On Queries to the Root

Geoff Huston & George Michaelson
research@apnic.net

The story so far…

- There have been a number of studies of the queries seen at the root
  - DNS OARC Workshops
  - Name Collision Workshop, March 14
- A number of desktop studies of the query behaviour of certain OS / Browser combinations when they perform name resolution (e.g. ICANN SECSAC report)
- But the two classes of studies are looking at each “end”
- How do they relate to each other?
  - How is local end system name resolution behaviour reflected as queries at the root?

OS System Library Behaviour

How do local search lists interact with user-provided names when performing name resolution when using the OS system library for name resolution?

never: the local search list is not used

always: the local search list is always appended to the name

pre: the local search list is appended, if NXDOMAIN, then the name is queried

post: the name is queried, if NXDOMAIN then the search list is appended

System\Query	Absolute *server.*	Relative Single Label *server*	Relative Multi-Label *www.server*
MAC OSX 10.9	never	always	never
Windows XP	never	always	post
Windows Vista	never	always	never
Windows 7	never	always	never
Windows 8	never	always	never
FreeBSD 9.1	never	pre	post
Ubuntu 13.04	never	pre	post

http://www.potaroo.net/ispcol/2013-10/dotless.html

OS + Browser Behaviour

MAC OSX 10.9

Browser\Query	**Single Label** *label1*	**Multi-Label** *label1.label2*
Chrome (31.0.1650.39)	always	post
Opera (12.16)	always*	never
Firefox (25.0)	always	never
Safari (7.0 9537.71)	always	never

** opera also looks up **www.**label1.(**com**|**org**|**net**|**edu**|**gov**)*

Windows 8.1

Browser\Query	**Single Label** *label1*	**Multi-Label** *label1.label2*
Chrome (30.0.1599.101 m)	always	never
Opera (17.0)	always	never
Firefox (25.0)	always	never
Safari (5.1.7 7545.57.2)	always	never
Explorer (11.0.900.16384)	always	never

http://www.potaroo.net/ispcol/2013-10/dotless.html

Can we…

Set up an experimental framework where the identity of the user's browser and OS can be linked to query behaviour as seen by the DNS root servers?

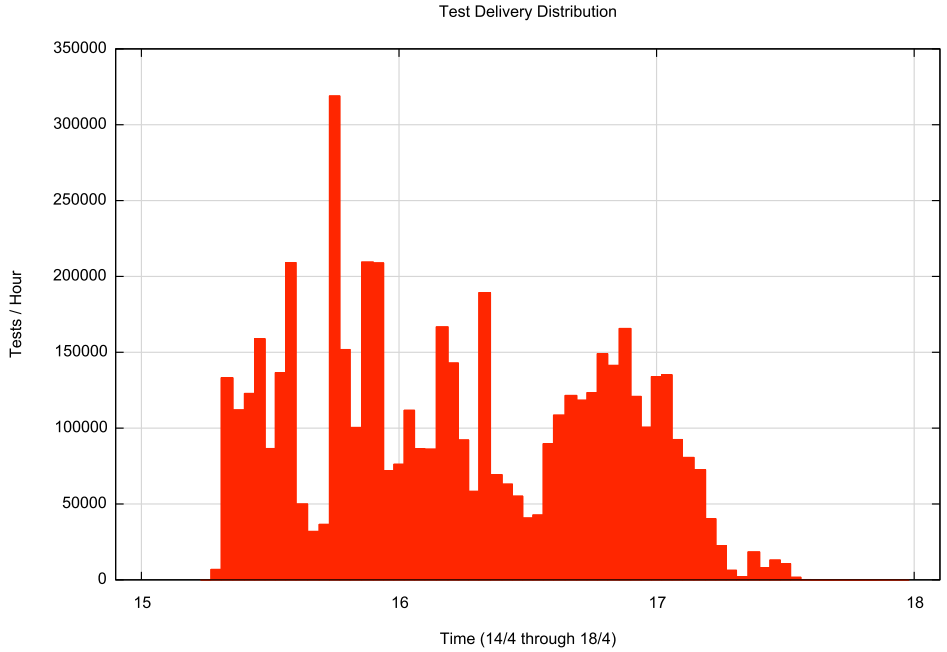
This Experiment

- Used Google's AD network to deliver a volume of end user tests, timed to coincide with the DNS OARC "Day In The Life" (DITL) data collection of queries made to root zone servers
- The embedded code used 4 URL elements:
  - URL with a single label (unique) name
  - URL with a two label (unique name)
  - URL to the experiment server

