Wow, that is still a lot of packets

Roy Arends | ICANN | ICANN DNS Symposium

Introduction

- How we measure large datasets
- Simple classifications
- Aggregate counts
- Notable things
- Conclusion

How we measure large datasets

- L root: strong XZ compressed CBOR files
  - About 235 servers over 145 locations
  - Every 10 minutes (288 files per day, per server)
  - Total 67680 files per day
    - Process faster than 0.78 files/sec to avoid overflow

- B root: strong XZ compressed PCAP files
  - Large set, ”rotated” by size, not by time.
  - Size is about 2G uncompressed.
  - About 1200 files per day, on average
    - 20 seconds to decompress
    - 6 seconds to parse (dns_parse)
    - 4 seconds to grep/sed/awk
  - 1 day of traffic would take half a day to process

How we measure large datasets

- We looked at
  - Hadoop, Hbase, MongoDB, Cassandra, Turing, etc, etc

- While we're getting all the data in, we needed a temporal solution.

- UNIX file system.
- ASCII files as "index" to PCAP files.
- Use GREP to find strings/addresses
- Use AWK to count, find substrings, etc
- Use SORT to sort
- Gnu parallels to use multiple cores

- Simpler to use than most of the other solutions.
- Fast enough to do everyday analytics

How we measure large datasets

- Solution: build ASCII indexes from responses only
  - Optimised version of dns_parse
- Decompress and parse each each file to ASCII output once
  - In parallel (gnu parallel is your friend)
- Optimise further by:
  - Storing ascii output per RD_OPCODE_ANCOUNT_AA_RCODE file
  - One DNS response per line
  - Lowercase qname & Uppercase IPv6 avoids grep collision
  - Escape real dots & commas
    - Since commas are separators within a line
  - Ignore Query Count, Authority and Additional count
  - Keep all flags and some EDNS info
- Processing this takes about an hour per day of traffic (b-root)
  - Only needs to be done once.

How we measure large datasets

Parsing file: 20170210-121520-00590950.lax.pcap.xz leads to 15 new files

20170210-121520-00590950.lax.pcap.ND_NOTIFY_0_NA_REFUSED
20170210-121520-00590950.lax.pcap.ND_UPDATE_0_NA_REFUSED
20170210-121520-00590950.lax.pcap.ND_QUERY_0_AA_NOERROR
20170210-121520-00590950.lax.pcap.RD_QUERY_0_AA_NOERROR
20170210-121520-00590950.lax.pcap.ND_QUERY_0_AA_NXDOMAIN
20170210-121520-00590950.lax.pcap.RD_QUERY_0_AA_NXDOMAIN
20170210-121520-00590950.lax.pcap.ND_QUERY_0_NA_FORMERR
20170210-121520-00590950.lax.pcap.RD_QUERY_0_NA_FORMERR
20170210-121520-00590950.lax.pcap.ND_QUERY_0_NA_NOERROR
20170210-121520-00590950.lax.pcap.RD_QUERY_0_NA_NOERROR
20170210-121520-00590950.lax.pcap.ND_QUERY_0_NA_REFUSED
20170210-121520-00590950.lax.pcap.RD_QUERY_0_NA_REFUSED
20170210-121520-00590950.lax.pcap.ND_QUERY_1_AA_NOERROR
20170210-121520-00590950.lax.pcap.RD_QUERY_1_AA_NOERROR
20170210-121520-00590950.lax.pcap.ND_UPDATE_0_NA_NOTAUTH

Which contains lines like:
216.109.3.167,e6987.a.akamaiedge.net.,1,U-D
(Source address, qname, qtype, bits)

Simple classification

- For all responses, classify per:
  - RD bit
  - Opcode
  - Are there Answers in Answer section
  - AA bit Set or Clear
  - RCODE

Simple classification

AA	RCODE	Ans	Description
Set	NOERROR	0	NODATA
Set	NOERROR	1+	Auth Answer
Set	NXDOMAIN	0	NXDOMAIN
Set	NXDOMAIN	1+	NXCNAME
Clear	NOERROR	0	Delegation
Clear	NOERROR	1+	Cached ans.
Clear	NXDOMAIN	0	Cached NXD.
Clear	NXDOMAIN	1+	Cached NXCNAME

Crickets chirping

Wow, That's a Lot of Packets

Duane Wessels, Marina Fomenkov

Abstract— **Organizations operating Root DNS servers report loads exceeding 100 million queries per day. Given the design goals of the DNS, and what we know about today's Internet, this number is about two orders of magnitude more than we would expect.**

With the assistance of one root server operator, we took a 24-hour trace of queries arriving at one of the thirteen root servers. In this paper we analyze these data and use a simple model of the DNS to classify each query into one of nine categories. We find that, by far, most of the queries are repeats and that only a small percentage are legitimate.

