DNS(SEC) client analysis

Bart Gijsen (TNO)  DNS-OARC, San Francisco, March 2011
Focus of DNS analysis has been on resolver and authoritative bulk data analysis
Key question:

› How will DNSSEC change the behavior of DNS client querying?

   More specific …

› How do DNS stub resolvers react to response types such as ServFail, responses > 512 Bytes, …?
Experimental set-up

Configure OS / browser on client machine

- **OS**: Windows XP, Windows 7, Ubuntu, Linux, Mac OS X
- **Browsers**: IE, Firefox, Chrome, Safari
- *not all combi’s, but quite some *
- clean OS image
- all settings left on defaults
Test execution

- Execute test run
  - query each URLs with predefined response (ldns tool)
    - Valid, Valid (>512 Bytes), NXdomain, Partial, ServFail, No reply, Truncated, Recursion refused
  - query via ping (=> OS only) and via browser (=> browser & OS)
  - repeat query once to check impact of caching

- Observe the number of repeated queries and delays
Example of DNS client behaviour: Linux-Ubuntu /w Firefox

15:26:38.694673 IP 10.0.3.2.96600 > 10.0.3.1.59: 7000 + AAAA? servfail.dnslab.nl. [36]
15:26:38.704094 IP 10.0.3.2.1.53 > 10.0.3.2.58600: 7000 ServFail 0/0/0 [36]
15:26:38.704779 IP 10.0.3.2.46093 > 10.0.3.1.59: 7000+ AAAA? servfail.dnslab.nl. [36]
15:26:38.711536 IP 10.0.2.1.53 > 10.0.3.2.58600: 7000 ServFail 0/0/0 [36]
15:26:38.712139 IP 10.0.3.2.34059 > 10.0.2.1.53: 751+ AAAA? servfail.dnslab.nl. [36]
15:26:38.720254 IP 10.0.2.1.53 > 10.0.3.2.34059: 751 ServFail 0/0/0 [36]
15:26:38.722147 IP 10.0.3.2.80615 > 10.0.2.1.53: 751+ AAAA? servfail.dnslab.nl. [36]
15:26:38.732281 IP 10.0.2.1.53 > 10.0.3.2.80615: 751 ServFail 0/0/0 [36]
15:26:38.732825 IP 10.0.3.2.35267 > 10.0.2.1.53: 624764+ ? servfail.dnslab.nl. [36]
15:26:38.741631 IP 10.0.2.1.53 > 10.0.3.2.35267: 624764 ServFail 0/0/0 [36]
15:26:38.742221 IP 10.0.3.2.55543 > 10.0.2.1.53: 624764+ ? servfail.dnslab.nl. [36]
15:26:38.750443 IP 10.0.2.1.53 > 10.0.3.2.55543: 624764 ServFail 0/0/0 [36]
15:26:38.750843 IP 10.0.3.2.85149 > 10.0.2.1.53: 40336+ A? servfail.dnsiac.nl. [36]
15:26:38.752003 IP 10.0.2.1.53 > 10.0.3.2.55546: 40336 ServFail 0/0/0 [36]
15:26:38.758693 IP 10.0.2.1.53 > 10.0.3.2.55546: 40336 A? servfail.dnsiac.nl. [36]
15:26:38.758759 IP 10.0.3.2.85149 > 10.0.2.1.53: 55536+ A? servfail.dnsiac.nl. [36]
15:26:38.769794 IP 10.0.3.2.33077 > 10.0.2.1.53: 55536+ AAAA? servfail.dnslab.nl. [36]
15:26:38.776757 IP 10.0.2.1.53 > 10.0.3.2.33077: 55536 ServFail 0/0/0 [36]
15:26:38.778971 IP 10.0.3.2.82085 > 10.0.2.1.53: 55536+ AAAA? servfail.dnsiac.nl. [36]
15:26:38.779225 IP 10.0.2.1.53 > 10.0.3.2.82085: 55536 ServFail 0/0/0 [36]
15:26:38.777582 IP 10.0.3.2.80122 > 10.0.2.1.53: 55536+ AAAA? servfail1.dnslab.nl. [36]
15:26:38.776045 IP 10.0.2.1.53 > 10.0.3.2.80122: 55536 ServFail 0/0/0 [36]
15:26:38.776995 IP 10.0.3.2.46728 > 10.0.2.1.53: 55536+ AAAA? servfail1.dnslab.nl. [36]
15:26:38.720932 IP 10.0.2.1.53 > 10.0.3.2.46728: 55536 ServFail 0/0/0 [36]
15:26:38.789735 IP 10.0.3.2.60114 > 10.0.2.1.53: 40195+ A? servfail.dnsiac.nl. [36]
15:26:38.800019 IP 10.0.2.1.53 > 10.0.3.2.60114: 40195 ServFail 0/0/0 [36]
15:26:38.810846 IP 10.0.3.2.35270 > 10.0.2.1.53: 40195+ A? servfail.dnsiac.nl. [36]
15:26:38.812326 IP 10.0.2.1.53 > 10.0.3.2.35270: 40195 ServFail 0/0/0 [36]
15:26:38.823351 IP 10.0.3.2.57149 > 10.0.2.1.53: 50408+ ? servfail.dnsiac.nl. [36]
15:26:38.820747 IP 10.0.2.1.53 > 10.0.3.2.57144: 50408 ServFail 0/0/0 [36]
15:26:38.839961 IP 10.0.3.2.35560 > 10.0.2.1.53: 50408+ ? servfail.dnsiac.nl. [36]
15:26:38.839032 IP 10.0.2.1.53 > 10.0.3.2.35560: 50408 ServFail 0/0/0 [36]

Example: servfail response

3 immediate retries in case of servfail response

and IPv4?

