IPv4 reverse measurements

Mattijs Jonker
Introduction

- Over five years ago, we started with an idea:
  
  "Can we measure (large parts) of the global DNS on a daily basis?"

- This idea led to the OpenINTEL project
  
  (Raffaele presented the gist of it earlier today)

- IN-ADDR.ARPA. is part of the global DNS, amirite?

- In this talk, I will discuss:
  
  - The (very recent) addition of reverse v4 measurements
  - Why, how
Reverse DNS 101

- Reverse DNS maps IP addresses to names
  
  ... using a reversed IP address as name
  
  e.g., 192.168.1.15 becomes 15.1.168.192.in-addr.arpa.

- Name space managed by IANA and the RIRs

- Delegated to address space holders when the address space is assigned
Why measure?

- Check consistency with forward DNS -- especially for e-mail reverse and forward DNS mapping must be consistent (part of MTA authentication)
- Provides visibility into cloud infrastructures and network infrastructural elements, e.g.:
  - Names reflecting in which data centres clouds VPSes are hosted
  - Names of router interfaces [Chabarek13, Huffaker14]
- Gain insight in address space usage
How we perform our measurements

- The measurement process involves two stages
  1. Active measurement
  2. Streaming and persisting data
Stage I: main measurement

- We want to measure efficiently -- first find parts of the name space that are actually delegated
- Intuition: perform SOA and NS queries for /8, /16 and /24 levels (in IPv4) to find delegation points
- Yields one of the following:
  - Delegation point
  - Empty non-terminal response (RFC 8020) -- indicating no delegation exists, but names exist below
  - NXDOMAIN -- there are no names below
Stage I: main measurement

- Adapted existing OpenINTEL measurement code
- Goal: one measurement every 24h
- Challenge: do not overload authoritative servers with queries
- Solution:
  - Randomize measurement
  - Monitor traffic for the first few measurement runs
Stage I: main measurement

- We use a similar trick to ZMap, that is: leverage properties of a group of prime order

- Need a permutation over 256 and 65536 possibilities for our implementation (to randomise individual labels in an IPv4 reverse name and to randomise /16 blocks sent to worker nodes respectively)

- Take the group of prime order $\mathbb{Z}_{p}^{\ast}$ for $p = 257$ or $p = 65537$
- Pick a generator $g \in \mathbb{Z}_{p}^{\ast}$ and a random starting element $r \in \mathbb{Z}_{p}^{\ast}$
- Because $\mathbb{Z}_{p}^{\ast}$ is of prime order it has $(p-1)$ generators, yielding 128 or 32768 possible permutations for $p = 257$ or $p = 65537$ respectively.
- Iterate over $\mathbb{Z}_{p}^{\ast}$ by repeatedly computing $r \cdot g$
Stage I: main measurement

- We adapted Duane Wessels' `dnstop` to track query loads and report average and maximum queries per second
- Result: average upstream loads very reasonable (maxing out around the 100 queries/second on average)
- Modified code: [https://github.com/rijswijk/dnstop](https://github.com/rijswijk/dnstop)
Stage II: storage and persistence

- Data is persisted in HDFS
  - allowing batch-based, analyses
- We stream the data to a Kafka cluster
  - enabling stream-based analysis
- Will clone data to CAIDA (WIP)
What do we have, in simple numbers

- Started measuring February 17, 2020
- This adds approx $1.1 \cdot 10^9$ data points each day (SOA, NS, PTR)
- 45% increase w.r.t. what we were already getting daily
Which data do we share

- This type of data: not yet
- No real obstacles
- Should probably think of *how*? and not *whom with*?
Case study: forward-confirmed rDNS

• Checked, for our “forward“ active DNS data, which IP addresses are forward confirmed

• 1.1M / 6.08M [18%] are
Case study: multi PTR

<table>
<thead>
<tr>
<th>dptr_name_count</th>
<th>dquery_name_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1149126160</td>
</tr>
<tr>
<td>0</td>
<td>15137003</td>
</tr>
<tr>
<td>2</td>
<td>4189153</td>
</tr>
<tr>
<td>6</td>
<td>208533</td>
</tr>
<tr>
<td>3</td>
<td>185014</td>
</tr>
<tr>
<td>4</td>
<td>47997</td>
</tr>
<tr>
<td>5</td>
<td>17503</td>
</tr>
<tr>
<td>17</td>
<td>9146</td>
</tr>
<tr>
<td>7</td>
<td>6818</td>
</tr>
<tr>
<td>8</td>
<td>5302</td>
</tr>
<tr>
<td>9</td>
<td>3669</td>
</tr>
<tr>
<td>10</td>
<td>3049</td>
</tr>
<tr>
<td>11</td>
<td>2576</td>
</tr>
<tr>
<td>12</td>
<td>2027</td>
</tr>
<tr>
<td>13</td>
<td>1586</td>
</tr>
<tr>
<td>14</td>
<td>1466</td>
</tr>
<tr>
<td>16</td>
<td>1011</td>
</tr>
<tr>
<td>15</td>
<td>988</td>
</tr>
<tr>
<td>20</td>
<td>850</td>
</tr>
<tr>
<td>18</td>
<td>746</td>
</tr>
<tr>
<td>22</td>
<td>653</td>
</tr>
<tr>
<td>21</td>
<td>609</td>
</tr>
<tr>
<td>19</td>
<td>585</td>
</tr>
<tr>
<td>27</td>
<td>580</td>
</tr>
<tr>
<td>23</td>
<td>548</td>
</tr>
</tbody>
</table>

only showing top 25 rows
Case study: multi PTR

```
dig +tcp -t ptr 71.184.197.146.in-addr.arpa @208.67.222.222
```
Cast study: Amazon EC2
Future work

- Verify consistency against existing work by CAIDA (Young Hyun)
- Check missing empty non-terminals on name servers that do not conform to RFC 8020
- Make data public: comments, thoughts?
Questions ?