DNS and IPv6 (and some IPv4 depletion stats)

Mark Beckett, VP Marketing and Product Management
Secure64 Software Corporation
Secure64 Software Corporation

Privately funded, Colorado-based corporation, founded in 2002

Focused on making the DNS trustworthy and secure

Secure64 products: ‘DNS Authority’, ‘DNS Signer’ & ‘DNS Cache’

All our products are IPv6 ready
Topics For Today

- Some IPv4 depletion statistics
- Quick primer on the DNS and IPv6
- A few real-world DNS operational issues
- A surprise benefit of IPv6 against cache poisoning
IPv4 Depletion

- RMv6TF 2010 meeting in Denver, April 21st.
  - Days to IANA depletion: 583 days
- TXv6TF meeting in Houston, November 4th
  - Days to IANA depletion: 488 days
- RMv6TF 2010 meeting in Denver, May 27th.
  - Days to IANA depletion: 300 days

IPv4 addresses will be exhausted by next year’s IPv6 conference
Many Steps to IPv6 only DNS

- AAAA resource records
- Function over IPv6 transport
- Necessary Intermediate Steps
- Handle IPv6 specific issues
- Features to help migrate away from IPv4
- Features to help migrate to IPv6

Most DNS implementations have only addressed these two
DNS and IPv6 Primer

- DNS is used to:
  - Map a hostname to an IP-address
  - Map an IP-address to a hostname
  - Identify servers for other protocols and systems (mail, AD, etc.)
- DNS “mandatory” in IPv6, even for internal hosts, router, switches, etc.
  - IPv6 addresses are 128 bits, hard to remember
  - Easier to SSH to router-10.secure64.com rather than 2001:12EF:1AB9:3391:4510:100F:8FFE:E63C
- If you put everything in the DNS, the only address you have to remember is the DNS server address
  - Pick an easy to remember address for your DNS server
  - In the simplest form, just add AAAA records for everything else
Reverse Delegation In IPv6

- Reverse delegation in IPv6 is done in the ip6.arpa zone
  - ip6.int deprecated

Address 4321:0:1:2:3:4:567:89ab would be:

b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.0.1.2.3.4.IP6.ARPA.
Real-World DNS and IPv6 Issues

- Populating reverse DNS

- Transitioning to IPv6
  - Running dual stack to the client
  - Running DNS64/NAT64
Reverse Delegation Issue

- All hosts on internet should have a reverse delegation (RFC)

- In reality, not always as easy as previous slide suggests
  - In IPv4 service providers pre populate the entire reverse tree:
    - adsl-70-250-178-4.dsl.rcsntx.swbell.net.
  - Reverse delegation of just a single /64 would require 4 billion 400 G disks of storage

- IETF draft lays out 4 alternatives:
  - Do nothing
  - Use wildcards
  - Use dynamic DNS
  - Synthesize records on the fly
## Comparison of IETF Options

<table>
<thead>
<tr>
<th></th>
<th>Do Nothing</th>
<th>Wildcards</th>
<th>Dynamic DNS</th>
<th>Synthesize</th>
</tr>
</thead>
<tbody>
<tr>
<td># new servers</td>
<td>0</td>
<td>0</td>
<td>Hundreds</td>
<td>0</td>
</tr>
<tr>
<td>Requires development</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Reverse record exists</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Reverse record matches forward record</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Works with DNSSEC</td>
<td>✓</td>
<td>Difficult</td>
<td>Difficult</td>
<td></td>
</tr>
</tbody>
</table>

DNS solutions need to evolve to simplify reverse IPv6 DNS.
Dual Stack Issue

- Dual stack is the IETF recommended transition mechanism, but
- There are problems when client IPv6 connection is broken
  - Extreme slowdown as client retries AAAA and then A lookups
- Estimated 0.078% of clients have this problem
  - Some older Opera browsers, some older Apple OSes, etc.
  - Amounts to millions of users for some large content providers like Google, Yahoo, etc.
- Dual stack is temporary, IPv6 only is the final goal

