Is Your Caching Resolver Polluting the Internet?

Duane Wessels

CAIDA & The Measurement Factory, Inc.

wessels@measurement-factory.com

Abstract—

Previous research has shown that most of the DNS queries reaching the root of the hierarchy are bogus [1]. This behavior derives from two constraints on the system: (1) queries that cannot be satisfied locally percolate up to the root of the DNS; (2) some caching nameservers are behind packet filters or firewalls that allow outgoing queries but block incoming replies. These resolvers assume the network failure is temporary and retransmit their queries, often aggressively.

DNS pollution may not be causing any perceivable performance problems. The root servers seem well equipped to handle the load. Since DNS messages are small, the pollution does not contribute significantly to the total traffic generated by most organizations. Nonetheless, this paper provides a few reasons why network operators should take the time to investigate and fix these problems.

*Index Terms—***DNS caching root server**

I. INTRODUCTION

The Domain Name System (DNS) is the subject of critical attention and dramatic claims: how robust it is, how broken it is, how well it scales, and why it will fail.

The subject of this paper is *DNS pollution*, a term we use to refer to DNS queries (or responses) that should not occur on the wide-area Internet. We have been privileged to perform a number of studies of traffic reaching the "F" root DNS server. Without exception these studies confirm that many queries reaching the DNS root are bogus, i.e., constitute pollution of the global DNS.

Danzig *et al.* performed the first study of wide-area DNS misbehavior in 1992 [2]. They observed a variety of problems, such as recursion loops, uselessly repeated queries, and poor failure detection algorithms. Many of these problems remain with us today. Nine years later, Brownlee *et al.* did a similar study [3] where they found repeated queries, invalid TLDs, queries from and for RFC 1918 address space, attempts to update the root zone, and bogus A queries. Our 2003 study [1] found that 70% of F-root's queries were repeats and also characterized some of the busiest query sources. Our recent paper [4] studied the caching behavior of popular DNS nameserver implementations.

We are currently monitoring DNS traffic on two of the 20 F-root nodes.[1] The F-root, like many DNS root servers, uses IP anycast to make numerous, widely distributed nodes appear as a single service [5]. One of these anycast nodes is located in San Francisco, USA (named SFO2), and the other in Palo Alto, USA (named PAO1). On a normal weekday, each node receives approximately 2000 DNS queries per second. About 35% of those queries are immediately identifiable as pollution. That figure excludes repeated queries, which we do not address in this paper.

II. TYPES OF DNS POLLUTION

A. A-for-A Queries

An A-for-A query happens when a DNS client asks a question such as, "what is the IP address of 1.2.3.4?" Such a query is bogus because the query name is already an IP address. Software that originates such queries should be smart enough to not send them in the first place. Line 1 of Figure 1 shows an example of this type of query.

CAIDA's earlier paper [3] helped identify this behavior as a bug with the Microsoft Windows NT stub resolver. Usually, the resolver library, i.e., the code applications rely on to convert domain names to IP addresses, detects these bogus queries. For example, if an application passes an IPv4 address to the Unix *gethostbyname()* function, the library generates the answer internally without contacting any DNS servers.

However, some caching resolvers do not recognize this type of query as bogus. Instead, they will try to forward the query to an authoritative server, as if the last IP address octet were a valid top-level domain. The *dnscache* [6] nameserver automatically recognizes and answers these queries.

Since Microsoft found and fixed this bug with Windows NT Service Pack 2, we have observed a decrease in the percentage of queries that fall into this category. In the January 2001 study [3], bogus queries constituted about 14% of the total load measured at F-root. By October 2002, that number dropped to 7%, where it remains today

---

[1]http://f.root-servers.org/

```
1    23:15:45.225068 x.x.x.x > 192.5.5.241.53:    7416 A? 152.146.137.98. (32)
2    23:15:45.384598 x.x.x.x > 192.5.5.241.53:    231 PTR? 154.5.24.172.in-addr.arpa. (43)
3    23:16:18.266078 x.x.x.x > 192.5.5.241.53:    56503 [1au] A? kloun.com/index.htm. (48)
4    23:17:25.334112 x.x.x.x > 192.5.5.241.53:    18224 [1au] MX? smileys.html. (41)
5    23:17:46.072526 x.x.x.x > 192.5.5.241.53:    59041 [1au] A6? H.ROOT-SERVERS.NET. (47)
6    23:17:46.072551 x.x.x.x > 192.5.5.241.53:    34825 [1au] AAAA? H.ROOT-SERVERS.NET. (47)
7    23:17:46.072945 x.x.x.x > 192.5.5.241.53:    58725 [1au] A? H.ROOT-SERVERS.NET. (47)
8    23:17:46.072957 x.x.x.x > 192.5.5.241.53:    20650 [1au] A6? G.ROOT-SERVERS.NET. (47)
9    23:17:46.073957 x.x.x.x > 192.5.5.241.53:    47791 [1au] AAAA? G.ROOT-SERVERS.NET. (47)
10   23:17:46.074568 x.x.x.x > 192.5.5.241.53:    5951 [1au] A? G.ROOT-SERVERS.NET. (47)
```

Fig. 1. A few examples of DNS pollution as seen by *tcpdump*.