We also characterize a few of the "root server abusers," that is, clients sending a particularly large number of queries to the root server. We believe that much of the root server abuse occurs because the querying agents never receive the replies, due either to packet filters, or to routing issues.

Keywords— **DNS root server**

I. Background: *DNS 101*

The Domain Name System (DNS) is a fundamental component of the modern Internet [1], providing a critical link between human users and Internet routing infrastructure by mapping host names to IP addresses. The DNS utilizes a hierarchical name space divided into zones, or domains. This hierarchy is manifested in the widespread "dots" structure. For example, `com` is the parent zone for `example.com`, `microsoft.com`, `cnn.com`, and approximately 20 million other zones.

Each zone has one or more authoritative name servers. These are dedicated servers, whose job is to answer queries for names within their zone(s). For example, UCSD has three authoritative name servers. An application that needs to know the IP address for `www.ucsd.edu` can send a DNS query to one of those servers, which then returns an authoritative answer. If the application does not know where to send a query it asks the servers in the parent zone. In the example above, not knowing anything about `ucsd.edu`, the application should send a query to the authoritative server for the `edu` zone. If the application does not know about the `edu` zone, it queries the "root zone." This process is called *recursive iteration*.

The DNS root zone is served by 13 name servers (not to be confused with the 13 generic top-level domain servers) distributed across the globe. Thirteen is the maximum number of root servers possible in the current DNS architecture because that is the most that can fit inside a 512-byte UDP reply packet. Ten root servers are located in the U.S., two are in Europe, and one is in Asia.[1] The root zone and the root name servers are vital because they are the starting points for locating anything in the DNS. Without them, the DNS and hence almost every application we use (the Web, ssh, email) would be rendered unusable.

DNS clients, or resolvers, that query name servers, come in one of two flavors: stub and recursive. Stub resolvers, typically found in user applications, such as web browsers, ssh clients, and mail transfer agents, are rather primitive and mostly rely on smarter recursive resolvers that understand name server referrals. Recursive resolvers are usually implemented in specialized DNS applications such as the Berkeley Internet Domain Name (BIND) [2] server and Microsoft's DNS server. Most organizations operate local recursive name servers.

Recursive name servers cache name server responses, including referrals. Caching conserves network resources because intermediate servers do not need to query the root name servers for every request. For example, the name server learns that `a.gtld-servers.net` and others are authoritative for the `com` zone and sets the time-to-live (TTL) for this information. Typical TTLs for top level domains are on the order of 1–2 days.

In theory, a caching recursive name server only needs to query the root name servers for an unknown top level domain or when a TTL expires. However, a number of studies have shown that the root name servers receive many more queries than they should. In this paper we thoroughly investigate and characterize root name server traf-

The Measurement Factory, Inc., Boulder, Colorado, E-mail: `wessels@measurement-factory.com`.

CAIDA, San Diego Supercomputer Center, University of California, San Diego. E-mail: `marina@caida.org`.

Support for this work is provided by WIDE and DARPA NMS N66001-01-1-8909.

[1] In fact many of the root name servers are actually multiple hosts behind network load balancers. Some of them even occupy a few physical locations, employing IPv4 anycast to operate under a single IP address.

Wow, That's a Lot of Packets

Duane Wessels, Marina Fomenkov

Type	Count	Percent
Unused Query Class	36,313	.024
A for A	10,739,857	7.03
Unknown TLD	19,165,840	12.5
Nonprintable in query	2,962,471	1.94
RFC1918 PTR	2,452,806	1.61
Identical Query	38,838,688	25.4
Repeated Query	68,610,091	44.9
Referral Not Cached	6,653,690	4.36
Legitimate	**3,284,569**	**2.15**

TABLE II

QUERY CLASSIFICATION RESULTS (24-HOUR PERIOD ON 4 OCTOBER 2002 AT THE F-ROOT DNS SERVER).

more queries than they should. In this paper we thoroughly investigate and characterize root name server traf-

[1] In fact many of the root name servers are actually multiple hosts behind network load balancers. Some of them even occupy a few physical locations, employing IPv4 anycast to operate under a single IP address.