  (10 second wait)
  - URL to the experiment server

Experiment Statistics

- The Ad campaign was active from the 15th April to the 17th April 2014
  - 5,302,927 instances of the ad were seen to make HTTP queries to the experiment's servers



Root Queries

We used the DNS OARC DITL collection of logs of queries at root servers, and looked for queries that were the result of our experiment

- Logs were available for A, C, E, F, H, I, J, K, M roots
  - No logs from L ☹
- 11,177,623 DNS queries for the experiment's identifying string were found in the DITL log files
- 6,771,931 unique experiments made DNS queries that were seen at a root server

The Experiment

Two URLs were passed to the user's browser upon impression of the ad that are of interest with respect to root server queries

1. A single DNS label

   http://<*locate-string*>-<*unique-label*>-single-label-name/index.html

2. A 2-label name

   http://second-label.<*locate-string*>-<*unique-label*>-single-label-name/index.html

The Theory (1)

The single label name:

- If there is a local search string defined then this string will be appended to the label and the DNS query is formed
- If the search string is a defined name then the name servers for the name should be cached, and the query should not appear at the root
- If there is no local search string then the name should be queried, but as this is a unique name, it is not cache, so the query should be visible to the root servers (*)

The 2-label name:

- The name should not have any search string appended
- As the name is unique, the name should be the subject of a query to the root servers (*)

The Theory (2)

- The root server query logs should contain some single label queries (presumably where there are no locally defined search strings)
- A comprehensive root server query log collection should contain all 2 label name queries (*)

(*) Well… not quite “all”

- A number of recursive DNS resolvers (including Google’s PDNS) maintain a local copy of the signed root zone, and are able to respond to incoming queries from their local root zone without passing the query to a root server

The Practice

- The DITL root logs are incomplete
  - Not all root servers
  - Not all anycast instances
- Not all resolvers forward all queries relating to non-cached names to a root server

Some results

- Experiments seen at the root and seen at the server: 3,326,162
- Most of these align with theory
  - i.e. we see the two-label name queries at the root, but not the single label name

```
46.185.98.0 Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116 Safari/537.36
k A second-label.q1w2e3r46t-10002ee-single-label-name.

61.28.160.0 Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116 Safari/537.36
i A second-label.q1w2e3r46t-100031e-single-label-name.

37.244.130.0 Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116 Safari/537.36
j A second-label.q1w2e3r46t-1000344-single-label-name.
```

Collected Data Set

- End Client Public IP address
- End Client OS and Browser signature (User Agent String)
- The DNS resolver used by the End Client (at the end of any forwarder chain)
- The Root server
- The Root server anycast instance
- The query type
- The query string
- Time

Results (2)

However, some 539,445 experiments do not precisely align with this theory

- Single Label names are meant to have the local search string appended before querying the DNS
- Mostly this search string means that the resultant name is resolved without recourse to querying the root
- But sometimes these queries leak …

Local Search Strings

12,101 unique local search strings were seen

Top 25:

home.	55,997
homestation.	20,861
belkin.	6,168
lan.	5,047
dlink.	3,690
localdomain.	3,427
arris.	3,200
hitronhub.home.	3,148
domain.	2,347
local.	2,258
asus.	2,082
local.lan.	1,719
router.	1,029
co.yu.	869
fcname.	773
homerouter.cpe.	719
yu.	602
home.network.	518
private.	517
org.home.	466
gateway.	461
tendaap.	450
enhwi-n3.	439
wimax.	434
mynet.	429
btc-adsl.	422

Where's ".corp"?

.corp as a single label search string suffix is not widely used

But <name>.corp and corp.<name> are used, which makes this look like a widely used alternate coordinated name space, although this could easily be an outcome of local config recipes instead of a shadow tld

```
estudiodemoda.corp.    15
corp.int-ads.          13
eua.bvcorp.corp.       12
personal.corp.         11
global.ecolab.corp.    11
corp.local.            10
corp.pri.               9
ncs.corp.int-ads.       9
parcorretora.corp.      9
bvcorp.corp.            8
corona.corp.            8
res.hu.corp.            7
ihs.internal.corp.      7
lat.corp.               7
porta.corp.             7
iusacell.corp.          6
corp.oprema.local.      6
corp.vrx.               6
corp.                   6
corp.internal.          6
casetek.corp.pegatron.  6
heathco.corp.           5
farous.corp.            5
realogy.corp.rlg.       4
twn.rexchip.corp.       4
quanta.corp.            4
laz.bvcorp.corp.        4
centralcervejas.corp.   4
amer.zurich.corp.       4
```

And where's ".local"?