(April 2004). It is also interesting to note that this percentage is higher during weekends and lower during the work week, indicating that these queries are likely generated by automated agents rather than humans.

B. Queries and Updates for RFC 1918 Addresses

RFC 1918 [7] defines three IPv4 address blocks for use on private, internal networks. In theory, references to these addresses should never leak out into the public Internet. An organization that uses RFC 1918 addresses internally should also create authoritative *in-addr.arpa* zones for them on one or more nameservers. In this way, queries for RFC 1918 addresses remain within the organization.

The reality, of course, is that the root servers see millions of queries each day for RFC 1918 addresses. Line 2 of Figure 1 shows an example of this type of pollution.

Even if you do not use RFC 1918 addresses yourself, your caching nameserver may be a source of bogus queries. This behavior can occur if you have a service, such as an SMTP or HTTP server, that makes PTR queries for validation or logging. If your ISP does not block RFC 1918 addresses in its routers, you may receive packets from these source addresses. Your applications may attempt reverse DNS lookups even though that IP address is unroutable.

As of April 2004, bogus RFC1918 queries comprise about 1–3% of the total load at F-root. In fact, there are many more RFC 1918 queries out there that DNS root servers do not even see. Most of these queries go a server that has been delegated to be authoritative for the private address space *just to mitigate the pollution caused by these unnecessary and inappropriate queries*. In fact, there are actually 20 or so servers distributed around the Internet, using IP anycast to minimize the wide area bandwidth consumed by this pollution [8].

C. Queries for Invalid TLDs

We use "invalid TLD" to refer to a top-level domain that is not officially recognized by ICANN and IANA. In ad-

dition to the 243 Country Code TLDs, there are currently 14 Generic TLDs, and one infrastructure domain.[2]

Due to local configuration problems, queries containing many invalid TLDs frequently reach DNS root servers. Common invalid TLDs are: *localhost*, *local*, *corp*, *workgroup*, and *domain*. We also see TLDs that are more likely filename extensions, such as *htm*, *txt*, and *c*. See lines 3 and 4 in Figure 1.

Several circumstances could cause invalid domain names leak into the wide-area Internet:

1) Internet-connected hosts are configured with an incorrect domain name or none at all.
2) Human errors and software bugs may cause applications to attempt resolution of strings that are not really domain names.
3) Mobile devices, such as laptops, move from corporate intranets to the public Internet.
4) Products are distributed with bogus default values, which ignorant users do not change.

Queries with invalid TLDs are the most common type of DNS pollution. As of April 2004, 15% of queries reaching PAO1, and 20% reaching SFO2, fall into this category. Unfortunately, root servers cannot offload these queries to other servers, as they can with RFC 1918 *in-addr.arpa*, for example. They can only respond with a name error (NXDomain) message and hope that the client implements negative caching [9]. As far as we know, all of the well-known resolvers implement negative caching.

Invalid TLDs are particularly problematic because the DNS does not provide any way for a cache to realize that the TLD is invalid. For example, if a user makes a query for *aa.foobar*, the caching resolver contacts a root server and receives a name error. Subsequent queries for *aa.foobar* result in a negative cache hit. However, if the user then issues a query for *bb.foobar*, the caching resolver again contacts a root server. It does not realize that *foobar* is the problem, rather than *aa* or *bb*.
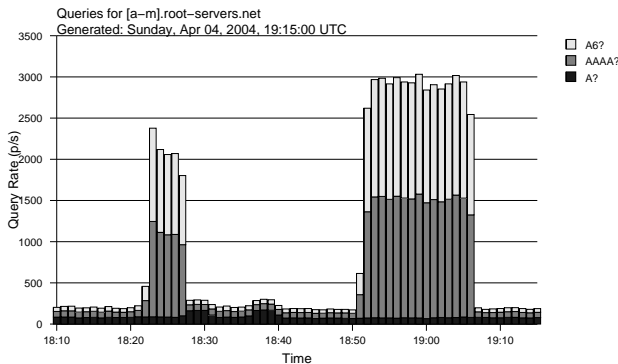
---

[2] http://www.iana.org/domain-names.htm

Fig. 2. Two spikes in IPv6 queries received by F-root for *[a-m].root-servers.net*.