OS sends servfail to FireFox; Firefox makes OS retry

16 queries in 0.14 seconds
## Browser & OS DNS query amplification

<table>
<thead>
<tr>
<th>Response type</th>
<th>Firefox</th>
<th>Linux</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>x1</td>
<td>x1</td>
<td>x1</td>
</tr>
<tr>
<td>NXdomain / Partial</td>
<td>x2</td>
<td>x2</td>
<td>x4</td>
</tr>
<tr>
<td>ServFail / No response / Refused</td>
<td>x2</td>
<td>x4</td>
<td>x8</td>
</tr>
<tr>
<td>Truncated</td>
<td>x1</td>
<td>1+TCP</td>
<td>1+TCP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response type</th>
<th>Safari</th>
<th>Mac OSX</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>x1</td>
<td>x1</td>
<td>x1</td>
</tr>
<tr>
<td>NXdomain / Partial</td>
<td>x1</td>
<td>x2</td>
<td>x2</td>
</tr>
<tr>
<td>ServFail / No response / Refused</td>
<td>x1</td>
<td>x4</td>
<td>x4</td>
</tr>
<tr>
<td>Truncated</td>
<td>x1</td>
<td>1+TCP</td>
<td>1+TCP</td>
</tr>
</tbody>
</table>

**DNS query count in case of:**

- single authoritative NS; in case of primary and secondary => 2x
- only IPv4; in case of IPv4 and IPv6 => 2x
### Browser & OS DNS query amplification

<table>
<thead>
<tr>
<th>Response type</th>
<th>IE</th>
<th>Windows XP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid / NXdomain</td>
<td>x1</td>
<td>x1</td>
<td>x1</td>
</tr>
<tr>
<td>Partial / ServFail / Refused</td>
<td>x1</td>
<td>x1</td>
<td>x1</td>
</tr>
<tr>
<td>No response</td>
<td>x1</td>
<td>x5</td>
<td>x5</td>
</tr>
<tr>
<td>Truncated</td>
<td>x1</td>
<td>1+TCP</td>
<td>1+TCP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response type</th>
<th>Chrome</th>
<th>Windows XP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid / NXdomain</td>
<td>x1</td>
<td>x1</td>
<td>x1</td>
</tr>
<tr>
<td>Partial / ServFail / Refused</td>
<td>x1</td>
<td>x1</td>
<td>x1</td>
</tr>
<tr>
<td>No response</td>
<td>x1</td>
<td>x5</td>
<td>x5</td>
</tr>
<tr>
<td>Truncated</td>
<td>x1</td>
<td>1+TCP</td>
<td>1+TCP</td>
</tr>
</tbody>
</table>

In fact, same behaviour for IE, Chrome, Firefox, Safari on Windows XP or Windows 7
Other sources of aggressive DNS clients
(not investigated)

› Greedy – synchronisation apps: bonjour, facebook apps, ...
› may generate continuous stream of DNS requests

› Browser pre-fetching
› Firefox by default queries “anticipated next URLs” for a page
› Chrome pre-fetches stored, successfully retrieved URLs, when started

› Ubuntu Linux: by default no DNS caching
Impact of the caching resolver

- Some damping of aggressive client behaviour by (BIND9) resolver
  - In case of no-response the resolver retries (7 retries, with exponential timer back-off), while holding back client side retries
  - Valid, NXdomain and truncated responses are cached
  - TCP session for truncated responses is handled by resolver

- But also some amplification / modification by the resolver
  - Resolver ‘double checks’ ServFail responses
  - Unvalidatable response is returned as ServFail to client by non-DNSSEC enabled resolver
  - Also: partial, recursion refused and timeout are fed back as ServFail
Causes of aggressive DNS client behavior?

- GNU Library C (‘glibc’) DNS service
  - static code analysis:
    - overall glibc no ordinary characteristics found
  - dynamic code analysis of DNS part:
    - ‘responsible’ code part is pinpointed
    - code part is complex ⇔ improvement not found yet

- Ok, before we drill down to the cause … what’s the impact?
Impact model ("perfect behavior")
Impact on average DNS traffic volume

→ Predicted query load reduction as result of modifying aggressive Linux/Mac behavior is small
  → penetration of Linux / Mac OSX relatively low
  → behavior occurs in case of ‘exceptions’ (ServFail, NXdomain, …)
Impact outlook
- scenario: 10% DNSSEC validation error for SLD

- DNSSEC configuration errors at a domain will attract more traffic, due to observed behavior
Impact outlook
- scenario: NXdomain caching disabled at resolver

- Some amplification of bogus traffic to the Root
DNS client analysis

Experiments

Impact

Summary & next steps
Summary

- Linux and Mac clients display aggressive DNS behavior, in case of non-valid responses
  - Resolvers partly damp aggressive behavior, but also amplify it

- Impact of client behavior on average DNS traffic is relatively low
  - because fraction of Mac / Linux traffic is relatively low and
  - behavior occurs in particular for minority of DNS responses

- Although, for some particular cases the behavior amplifies traffic volume and rate
Next steps

- Share experiences with other experts

- Contribute to improving DNS function in the glibc(?)
  - alternative for pinpointed code part causing the amplification

- Further quantitative scenario impact analysis
  - further verification with ISP (SURFnet), SIDN data
  - compare to greedy apps behavior

- Is mobile internet different from other ISP traffic?
  - ABI Research: “in 2015 62% of mobile device will be Linux-based” …