DNSwitness: recent developments and the new passive monitor

Stéphane Bortzmeyer
AFNIC
bortzmeyer@nic.fr

RIPE 59 - Lisbon - October 2009

Where are we in the talk?

1. Reminder about DNSwitness
2. Measurements based on passive observations
3. Preliminary Results
4. Future work
5. Measurements based on active queries

AFNIC

What is AFNIC

AFNIC is the registry for the TLD "`.fr`" (France) .

54 employees, 1.5 million domain names and a R&D department.

Motivation

A DNS registry has a lot of information it does not use.

Our marketing team or the technical team ask for all sorts of things ("How many of our domains are used for e-mail only?") for which we **may** have the answer.

More specific motivation

Getting information about the deployment of new techniques like IPv6

We focus on things that we can obtain from the DNS because we are a domain name registry.

More specific motivation

Getting information about the deployment of new techniques like IPv6

We focus on things that we can obtain from the DNS because we are a domain name registry.

Possible surveys: IPv6, SPF, DNSSEC, EDNS0, Zonecheck... Let's build a multi-purpose platform for that!

AFNIC

Other aims

1. **Versatile**, able to do many different surveys (most known tools deal only with one survey),
2. Works unattended (from cron, for instance), for periodic runs,
3. Stores raw results, not just aggregates, for long-term analysis,
4. Designed to be distributable,
5. Designed to be usable by small and medium actors ("**send the program to the users, not the data to a centralized analysis fabric**").

What we can learn from the DNS (and beyond)

- What we send **out**: active DNS queries sent to domain name servers. **Active** measurements. (Presented at the RIPE 57 meeting in Dubai.)

What we can learn from the DNS (and beyond)

- What we send **out**: active DNS queries sent to domain name servers. **Active** measurements. (Presented at the RIPE 57 meeting in Dubai.)
- What comes **in**: DNS queries received by authoritative name servers, passively monitored ("Who knocks at the door and what are they asking for?"). **Passive** measurements.

What we can learn from the DNS (and beyond)

- What we send **out**: active DNS queries sent to domain name servers. **Active** measurements. (Presented at the RIPE 57 meeting in Dubai.)
- What comes **in**: DNS queries received by authoritative name servers, passively monitored ("Who knocks at the door and what are they asking for?"). **Passive** measurements.

We work on both, study the long-term evolution and publish results.

Where are we in the talk?

AFNIC

Passive observation of queries

It works by passive monitoring of the "`fr`" name servers. We are talking about long-term monitoring, not just the quick glance that DSC offers.

The idea is to address the needs of the R&D or of the marketing, not just the needs of the NOC.

Passive observation of queries

It works by passive monitoring of the "`fr`" name servers. We are talking about long-term monitoring, not just the quick glance that DSC offers.

The idea is to address the needs of the R&D or of the marketing, not just the needs of the NOC.

It works mostly by Ethernet port mirroring.

Expected uses of the passive measurements

It allows us to survey things like:

Expected uses of the passive measurements

It allows us to survey things like:

- Percentage of servers without SPR (Source Port Randomisation, see "`.at`" publications).

Expected uses of the passive measurements

It allows us to survey things like:

- Percentage of servers without SPR (Source Port Randomisation, see "`.at`" publications).
- Percentage of queries done over IPv6 transport (unlike DSC, we will be able to study long-term trends).

Expected uses of the passive measurements

It allows us to survey things like:

- Percentage of servers without SPR (Source Port Randomisation, see "`.at`" publications).
- Percentage of queries done over IPv6 transport (unlike DSC, we will be able to study long-term trends).
- Percentage of queries with EDNS0 or DO.

Expected uses of the passive measurements

It allows us to survey things like:

- Percentage of servers without SPR (Source Port Randomisation, see "`.at`" publications).
- Percentage of queries done over IPv6 transport (unlike DSC, we will be able to study long-term trends).
- Percentage of queries with EDNS0 or DO.
- Top N domains for which there is a NXDOMAIN reply.

Expected uses of the passive measurements

It allows us to survey things like:

- Percentage of servers without SPR (Source Port Randomisation, see "`.at`" publications).
- Percentage of queries done over IPv6 transport (unlike DSC, we will be able to study long-term trends).
- Percentage of queries with EDNS0 or DO.
- Top N domains for which there is a NXDOMAIN reply.
- But the list is open. . .