Much the same as .corp, in that there is widespread use of <name>.local and local.<name>

A similar picture is seen for "localdomain"

local.	2,258
local.lan.	1,719
sarmientoba.local.	219
local.tld.	89
domain.local.	52
dmu.local.	36
champestate.local.	32
buttcon.local.	31
lufrance.local.	29
dialok.local.	28
es.gov.br.local.	25
org.local.	21
riops.local.	20
cameiap.local.	19
metabenessere.local.	16
spec.local.	14
vescon.local.	14
gov.br.local.	14
tapdoanbaominh.local.	13
phlaurent.local.	13
net.local.	13
kld.local.	13
lutacom.local.	13
com.local.	11
local.larandia.	11
hattrick.local.	11
v6-632.local.	11
telecomitalia.local.	10
mifi.local.	10
dominio.local.	10

Name prepending…

```
pmpzajgi8cz9g.second-label.q1w2e3r46t-3c108cb618c058c0-single-label-name.
ze3gv3ye31jzq.second-label.q1w2e3r46t-3c109cb614af8b70-single-label-name.
u0p5rqvnaktql.second-label.q1w2e3r46t-3c10bcb614af8b70-single-label-name.
gcz616z-7ey6f.second-label.q1w2e3r46t-3c11fcb614afb280-single-label-name.
gq10xgmasspn.second-label.q1w2e3r46t-3c122cb614afb280-single-label-name.
d459jc760yb0k.second-label.q1w2e3r46t-3c122cb614afb280-single-label-name.
ym1apqonty69d.second-label.q1w2e3r46t-3c122cb614afb280-single-label-name.
sbyet9x3ttv3l.second-label.q1w2e3r46t-3c123cb614afb280-single-label-name.
uo396n6z46irq.second-label.q1w2e3r46t-3c125cb61acc2e50-single-label-name.
zm9a340nbe57l.second-label.q1w2e3r46t-3c12ccb61acc2e50-single-label-name.
okik8a7kqsfkj.second-label.q1w2e3r46t-3c132cb614afd990-single-label-name.
b70x6przr6gjo.second-label.q1w2e3r46t-3c135cb61acc5560-single-label-name.
b1py34eppcjsk.second-label.q1w2e3r46t-3c139cb61acc5560-single-label-name.
vmekjb648b5af.second-label.q1w2e3r46t-3c13ccb61acc5560-single-label-name.
b7q-ok2v6t2ol.second-label.q1w2e3r46t-3c140cb618c0cdf0-single-label-name.
yfvoss0eti1nk.second-label.q1w2e3r46t-3c143cb614afd990-single-label-name.
gybst9bkv9jap.second-label.q1w2e3r46t-3c143cb61acc5560-single-label-name.
ei3zq73xp8n7l.second-label.q1w2e3r46t-3c144cb614afd990-single-label-name.
kj7v0-j51ssbp.second-label.q1w2e3r46t-3c147cb618c0cdf0-single-label-name.
i77xgr83qbfpj.second-label.q1w2e3r46t-3c151cb614b000a0-single-label-name.
ig7fai29iitzn.second-label.q1w2e3r46t-3c152cb61acc7c70-single-label-name.
y77440po1p55d.second-label.q1w2e3r46t-3c15bcb614b000a0-single-label-name.
u3e0tz6jm0qri.second-label.q1w2e3r46t-3c160cb614b000a0-single-label-name.
h9fvkqnnj82il.second-label.q1w2e3r46t-3c161cb614b000a0-single-label-name.
k55k239xcig8m.second-label.q1w2e3r46t-3c164cb614b000a0-single-label-name.
v-9qgckt3qbzk.second-label.q1w2e3r46t-3c164cb61acca380-single-label-name.
m9tza3s7qmv7g.second-label.q1w2e3r46t-3c16acb614b000a0-single-label-name.
```

There are 322,490 instances of this, and there is no clear 1:1 association with a particular browser – is this some kind of DNS recursive resolver behaviour performing a form of wild card detection?

