing Dropbox, both of which use Amazon EC2 for hosting certain portions of their webservice. Again, the same problem is similar. If two services were accessed simultaneously with FQDN1 and FQDN2 pointing to same server IP, I am not sure the proposed solution can easily differentiate it, even with a replica of client’s DNS cache at the vantage point.

In either of these cases, I thought the paper did not complete the loop for me. It started out with these great examples where their observation makes sense and would be interesting. However, they did not follow up either of these scenarios and exhibited that their system DN-Hunter solves these problems.

The other limitation is that it will only work for applications that depend on the DNS system, as the authors pointed out. To me, this is not such a big deal since the problem is very difficult to solve perfectly for all applications anyways.

One other limitation, which was also explicitly mentioned, was that it requires access to DNS queries/responses. I am more concerned about this limitation since in the paper, the authors collect data from a PoP. Why will the DNS traffic be visible to the PoP since a client with IP A will contact a local DNS server for resolving various DNS traffic that will not cross the PoP boundary. Even if the local resolver will eventually contact the root server and so on, association between the resolver’s IP address say IP B and IP A will be difficult at the PoP level. I am not sure how the authors solve this problem since they do not mention this further in this paper.

The advanced analytics portion is interesting, but entirely straightforward. Essentially, they talk about splitting the FQDN - IP address mapping stuff collected from DNS and ask a lot of simple questions, such as the set of IP addresses corresponding to a FQDN and so on that can be easily answered from here. The automatic service tag extraction is interesting, but I am not sure how can use it in a real system. Since all it is showing is that there is correlation between port number and the service tag in the 3rd level token in the FQDN. How does one use this for traffic identification?

Reviewer #2

Summary: The paper describes a traffic probe called DN-Hunter that passively sniffs DNS responses and associates traffic flows with target DNS names. This provides better visibility, e.g., into encrypted traffic, enabling to enforce finer traffic policies. Most of the paper shows examples of how DN-Hunter is useful to map the spatial distribution of services, the content offered by CDNs, and the service of traffic flows. I was quite convinced that DN-Hunter is useful.
**Strengths:** I like the approach of annotating traffic with DNS information. It provides useful additional information about used services. It totally makes sense.

The paper gives some nice examples (Figures 7 & 8) of how DN-Hunter provides better visibility into traffic by mapping how certain cloud and CDN services are offered.

The paper is well written.

**Weaknesses:** The main idea of the paper has been recently proposed in [13].

**Comments to authors:** Besides, the authors argue that "Surprisingly, the results presented in this section for motivating the need for a solution like DN-Hunter could not have been produced if we did not have DN-Hunter!" This is not really true. Similar studies of how tangled the web is have been possible with active DNS probing (e.g. [9]). Besides, it is easier to probe a larger number of servers (and therefore capture a more complete picture) than to collect and analyze passive traces from many locations.

Sections 4 and 5 seem to go a bit into the direction of characterizing how tangled the Internet is. However, a characterization study in not well inline with the main goal of providing visibility into traffic to enable finer policies. The authors should try to untangle this two parts of the paper. For example, it was not clear how the experiments of Section 5.1 would be useful for an administrator.

I liked Figures 7 and 8 showing how content is served by different infrastructures. It would be cool to extend this into some nice viz software. Are these pictures consistent across traces? Would it make sense to visualize how they change with time?

How are the IP addresses on the x-axis of Figure 9 sorted?

In Tables 6 and 7 I was confused with the ground truth column. Does it denote the most popular service? What happens if the same port is used for different services in different systems?

What is the color-code in Figure 11? You might want to add a legend.

I didn’t get the point regarding ids 26-31 in Figure 11.

**Reviewer #3**

**Summary:** This paper explores the idea of labeling IP flows with domain-level information obtained from DNS lookups occurring in the traffic prior to the flows. The authors use five datasets recorded at POPs (i.e. with full access to clients’ traffic) in European and US providers, between 3h and 24h in length, recorded at various times in 2011.

The authors first describe the architecture of DN-Hunter, their DNS sniffer. It uses a two-level table hierarchy (for client and server IP addresses) and a maximum number of total entries stored (enforced via a ring buffer).

The authors propose three applications for DN-Hunter: spatial discovery of servers (where are the servers corresponding to a given domain), content discovery (what kinds of resources are hosted by which CDNs), and service tag extraction (identifying app-layer protocols via evidence in the hostname, e.g. "smtp" or "www"), and evaluate them using their five traces.

**Strengths:** The problem of identifying the organization that conceptually is responsible for serving a particular flow is interesting.

**Weaknesses:** The paper is missing any sort of correctness analysis of the flow association approach. The techniques could generally be implemented at the DNS level alone, without the need for association with transport-level flows, and the novelty is marginal as previous work has already studied DNS-provided name-to-IP/organization/CDN/location mappings.

**Comments to authors:** I had a very hard time with this paper. The basic idea in your paper – association of IP-level flows with destination labels provided by DNS – is potentially useful, but it’s (i) not clear why there’s a need for any kind of association when the DNS information alone suffices, and when you make the association, there’s room for error, which you do not analyze.

