DNSwitness, a versatile platform for DNS metrics

Stéphane Bortzmeyer
AFNIC
bortzmeyer@nic.fr

G6 - 18 march 2009

Where are we in the talk?

What is AFNIC

AFNIC is the registry for the TLD “.fr” (France) .

51 employees, 1.2 million domain names and a quite recent R&D department.

Motivation

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

Our marketing team or the technical team are asking for all sort 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!

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.

What we can learn from the DNS (and beyond)

- What we send **out**: active DNS queries sent to domain name servers. (**The harpoon.**)

What we can learn from the DNS (and beyond)

- What we send **out**: active DNS queries sent to domain name servers. (**The harpoon.**)
- What comes **in**: DNS queries received by authoritative name servers, passively monitored ("Who knocks at the door and what are they asking for?"). (**The net**.)

What we can learn from the DNS (and beyond)

- What we send **out**: active DNS queries sent to domain name servers. (**The harpoon.**)
- What comes **in**: DNS queries received by authoritative name servers, passively monitored ("Who knocks at the door and what are they asking for?"). (**The net**.)

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

Where are we in the talk?

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.

We use port mirroring.

Expected uses of the passive measurements

It will allow us to survey things like:

Expected uses of the passive measurements

It will allow us to survey things like:

- Percentage of servers without SPR (Source Port Randomisation).

Expected uses of the passive measurements

It will allow us to survey things like:

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

Expected uses of the passive measurements

It will allow us to survey things like:

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

Expected uses of the passive measurements

It will allow us to survey things like:

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

Expected uses of the passive measurements

It will allow us to survey things like:

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

Related work

- See “.at” publications or “.com” or “.jp” similmar work on Source Port Randomization

How it works

It mostly works by watching DNS traffic with port mirroring on a name server.

How it works

It mostly works by watching DNS traffic with port mirroring on a name server.

Implementation: a C program reads the pcap file, dissects it (watch the malformed packets!), and loads the data in a DBMS (like IIS.se's DNS2DB). Analysis is then performed with SQL only.

Examples of SQL queries

- Recherche des NXDOMAIN : `SELECT DISTINCT qname AS domain,count(qname) AS num FROM DNS_packets WHERE NOT query AND rcode=3 GROUP BY qname ORDER BY count(qname) DESC;`
-
-

Examples of SQL queries

- Recherche des NXDOMAIN : `SELECT DISTINCT qname AS domain,count(qname) AS num FROM DNS_packets WHERE NOT query AND rcode=3 GROUP BY qname ORDER BY count(qname) DESC;`
- Recherche de clients IPv6 : `SELECT count(id) FROM DNS_Packets WHERE family(src_address) = 6;`
-

Examples of SQL queries

-
- Recherche de clients IPv6 : `SELECT count(id) FROM DNS_Packets WHERE family(src_address) = 6;`
- Recherche de clients Conficker : `SELECT count(id) FROM dns_packets WHERE NOT query AND rcode=3 AND qname IN (select name from conficker.candc_domains);`

Examples of SQL queries

-
-
- Recherche de clients Conficker : `SELECT count(id) FROM dns_packets WHERE NOT query AND rcode=3 AND qname IN (select name from conficker.candc_domains);`

Enlarge your data

pcap files are humongous and you cannot store everything.

(Think about our poor colleagues of "`.com`".)

Some figures:

- A run of 24 h on `a.nic.fr` is 80 millions packets and 20 gigabytes.
- Dissecting it and loading it in the DBMS takes several hours

Enlarge your data

Some figures:

- A run of 24 h on `a.nic.fr` is 80 millions packets and 20 gigabytes.
- Dissecting it and loading it in the DBMS takes several hours

Thanks to PostgreSQL authors, the DBMS handled it easily.

First results in march 2009

- 0.8 % of the requests arrive on IPv6
- 12 % of address requests are for IPv6 (this reflects the policies of the resolvers)
- We never see stacked IPv6 headers (remember our clients are name servers, not mobile PCs)
- SPR: 10 % of resolvers are “bad” (but they account for 17 % of requests)
- a lot of funny stuff happends (IPv6 resolvers do not seem much better, some resolvers change the port but among 20 or 30 values, . . .)

Future work

- Long-term surveys (this will require more disk space!)

Where are we in the talk?

Related work

- Patrick Maigron's measurements on IPv6 penetration `http://www-public.it-sudparis.eu/~maigron/`
- JPRS, the ".jp" registry makes for a long time detailed measures on IPv6 use (not yet published, see `http://v6metric.inetcore.com/en/index.html`)
- "`iis.se`" "engine", part of their dnscheck tools, allows scanning the entire zone to test every subdomain is properly configured `http://opensource.iis.se/trac/dnscheck/wiki/Engine`
- And many others

How it works

It mostly works by asking the DNS. It loads a list of delegated zones and queries them for various records.

How it works

It mostly works by asking the DNS. It loads a list of delegated zones and queries them for various records.

But it can also perform other queries: HTTP and SMTP tests, running Zonecheck...

The first algorithm

Crude version of this program (everyone at a TLD registry wrote such a script at least once). Here, to test SPF records:

```
for domain in $(cat $DOMAINS); do
    echo $domain
    dig +short TXT $domain | grep "v=spf1"
done
```

The first algorithm

Crude version of this program (everyone at a TLD registry wrote such a script at least once). Here, to test SPF records:

```
for domain in $(cat $DOMAINS); do
    echo $domain
    dig +short TXT $domain | grep "v=spf1"
done
```

Problems: does not scale, a few broken domains can slow it down terribly, unstructured output, difficult to extend to more complex surveys.