D. Queries for [a-m].root-servers.net

All caching nameservers use a list of root server "hints" when they launch, which is how the nameserver knows where to send initial queries, until it learns about delegated nameservers through referrals. From time-to-time, the DNS cache refreshes the IP addresses of all root servers. See lines 5–10 of Figure 1. For a well-behaved caching resolver, this should happen when the cached records expire.

We currently observe a significant amount of queries for *[a-m].root-servers.net* at F-root: 10% of the total on PAO1 and 5% on SFO2. One reason is that this root server, and indeed most roots, are also authoritative for the *root-servers.net* domain. However, this rate of 100–200 queries/sec seems disturbingly high given that the A records for *[a-m].root-servers.net* have a 1000 hour TTL. We suspect that many of these queries are repeats from caches that, due to firewall or routing configuration issues, can send to, but not receive from, the root servers.

Even worse, we often see large query spikes (see Figure 2) at F-root for IPv6 addresses of *[a-m].root-servers.net*. During these spikes, which generally last from 5 to 20 minutes, these fruitless queries account for 25–50% of all queries received at the two F-root nodes. At this time, the *root-servers.net* zone does not contain any IPv6 address records, so these queries all result in negative answers. Investigation of the sources of these queries suggests that they are due to a bug in BIND versions 8.3.3 and 8.3.4.

E. IPv6 Address Queries

As mentioned in the previous section, IPv6 queries for *[a-m].root-servers.net* are a common source of pollution. In fact, about 1% of all the queries arriving at F-root are asking for the IPv6 address of one of the root servers. An-

other 6% of F-root's queries are asking for IPv6 addresses of other nameservers or hosts.

The DNS has two types of IPv6 address records: `AAAA` and `A6`. `AAAA` records are simple but long. `A6` records were designed to ease address portability issues at the expense of complicating the protocol. Since IPv6 is still rare compared to IPv4, most DNS zones do not have any `AAAA` or `A6` records for their nameservers.

BIND8 and BIND9 [10] are the only implementations known to issue `AAAA` and/or `A6` queries for nameserver "glue"[3], at least by default. These IPv6 queries happen even if you do not have any IPv6 addresses configured on your host.

Not only are these queries a completely avoidable burden on the roots, but identical IPv6 queries are also repeated far more frequently than necessary. Consider a query for the name *a.root-servers.net*, for example. A successful answer to an `A` query for that name is cached for five weeks. However, the negative answer to an `AAAA` query for that name is cached for a much shorter time. BIND limits negative caching to three hours by default.

III. DETECTING DNS POLLUTION

Detecting DNS pollution is challenging because often user applications continue to work even though some DNS queries yield useless or no answers. Thus, there is often little incentive to track down and fix global DNS pollution.

General purpose applications such as *tcpdump* and *etherreal* allow one to examine DNS queries on the wire in real-time, but it is an awkward way to debug DNS problems, especially for a busy server. Detecting pollution from such voluminous output can be excruciating.

As a recommended alternative, we offer[4] the *dnstop* program [12]. *dnstop* uses the Berkeley Packet Filter interface to capture DNS queries and displays statistics using curses [13]. *dnstop* maintains tables of query counts, and supports displaying results categorized by:

- Top-level domain
- Second-level domain
- Query type
- Opcode
- Source IP address
- Destination IP address

dnstop also has a number of filters that make it easier to detect certain problems. For example, the *A-for-A* filter

---

[3]Glue is an address record for a nameserver that tells a resolver how to contact a domain's nameserver before it knows anything else about that domain. See [11] for more information.

[4]*dnstop* is available under a BSD-style license.

```
Wed Apr 14 19:14:08 2004
9 new queries, 3045 total queries

TLD                    count      %
------------------- ---------- ------
local                    414   13.6
localhost                251    8.2
txt                       85    2.8
147                       31    1.0
1                         23    0.8
invalid                   22    0.7
null                      22    0.7
belkin                    21    0.7
workgroup                 20    0.7
212                       19    0.6
50                        18    0.6
c                         17    0.6
105                       17    0.6
iec                       17    0.6
10                        17    0.6
emails                    17    0.6
2                         16    0.5
8                         15    0.5
56                        15    0.5
3                         14    0.5
lan                       14    0.5
5                         13    0.4
0                         13    0.4
universe                  13    0.4
4                         12    0.4
domain                    11    0.4
```

Fig. 3.   Sample output of *dnstop -s -f unknown-tlds eth0* on F-root.

shows only type A queries that already look like an IP address. Thus, it becomes transparent to identify clients that send these bogus queries.