Sampling

Packet trace files can grow very large

Dozens of gigabytes are very common. And, to process such humongous data, you need a lot of RAM!

Sampling

Packet trace files can grow very large

Dozens of gigabytes are very common. And, to process such humongous data, you need a lot of RAM!

Sampling is often the only solution, unless you have a **lot** of disk and machine power

A framework for sampling

- RFC 5474, A Framework for Packet Selection and Reporting (the general framework and the concepts)
- RFC 5475, Sampling and Filtering Techniques for IP Packet Selection (actual techniques)
- RFC 5476, Packet Sampling (PSAMP) Protocol Specifications (not used by DNSmezzo)

Among the sampling techniques listed by RFC 5475: systematic count-based, systematic time-based, random (with various distributions), ...

Limits of sampling

Sampling makes **sampling errors**. If a phenomenon is rare, sampling can make it disappear completely. . . or promote it if it falls in the sampling window!

Limits of sampling

Sampling makes **sampling errors**. If a phenomenon is rare, sampling can make it disappear completely. . . or promote it if it falls in the sampling window!

Do not forget to plot the error bars.

Limits of sampling

Sampling is not suitable for many security studies: the attack can be just between the sampled packets. Example: BIND dynamic update DoS attack of 2009 where one packet was enough. References: section 9 of RFC 5475 and S. Goldberg, J. Rexford, "Security Vulnerabilities and Solutions for Packet Sampling", IEEE Sarnoff Symposium, Princeton, NJ, May 2007 `http://www.cs.princeton.edu/~jrex/papers/psamp-security07.pdf`.

Implementation

DNSmezzo has three parts:

- The capture program, which does the sampling (AFNIC uses pcapdump, from ISC). Anything which produces pcap works (tcpdump, dnscap, etc).
- The **dissector** which parses the DNS packets and stores them in a rDBMS. Written in C at AFNIC.
- The reporting programs, typically a combination of SQL, Python and Gnuplot.

Hence, we completely separate trace files parsing from data analysis.

Capturing packets

We all know capture tools like tcpdump and the pcap format it popularized `http://www.tcpdump.org/`.

Writing your own capture tool is easy but there is one already made, which suited our requirments: pcapdump, from the pcaputils package `http://packages.debian.org/pcaputils`.

pcapdump can do the sampling, can rotate files and name them properly, etc.

Dissecting pcap files

A very common task, with a lot of code available on the Internet (I recommend Wireshark).

Dissecting pcap files

A very common task, with a lot of code available on the Internet (I recommend Wireshark).

But a dangerous task, especially in a language like C

Every possible error can be found in the wild. Either by malice or by bug.

Dissecting pcap files

But a dangerous task, especially in a language like C

Every possible error can be found in the wild. Either by malice or by bug.

If you love buffer overflows, dissecting pcap is for you. (See the list of security alerts for Wireshark.)

Examples: name compression pointers going outside of the packet, section counts > 0 while the corresponding section is empty, etc.

Dissecting pcap files

But a dangerous task, especially in a language like C

Every possible error can be found in the wild. Either by malice or by bug.

If you love buffer overflows, dissecting pcap is for you. (See the list of security alerts for Wireshark.)

Examples: name compression pointers going outside of the packet, section counts > 0 while the corresponding section is empty, etc.

Tests with Python were not good, speed-wise, so we moved to C. For DNS parsing, we could have used ldns or a similar lib. For further study.

Storing in the rDBMS

The relational DBMS gives us versatility and simplicity (everyone knows SQL): this is great for data analysis.

A few principles:

- As much as possible, store the original information. You never know what you will need. Example: we keep the original case of the QNAME, we do not normalize it.
- As far as possible, keep the history, store the packets, not aggregates. You never know what you will want to study in the future.

A few implementation choices

- Use integers for fields like the QTYPE or QCLASS: loses typing, less convenient but allows for unexpected QTYPE,
- Use a special type for domain names, allowing easy extract of things like the TLD (not yet finalized),
- Use a proper type for IP addresses, **not** text, to allow things like grouping per prefix,
- PostgreSQL (with its rich typing system).