Name Prepending

Google

Patents

Find prior art | Discuss this patent | View PDF | Download PDF

Systems and methods for prepending nonce labels to DNS queries to enhance security

US 8484377 B1

Publication number	US8484377 B1
Publication type	Grant
Application number	US 12/903,349
Publication date	Jul 9, 2013
Filing date	Oct 13, 2010
Priority date ⓘ	Oct 13, 2010
Inventors	Jeremy K. Chen, Alexander D. Nizhner, Paul S. R. Chisholm
Original Assignee	Google Inc.
Export Citation	BiBTeX, EndNote, RefMan

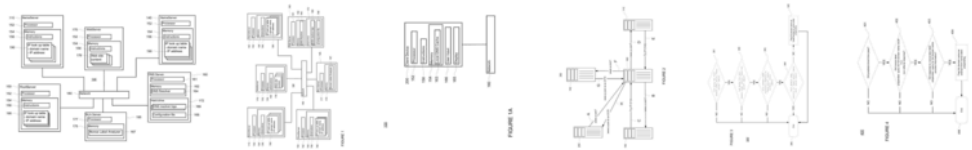
Patent Citations (2), Referenced by (1), Classifications (7), Legal Events (1)

External Links: USPTO, USPTO Assignment, Espacenet

ABSTRACT

A method for prepending nonce labels to DNS queries includes determining whether a log contains a past entry of a domain name resolution query ("query") to a name server for a full domain name that resulted in a positive reply indicating that the full domain name exists. It is determined whether the log contains a recent entry of the query that resulted in a negative reply indicating that the full domain name did not exist. The server is then queried with a nonce-less query for the full domain name. The server is queried again with a nonce label prepended query for the full domain name to determine if it currently results in the negative reply. The full domain name is flagged as inappropriate for nonce prepending upon determination that querying with a nonce prepended query results in a negative reply and a nonce-less query results in a positive reply.

IMAGES (6)



DESCRIPTION

FIELD OF INVENTION

The present invention relates generally to Internet security. More particularly, aspects of the invention relate to enhancing protection from hackers by appropriately pre-pending nonce labels to DNS queries.

BACKGROUND OF THE INVENTION

CLAIMS (16)

The invention claimed is:

1. A method for prepending nonce labels to DNS queries, the method comprising:

evaluating, with a processor, whether a log stored in memory contains at least one past entry of a domain name resolution query to a name server for a full domain name that resulted in a positive reply indicating that the full

Search Lists

- The .corp result is interesting:
  - Its not that large numbers of local DNS name resolvers add ".corp" to local DNS names prior to resolution
  - It's a little more subtle than that, and we see a significant set of instances where the local search string is a multi-label name that includes the label "corp"
- Out of 12,101 unique search strings seen, 1,654 used 2 or more labels

Local Search Lists and new TLDs

How many names from the set of new gTLDs are seen in these local search lists?

Out of 1,299 names: (http://icannwiki.com/index.php/All_New_gTLD_Applications)

- 121 names were seen in search strings
- 64,158 experiments used search strings containing applied names (out of 160,688 experiments that generated search string queries at a root)

Seen names and the search string occurrence as seen in this experiment

aaa	2
abc	1
acer	1
ads	12
airtel	1
amp	1
apple	1
art	1
asda	1
band	1
bank	1
bet	1
bingo	1
book	3
bosch	1
box	3
business	7
cam	1
casa	2
cba	1
center	2
cisco	4
clubmed	1
computer	1
corp	246
dell	1
dev	9
dhl	1
digital	1
dot	1
earth	1
ecom	2
exchange	1
fiat	1
fido	1
fox	1
garden	1
global	14
gold	3
google	1
group	16
health	1
here	1
home	105
hospital	1
host	4
hot	1
hotel	3
hotels	1
house	1
hsbc	9
hughes	1
ice	4
idn	1
iinet	2
imdb	1
inc	7
ing	1
kia	1
law	1
link	1
live	1
loan	1
lol	1
mail	2
matrix	1
mcd	1
med	3
microsoft	1
mma	1
mnet	4
mobile	1
moscow	1
movistar	2
msd	1
mtr	1
natura	1
network	9
new	1
nico	1
office	6
one	1
orange	8
pccw	1
prod	7
radio	1
ram	1
realty	1
safe	1
samsung	1
sarl	1
sbi	5
school	1
seven	1
sew	2
sfr	1
site	12
sky	1
sony	1
spa	2
sport	1
star	5
starhub	1
stc	2
svr	1
tata	1
team	1
telefonica	2
thai	1
tirol	1
top	2
toshiba	1
toyota	1
unicorn	1
web	2
win	5
windows	1
work	1
world	3
wow	1
youtube	1

Further Work

- Mapping users to root name server instances
- Measuring the “transparency” of undelegated gTLDs
- Number of root queries per trigger instance
- Consistency between predicted Browser/OS behavious and observed behaviours

Thanks

To Google for sponsoring our costs in the advertisement campaign used to generate the original use data for this analysis