Similarly, the *unknown-tlds* filter shows only queries where the TLD is invalid. For example, Figure 3 shows the output of *dnstop* with the *unknown-tlds* filter on F-root for a few seconds.

The *dnstop* program also has a filter to detect PTR queries for RFC 1918 address space and another to detect dynamic DNS updates [14] to RFC 1918 space.

It is probably unnecessary to use *dnstop* to detect queries for *[a-m].root-servers.net*; one can easily use *tcpdump* and *grep* to list offending hosts. To use *dnstop* for the same task, simply look at the second-level domain counts and see if *root-servers.net* is higher than it should be: a correctly configured cache should not generate more than about 25 queries per hour.

IPv6 queries for nameserver addresses are not hard to detect either. If you use BIND, chances are good that your nameserver emits them. You can *grep* for AAAA and A6 in *tcpdump* output, or look at *dnstop*'s query type table.

IV.   ELIMINATING DNS POLLUTION

One of the first things to troubleshoot is whether your nameserver is actually able to receive replies from the Internet. We suspect that substantial DNS pollution is due to firewalls and packet filters that allow outgoing queries but block incoming replies. This type of misconfiguration causes tremendous DNS pollution because your nameserver continues to send queries indefinitely, not realizing that it cannot receive any answers. If you really do not want your caching nameserver to talk to other DNS servers, block the outgoing queries (e.g., UDP packets leaving your network destined for port 53), instead of only blocking responses.

A.   A-for-A Queries

The best way to eliminate these types of queries is to identify the source and get it fixed. If it is a Windows box, upgrade it to the latest service pack. If it is some other kind of device, look for a software upgrade or complain to the manufacturer.

Alternatively, you can stop bogus A-for-A queries by creating 256 authoritative, numbered zones on your nameserver. These zones can even be empty. As long as your nameserver is authoritative for all zones that end in a valid IP address octet, i.e., between 1 and 256, your nameserver will not forward such queries out to the Internet. With an empty, authoritative zone, the requester receives a "name error" response, indicating that the hostname does not exist.

Users of djbdns's *dnscache* will not have this problem because *dnscache* recognizes the bogus query and replies with an answer containing the same IP address. With one of the other implementations, you can configure the numbered zones (0–255) to delegate their subzones (also 0–255) to a *dnscache* instance.

B.   Updates and PTR Queries for RFC 1918 Addresses

Whether or not you actively use RFC 1918 addresses, minimizing DNS pollution requires that your your caching nameserver authoritative for the following zones:

- *10.in-addr.arpa*
- *16.172.in-addr.arpa* through *31.172.in-addr.arpa*
- *168.192.in-addr.arpa*

Again, as long as your local nameserver is authoritative for the zone, you remove the risk that these queries pollute the global Internet.

C. Queries for Invalid TLDs

There is no easy fix for stopping invalid TLD queries. The best approach we have is to use *dnstop* to determine which invalid TLDs are floating around your network. After identifying an invalid domain, you can use *tcpump* and *grep* to find out which machines are sending those queries. Then, either update the system configuration to use the correct TLD, or add authoritative zones for the invalid TLD as necessary.

An extraordinarily common "invalid TLD" is *localhost*. Normally, *localhost* is a hard-coded name for the loopback interface. However, it may be missing on some systems, or a system may have been configured to query the DNS before checking any local databases (such as the */etc/hosts* file).

To be safe, it is always a good idea to add a *localhost* zone to your caching nameserver. Be sure to enter an A record for *localhost* with the IP address 127.0.0.1.

D. Queries for [a-m].root-servers.net

Most likely, if you are polluting the Internet with queries for *[a-m].root-servers.net*, it means your cache is unable to receive DNS replies from root servers. Check your firewall or packet filter configuration and verify that DNS responses are allowed in if DNS requests are allowed out.

E. IPv6 Address Queries

As mentioned previously, BIND8 and BIND9 are the only implementations known to initiate IPv6 queries (for nameserver glue). The best way to eliminate these unnecessary queries is to upgrade your BIND installation to version 8.4.4 or later.

As of BIND-8.4.0, BIND8 no longer issues A6 queries for nameserver glue, although it still sends both AAAA and A queries, and of course it still forwards A6 queries on behalf of clients. Furthermore, beginning with version 8.4.3, it only queries for address types that are configured for transport. In other words, you can configure BIND to not initiate any IPv6 queries, either by not configuring any interfaces for IPv6, or by using a command line option.