The architecture

It is composed of a generic socle, which handles:

- zone file parsing,
- and parallel querying of the zones.

and of a module which will perform the actual queries.

Modules

Thus, surveying the use of DNSSEC requires a DNSSEC module (which will presumably ask for DNSKEY records)

Modules

Thus, surveying the use of DNSSEC requires a DNSSEC module (which will presumably ask for DNSKEY records)

Surveying IPv6 deployment requires an IPv6 module (which will, for instance, ask for AAAA records for www.$DOMAIN and stuff like that).

Modules

Thus, surveying the use of DNSSEC requires a DNSSEC module (which will presumably ask for DNSKEY records)

Surveying IPv6 deployment requires an IPv6 module (which will, for instance, ask for AAAA records for www.$DOMAIN and stuff like that).

Not all techniques are amenable to DNS active querying: for instance, DKIM is not easy because we do not know the selectors.

Using it

Warning about the traffic

The program can generate a lot of DNS requests.
May be you need to warn the name servers admins.
As of today, uses a caching resolver, to limit the strain on the network.

Using it

Warning about the traffic

The program can generate a lot of DNS requests. May be you need to warn the name servers admins. As of today, uses a caching resolver, to limit the strain on the network.

UUID

To sort out the results in the database, every run generates a unique identifier, a UUID and stores it.

Options, arguments, . . .

Among the interesting options: run on only a random sample of the zone.

Complete usage instructions depend on the module

```
time $PROGRAM_NAME --num_threads=15000  \
       --debug=1 --module Dnssec fr.db --num_tasks=20
```

Reading the results

Querying of the database depends on the module.
Here, for DNSSEC:

```
SELECT domain,dnskey FROM Tests WHERE uuid='f72c33a6-7c3c-44e2-b

SELECT count(domain) FROM Tests WHERE uuid='f72c33a6-7c3c-44e2-b
                                  AND nsec;
```

Implementation

- Written in Python,
- The generic socle and the querying module are separated,
- Most modules store the results in a PostgreSQL database (we provide a helper library for that),
- Uses the DNS library dnspython from Nominum.

Everything works fine on small zones.

Larger zones may put a serious strain on the machine and on some virtual resources (lack of file descriptors, hardwired limits of `select()` on Linux...).

Parallelism

To avoid being stopped by a broken domain, it is **parallel**.

N threads are run to perform the queries.

For "`.fr`" (1.3 million domains), the optimal number of threads is around 15,000. The results are obtained in a few hours.

Developing a module

Several modules are shipped.

Should you want to develop one, you'll need mostly to write:

1. A class Result, with the method to store the result,
2. A class Plugin, with a method for the queries.

A Utils package is provided to help the module authors.

The example module

```
""" *dummy* module to illustrate what needs to be put in a
module. This module mostly prints things, that's all.

class DummyResult(BaseResult.Result):

    def store(self, uuid):
        print "Dummy storage of data for %s" % self.domain

class Plugin(BasePlugin.Plugin):

    def query(self, zone, nameservers):
        result = DummyResult()
        result.universe = 42 # Here would go the DNS query
        return result
```

Where are we in the talk?

Actual results

The data presented here were retrieved from "`.fr`" zones (march 2009).

The resolver used was Unbound, the machine was a two-Opteron PC, running Debian/Linux.

DNSSEC in “.fr”

Four hours for the run.

49 domains have a key. (That’s so low that sampling is a problem.)

But only 37 are actually signed (may be because of an error, such as serving the unsigned version of the zone file).

Side note: “.fr” is not signed, one domain in “.fr” is in the ISC DLV.

SPF in .FR

[RFC 4408]

188108 domains have SPF (15 %).

But there are only 4350 different records:

- Popular records like `v=spf1 a mx ?all`
- One big hoster added SPF for all its domains...

IPv6 in .FR

We measure several things:

- Presence of AAAA records for NS and MX
- Presence of AAAA records for `$DOMAIN`, `www.$DOMAIN`, ...
- Whether the machines reply to HTTP or SMTP connections.

IPv6, DNS only

When testing just the DNS, module runs during four hours and gives:

51355 (4 %) domains have at least one AAAA (Web, mail, DNS...)

410 (0,03 %) have a AAAA for all of the above three services.

Among the hosts, 435 different addresses. 24 are 6to4 and 8 are local (a lot of ::1...).

IPv6, with HTTP and SMTP tests

78630 IP addresses, 67687 (86 %) being HTTP. (For different addresses, HTTP and SMTP are 50/50.)

Among the 78630 addresses, 73122 (92 %) work (HTTP reply, even 404 or 500).

Warning: spurious addresses like ::1 are not yet excluded.

For the different addresses, only 292 (on 431, 67 %) work.

Wildcards?

227190 (18 %) have wildcards for at least one type.

Trends

Still too recent

No real change in the last months

IPv6 evolution

One big hoster activated IPv6 for all its Web sites in september 2008, completely changing the figures.

No such change occurred since. Things take time!

- "enabled": has at least one "important" service (Web, DNS, email) on IPv6
- "full": has all three important services availale via IPv6

Data

```
24 January 2009 : 0.03 %. full, 4 % enabled 17 January
2009 : 0.03 %. full, 4 % enabled 27 December 2008 : 0.04
%. full, 4 % enabled 13 December 2008 : 0.03 %. full, 4 %
enabled 02 December 2008 : 0.03 %. full, 4 % enabled
```

Where are we in the talk?

Future work on

- Asking directly the authoritative name servers, instead of going through a resolver.
- New modules, for instance testing the domains “email-only” or “web-only”. Or a module for Zonecheck “patrols”.

Distribution

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

Distributed under the free software licence GPL.

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 nice reports on trends.