The main application I can see for DN-Hunter actually focuses on the client, not the server: for policy enforcement you may want to know *who* requested content from an organization. You mention this briefly, but it’s clearly not the focus of the paper.

**Reviewer #4**

**Summary:** The paper presents a system called DN-Hunter that leverages information provided by DNS traffic to provide fine-grained traffic visibility. The system is envisioned to be used by network operators for network management and enforcing policy in the presence of widespread use of end to end encryption such as TLS/SSL. The paper addresses an interesting and relevant problem.
Strengths:

- A key strength of this paper is that it is describing a system that’s already under a limited deployment and use in ISP networks.
- Solves a relevant and important problem.

Weaknesses: The paper assumes by default that use of DPI is necessary for ISPs to perform traffic engineering, enforcing policy etc., and sets out to solve the problem of giving more control to networks operators in the presence of encryption. The paper should also discuss alternative ways of performing traffic engineering without doing it on a per application basis?

Reviewer #5

Summary: The current Internet is built in such a way that content owners are decoupled from network service providers. This, despite the simple roles defined, could cause lots of complexity when one tries to understand who owns content or to improve content delivery. To handle such discrepancy, this paper proposes a method that can re-assemble the content delivery and ownership information based on DNS logs.

While DNS is a well-explored topic, this paper gives an interesting take on how to map out the association between the content ownership and content delivery. The idea is simple and the paper presents several interesting results that revisit some of the findings from the previous IMC papers. These results include identifying the delivery network structure of CDNs, association between popular content owners and CDNs, as well as the practices on the scaling of servers throughout the time of the day.

Strengths: The idea of mapping IP addresses with domain names is cool. The analysis in Section 5 is interesting. Nice set of related work.

Weaknesses: Section 3 is too domain specific. Could it be shortened?

Comments to authors: Overall, I would have appreciated the paper much more if it clearly stated its motivation -- why it is important to label various applications in the Internet? The authors do not explain why. Furthermore, I was a bit turned down by the fact that the authors argue for a one-size-fits-all system (Section 4 and 5). From the literature, I understand that each of the goals discussed in Section 4 is extremely challenging. However, this paper claims that the proposed system could achieve these goals trivially, which is not the case.

Another important discussion that I couldn’t find from the paper was about its overheads and drawbacks. Despite owning several merits, the system seems to require storing a large amount of information to map ID addresses to domain names. Is this feasible in larger-scale networks? What happens in the case of attack traffic that is directed to random destinations? Wouldn’t the system become very inefficient to store information for such invalid traffic? What about in case of P2P traffic? Could it significantly inflate the size of the mapping table that should be stored? I’m curious to know if the proposed solution is feasible in large networks.

I was also not sure how scalable and useful the system would be in terms of storing all information behind the second-level domain (e.g., smtp2.mail.google.com). The examples given in the paper are clearly good cases in that the second-level addresses are human-interpretable (e.g., smtp2). However, the server names and service names may not be human-interpretable for many real websites. In this case, what are the benefits of storing such information as opposed to storing simply port numbers?

All things said, this paper was an interesting thought experiment in seeing what would be possible if we have the full domain and server name mappings for all the IP addresses in the world.

Response from the Authors

First, we would like to sincerely thank the reviewers and our shepherd for all their valuable feedback. The majority of the comments have been included in the camera-ready, resulting in a significantly improved version of the paper.

All reviewers found the idea of annotating the traffic flows with DNS information to be compelling, and all of them agreed that DNHunter is a valuable tool to offer increased visibility to network traffic. Besides all minor changes, when reviewing our paper, we focused mostly on the summary of issues raised by the TPC.

Considering CDN content discovery and mapping, the geographical locations of servers, we agree that it is a topic could have been addressed without the introduced approach. Yet, the major plus of DNHunter is that it is completely passive, and very simple. It provides information to the ISP that naturally reflects its end-users habits along with the configuration CDNs adopt for serving traffic generated by ISP’s end-users. In other words, it reflects the "current" usage of the network with respect to the traffic and external events (like CDN policy changes).

With regards to the limitations of DNHunter, reviewers raised the issue of possible "collisions", i.e., the event in which the same clientIP is accessing two or more FQDNs hosted at the same serverIP. DNHunter returns the last observed FQDN, thus possibly returning incorrect labels. We assessed the collision probability in our datasets, and found that in practice collisions seldom happen. Other similar implementation issues (like presence of NAT and adoption of multiple browser by a end-user) have been discussed in the paper.

Lastly, we tried to evaluate how much information can be extracted from FQDNs, e.g., service names that may not be human-interpretable. However, we were not able to find a suitable methodology to thoroughly evaluate this. In general DNS has been designed to provide useful information to humans. It is up to the DNHunter user with the right domain knowledge to understand the exposed information. In the paper, we provide some examples of scenarios in which DNHunter is useful, and we believe that the applications of DNHunter, and of the Analyzer in particular, are not limited to the ones presented in our work. Novel applications can leverage the information exposed by the labeled flows database.