Science-fiction

Recode everything on a shared-nothing architecture in the cloud

With MapReduce on Hadoop :-)

Querying DNS with SQL

All the data is stored in a rDBMS. Analysis is then performed with SQL, without interfering with pcap parsing issues.

```
-- Top non-existing requested domains
SELECT DISTINCT domain, count(domain) AS num FROM DNS_packets
             WHERE NOT query AND rcode = 3    -- NXDOMAIN
             GROUP BY domain
             ORDER BY num DESC;

-- Non-ASCII requests. QNAMEs are stored as UTF-8
SELECT src_address, qname FROM DNS_packets
         WHERE octet_length(qname) > length(qname);
```

SQL requests, the sequel

```
-- IPv6 requests
SELECT count(id) FROM DNS_packets WHERE query AND
                              family(src_address) = 6;

-- Most common QTYPE.
-- RR types are stored in an auxiliary table
SELECT (CASE WHEN type IS NULL THEN qtype::TEXT ELSE type END),
       meaning,
       count(results.id) AS requests FROM
             (SELECT id, qtype FROM dns_packets
                  WHERE query) AS Results
          LEFT OUTER JOIN DNS_types ON qtype = value
              GROUP BY qtype, type, meaning
      ORDER BY requests desc;
```

Querying DNS with SQL

The SQL way is often criticized for performance issues. A few methods to make things more manageable:

- Sampling, of course
- Liberal use of indexes (spend space to save time)
- PostgreSQL's excellent EXPLAIN command
- Add RAM

Performance measure

Test with 85 Mpackets (returning 192 tuples)

```
% echoping -n 3 -m postgresql localhost -c dbname=dnsmezzo2 \
       "SELECT * FROM DNS_packets WHERE qname='example.fr'"
Elapsed time: 1.269121 seconds
Elapsed time: 0.002879 seconds
Elapsed time: 0.002657 seconds
```

(Once it is in the cache, it works fast.)

Size of data

On a name server with 1,300 queries/s, with a (very aggressive) sampling of 1 % and a maximum capture size of 512 bytes, the typical daily pcap file is 250 megabytes.

```
% capinfos mezzo-a.nic.fr-SAMPLING-100.2009-08-31.22:00.pcap
...
Number of packets:   2114633
File size:           287498993 bytes
Capture duration:    86400 seconds
Start time:          Tue Sep  1 00:00:02 2009
End time:            Wed Sep  2 00:00:01 2009
Data byte rate:      2936.03 bytes/sec
Data bit rate:       23488.27 bits/sec
Average packet size: 119.96 bytes
Average packet rate: 24.47 packets/sec
```

Size matters

Storing it to the database expands it by a factor 5 (half of the expansion coming from the indices).

```
dnsmezzo2=> SELECT sum(storedpackets) FROM pcap_files;
   sum
----------
 71771702

dnsmezzo2=> SELECT pg_size_pretty(sum(filesize)) FROM pcap_files;
 pg_size_pretty
----------------
 9404 MB

dnsmezzo2=> SELECT pg_size_pretty(
                   pg_total_relation_size('DNS_packets'));
 pg_size_pretty
----------------
 55 GB
```

Where are we in the talk?

AFNIC

Actual results

No long-term studies yet, the program is too recent.

Actual results

No long-term studies yet, the program is too recent.

Still several biases (only one name server, caching at ISP, . . .).

AFNIC setup

- Sampling at 1 %, random,

AFNIC setup

- Sampling at 1 %, random,
- Data collection during 24 hours (as with DITL),

AFNIC setup

- Sampling at 1 %, random,
- Data collection during 24 hours (as with DITL),
- Just one name server,