Whereas BIND8 no longer initiates A6 queries, BIND9 does not initiate AAAA queries. Another interesting difference between the two is that BIND9 never fetches missing glue records, whereas it is configurable in BIND8. Note however that recent version of BIND9 initiate both A and A6 queries for *expired* glue records.

V. CONCLUDING REMARKS

Measurements at two DNS root server nodes indicate that 35% of the queries it receives fall into one of the five types of pollution described in this paper. Certainly some of these problems are worse than others and fixing them may require non-trivial effort on your part. Below are some reasons why you should care.

The fact that bogus DNS queries leave your network most likely means that your users' applications are not working properly, even if users are not aware of it. An obvious example is an HTTP server log file: if you use RFC 1918 addresses internally, an HTTP server may be making PTR queries for the access log. If these queries escape to the Internet and go unanswered, the HTTP server logs the IP address instead. Someone looking at the log file may never even realize that the server is configured to do reverse lookups.

Another possibility is that DNS pollution may be caused by software that should not be running on your network, such as a virus or some spyware. And in the context of software that should not be running on your network, consider that a sufficiently high-rate stream of DNS pollution is essentially a denial-of-service (DoS) attack. If you are careless, you may find that your caching nameserver is either intentionally or unintentionally participating in a DoS attack against another nameserver.

The vulnerability extends back to you. Bogus queries leaving your network may contain private information that an attacker can use against you; the dynamic DNS updates are an ideal example. Not only do those queries reveal to a listener what IP addresses you are using, they also often contain information such as machine types and usernames (e.g., *duanes-ibook*).

Bogus DNS queries consume network bandwidth but admittedly not much. The effect on your local network is also probably negligible. But if we can eliminate the 35% of pollution at a root server, such as F, which receives 2000 queries per second, we save 1.5 Mbit/s of Internet bandwidth and associated processing resources.

We are active advocates of improved software engineering for applications that use and implement the DNS protocol. But fixing a variety of pervasively deployed software modules is a daunting challenge, and in the interim we believe that enlightened configuration and deployment decisions of system and network administrators can considerably improve the integrity of the global DNS.

VI. ACKNOWLEDGEMENTS

Center (OARC) in conjunction with the Cooperative Association for Internet Data Analysis (CAIDA).

REFERENCES

[1] Duane Wessels and Marina Fomenkov, "Wow, That's a Lot of Packets," in *Proc. 2003 Passive and Active Measurements Workshop*, April 2003.

[2] P. B. Danzig, K. Obraczka, and A. Kumar, "An analysis of wide-area name server traffic," *ACM Comp. Commun. Review (SIG-COMM'92), Conference Proc.*, vol. 22, 4, pp. 281–292, 1992, `http://catarina.usc.edu/kobraczk/dns.ps.Z`.

[3] Evi Nemeth, k claffy, and Nevil Brownlee, "DNS Measurements at a Root Server," in *Proc. IEEE Globecom*, 2001.

[4] Duane Wessels, Marina Fomenkov, and Nevil Brownlee, "Measurements and Laboratory Simulations of the Upper DNS Hierarchy," in *Proc. 2004 Passive and Active Measurements Workshop*, April 2004.

[5] Joe Abley, "Hierarchical anycast for global service distribution," 2003.

[6] Daniel J. Bernstein, "djbdns," June 2003, `http://cr.yp.to/djbdns.html`.

[7] Y. Rekhter, B. Moskowitz, D. Karrenber, G. J. de Groot, and E. Lear, "RFC 1918: Address Allocation for Private Internets," February 1996.

[8] "The AS 112 Project," `http://www.as112.net`.

[9] Mark Andrews, "Negative Caching of DNS Queries (DNS NCACHE)," March 1998, Request For Comments 2038.

[10] Internet Software Consortium, "Berkeley Internet Name Domain (BIND) website," `http://www.isc.org/sw/bind/`.

[11] Paul Albitz and Cricket Liu, *DNS and BIND*, O'Reilly and Associates, 4th edition, April 2001.

[12] Duane Wessels, "dnstop," `http://dnstop.measurement-factory.com`.

[13] John Strang, *Programming With Curses*, O'Reilly and Associates, January 1986.

[14] P. Vixie, S. Thompson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)," April 1997, Request For Comments 2136.

[15] P. Mockapetris, "Domain names—concepts and facilities," November 1987, Internet Standard 0013 (RFCs 1034, 1035).

[16] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS Performance and the Effectiveness of Caching," 2001, `http://www.sds.lcs.mit.edu/papers/dns-imw2001.html`.