Simple ad-hoc traffic measurement

- B-ROOT RSSAC002 stats for 10th Feb 2017
  - total: 3.0 G responses (3.015.731.920)
    - 83.5% UDP-v4
    - 14.0% UDP-v6
    - 2.2% TCP-v4
    - 0.24 % TCP-v6
  - 99% UDP queries saw a response.
  - We're going to ignore TCP for this effort (not statistically significant)
- Observed in actual traffic: 2941687705 UDP responses
  - Error is 0.00037 (insignificant)

Simple classification

AA	RCODE	Ans	Description
Set	NOERROR	0	NODATA
Set	NOERROR	1+	Auth Answer
Set	NXDOMAIN	0	NXDOMAIN
Set	NXDOMAIN	1+	NXCNAME
Clear	NOERROR	0	Delegation
Clear	NOERROR	1+	Cached ans.
Clear	NXDOMAIN	0	Cached NXD.
Clear	NXDOMAIN	1+	Cached NXCNAME

Simple classification

AA	RCODE	Ans	Description
Set	NOERROR	0	NODATA
Set	NOERROR	1+	Auth Answer
Set	NXDOMAIN	0	NXDOMAIN
Clear	NOERROR	0	Delegation

Simple classification

AA	RCODE	Ans	Description
Set	NOERROR	0	NODATA
Set	NOERROR	1+	Auth Answer
Set	NXDOMAIN	0	NXDOMAIN
Clear	NOERROR	0	Delegation
---	---	---	---
	FORMERR		Huh?
	NOTIMP		Can’t do it
	REFUSED		Go Away
	NotAuth		Won’t do it

Simple classification

AA	RCODE	Ans	Description
Set	NOERROR	0	NODATA
Set	NOERROR	1+	Auth Answer
Set	NXDOMAIN	0	NXDOMAIN
Clear	NOERROR	0	Delegation
---	---	---	---
	FORMERR		Huh?
	NOTIMP		Can’t do it
	REFUSED		Go Away
	NotAuth		Won’t do it

Simple classification

AA	RCODE	Ans	Description
Set	NOERROR	0	NODATA
Set	NOERROR	1+	Auth Ans.
Clear	NOERROR	0	Delegation

Simple classification

AA	RCODE	Ans	Description		
Set	NOERROR	0	NODATA	25.3M	.84%
Set	NOERROR	1+	Auth Ans.	99.8M	3.31%
Clear	NOERROR	0	Delegation	1018M	34 %

34% of all queries result in delegations

What about caching?

- 34% of all queries result in delegations
- All delegation point NS records have a 2 day TTL
- Proper caching: at most 1 query for per TLD per source IP
- Of the 1018336727 delegation responses (34% of all responses):

What about caching?

- 26% of all queries result in delegations
- All delegation point NS records have a 2 day TTL
- Proper caching: at most 1 query for per TLD per source IP
- Of the 1018336727 delegation responses (34% of all responses):
  - 997673948 are duplicates
  - 98 %
- Conclusion: 20662779 “1st” responses, the rest could have been cached

What does bogus look like

- 2.7 % of all Authoritative NODATA responses are for type A6
- Large amount of proper delegations are for RFC1918 reverse address space (and other lame addresses)
- Reflection and Amplification attacks
- Spam traffic (loads of MX queries)
- DGA related traffic

Conclusion

- The root server system is One Big Filter for loads of bad queries
  - Only 34% result in a delegation
- The bulk of the 62% should never have been send in the first place
- The bulk of the 34% should have been properly cached.
- The 34% of delegations still contains loads of DGA, RFC1918 address space, spam traffic.
- It is nearly impossible to “fix” any of this “at the root”
  - (if you don’t respond, things get worse)
- Some recommendations for resolvers:
  - Properly cache, local root copy, ACLs, domain block lists

Questions?