AFNIC setup

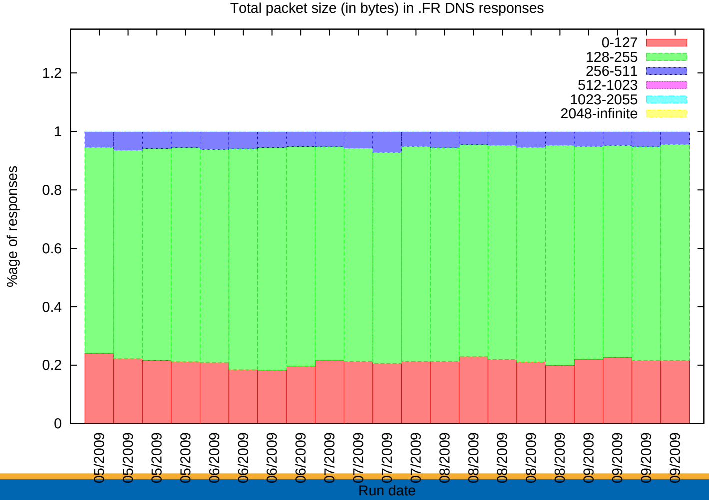
- Sampling at 1 %, random,
- Data collection during 24 hours (as with DITL),
- Just one name server,
- Capture with pcapdump.

IPv6

- 0,6 % of requests over IPv6 (no change in 2009)
- Other statistics do not seem to depend on the address family (for instance, non-SPR clients are as common with v6 and v4)

Size of the responses

Response size can be an issue for IP fragmentation, for instance.



AFNIC

Most queried domains

A important question for the management: what are the most popular domains?

Important, but there are many traps!

Most queried domains

A important question for the management: what are the most popular domains?

Important, but there are many traps!

- Caching at the ISP seriously change the pattern
- Domains with low TTL are queried more often
- "Infrastructure" domains (used on the right-hand side of the NS records) are the most popular. If they break, they take many domains with them.

Most queried domains

A important question for the management: what are the most popular domains?

Important, but there are many traps!

- Caching at the ISP seriously change the pattern
- Domains with low TTL are queried more often
- "Infrastructure" domains (used on the right-hand side of the NS records) are the most popular. If they break, they take many domains with them.

"`nic.fr`" is by far the most often queried.

Most queried domains

A important question for the management: what are the most popular domains?

Important, but there are many traps!

- Caching at the ISP seriously change the pattern
- Domains with low TTL are queried more often
- “Infrastructure” domains (used on the right-hand side of the NS records) are the most popular. If they break, they take many domains with them.

“`nic.fr`” is by far the most often queried.

The “Top N”study may be published separately. Wait for the paper :-)

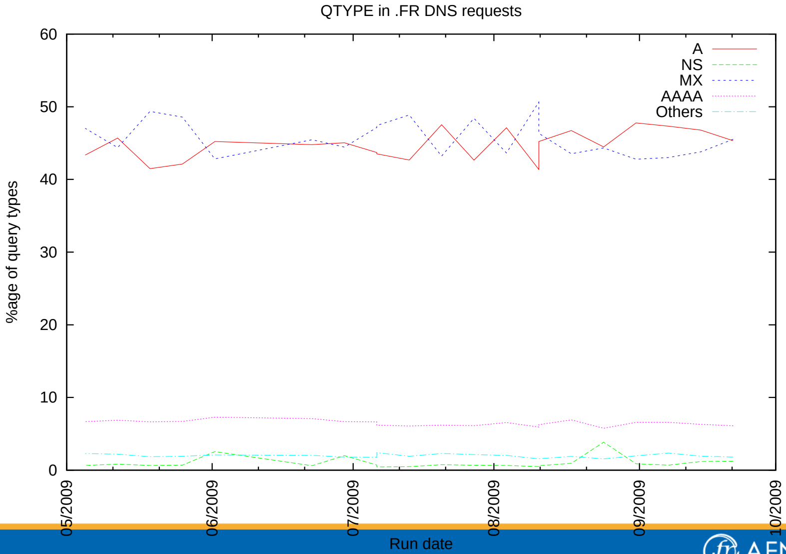
Kaminsky, one year after

Still 18 % of clients without SPR (less than one port per two requests)

They are not only small resolvers, they make 15 % of the requests.

Methodology: we eliminate small clients (not enough requests) and recursive requests (dig. . .).

Percentage of requests per query type



Comparison with other systems

- ISC SIE `https://sie.isc.org/`
- IIS.se dns2db `http://opensource.iis.se/trac/dns2db`
- DSC `http://dns.measurement-factory.com/tools/dsc/`

DNSmezzo and friends

- SIE is optimized for **huge** volumes of data, DNSmezzo for versatility.
- DNSmezzo typically works with sampled data (so it requires less hardware resources but it cannot do security analysis, only stats)
- DNSmezzo's code is published, we encourage the "perform your analysis yourself" which can be useful for a TLD.
- DSC is more targeted to real-time monitoring, its quantitative precision decreases with time (also, at AFNIC, it is not installed with QNAME parsing).
- DNSmezzo is very close, in its principles, to dns2db.

Distribution

`http://www.dnswitness.net/`

Distributed under the free software licence GPL.

Where are we in the talk?

AFNIC

Future work on DNSmezzo

- Parse some information that is currently ignored (such as EDNS option codes, for EDNS0-ping, for instance)
- Write more reports with the information we have
- Deploy more probes (warning: consolidation of data from different name servers is not obvious)

Where are we in the talk?

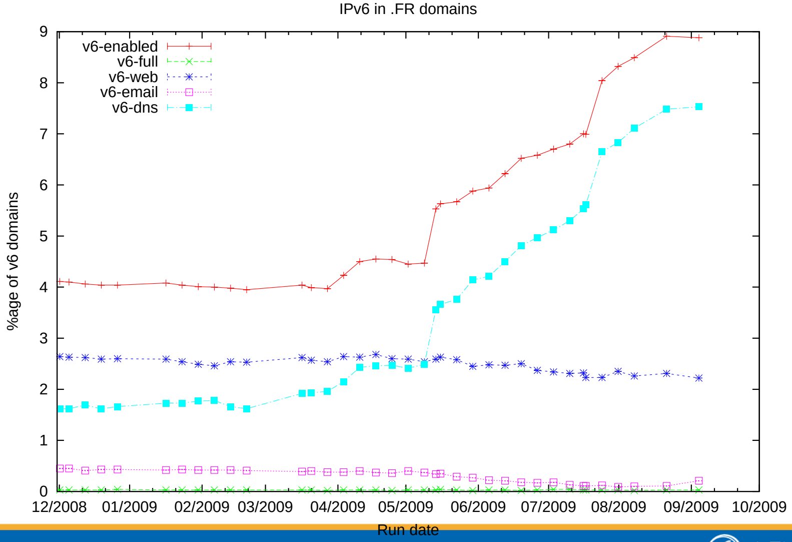
AFNIC

Active queries

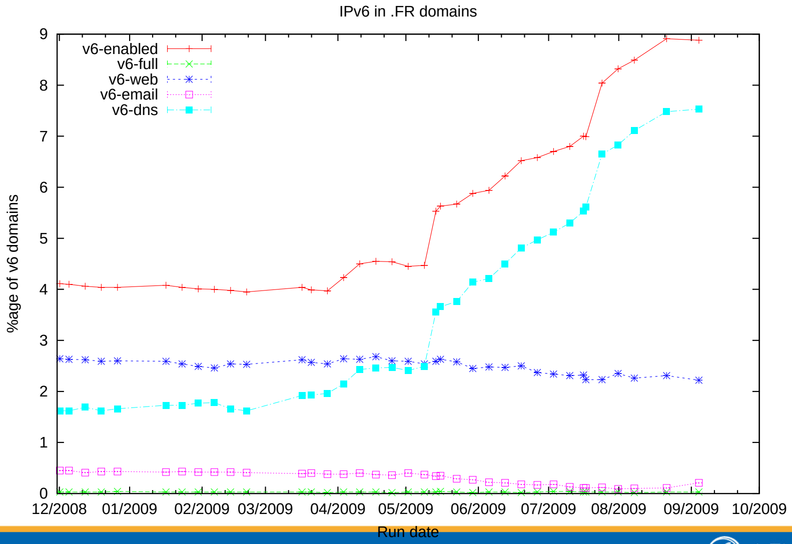
Active queries

This is the realm of our **DNSdelve** program.

Active queries



Active queries



AFNIC

Future work on the rest of the project

- Gather more users. Yes, you :-)

Future work on the rest of the project

- Gather more users. Yes, you :-)
- Come back in one year with trends, new applications, etc.