Today google is not giving out AAAA unless you are in IPv6 trials with them because it might break 0.078% of IPv4 clients = millions of users in google’s case
The lookup of A versus AAAA records is independent of whether the DNS packets are carried over IPv4 or IPv6:
- Client cannot know IPv4/IPv6 capabilities of the authoritative servers
- Authoritative server cannot know IPv4/IPv6 capabilities of the client
- Neither knows the IPv4/IPv6 capabilities of the intermediate network

Typically an IPv6 enabled client OS will send AAAA then A, but not always:
- Inconsistency across OSes is hard to deal with
- Combining this behavior with search domains (domain completion) can generate lots of DNS queries!

Increased client latency and DNS server load likely with dual stack!
One Proposed Solution Using DNS

- Caching side (ISP, consumer of content)
  - If query came in over IPv4, respond negatively to the AAAA request and wait for the A request

- Side effects:
  - Breaks DNSSEC
  - Turns off IPv6 for clients that can only do DNS queries over IPv4 (ie Windows XP)
An Alternative Transition Method

- Run pure IPv6 to client, not dual stack
  - Mandatory approach if you don’t have enough IPv4 addresses for dual stack
  - Only works when you control the client and the caching DNS server (think wireless providers, large internal networks)

- Must still be able to communicate with DNS servers that only support IPv4 and A records

- Use DNS64/NAT64 to bridge the gap
NAT64 / DNS64 Solution

- IETF draft
- IPv6-only network on the client side!
  - DNS rewrites A record responses to AAAA records using prefix
  - NAT64 translates IPv6 addresses to IPv4 and vice-versa
- User experience with NAT64 is (almost) the same as NAT44
- Only one network to maintain
NAT64 / DNS64 Under The Hood

---

Breaks hardcoded IPv4 addresses in web pages, but still compelling.
Kaminsky found a problem with DNS that allowed for cache poisoning.

The short term solution was to add a patch to do source port randomization.

The long term solution is DNSSEC:
- Digitally sign zones.
- DNSSEC is a complex standard, ZSK, KSK, rolling keys, signature expiration times, etc.

In the meantime, IPv6 offers additional protection from cache poisoning attacks.
The “Kaminsky Attack”

1. Send a query.
2. Send a lot of responses and guess the TX id (and port).
3. If unsuccessful goto 1

<table>
<thead>
<tr>
<th>Randomness Source</th>
<th>Unpatched DNS</th>
<th>Patched DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction ID</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Source Port</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Destination IP (avg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capitalization (avg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Bits</strong></td>
<td><strong>16</strong></td>
<td><strong>32</strong></td>
</tr>
</tbody>
</table>
The Patch is Not Enough

- Source port randomization makes the attack more difficult, doesn’t prevent it.
- High attack rates are easily reachable by botnets and compromised PCs.
- Patched code has already been compromised in <10 hours (using 80K-100K pps).
- A patient attacker can take more time to remain undetected.

<table>
<thead>
<tr>
<th>Attack Volume (pps)</th>
<th>Time to Poison</th>
<th>Number of random source ports</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
<td>1024</td>
</tr>
<tr>
<td>500</td>
<td>90 seconds</td>
<td>1 day</td>
</tr>
<tr>
<td>5,000</td>
<td>9 seconds</td>
<td>3 hours</td>
</tr>
<tr>
<td>50,000</td>
<td>1 second</td>
<td>16 minutes</td>
</tr>
</tbody>
</table>

The patch is a temporary fix – we need a long term solution.
How Can IPv6 Help You Here?

<table>
<thead>
<tr>
<th>Randomness Source</th>
<th>S64</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>16</td>
</tr>
<tr>
<td>Source Port</td>
<td>16</td>
</tr>
<tr>
<td>Destination IP (avg)</td>
<td>2</td>
</tr>
<tr>
<td>Capitalization (avg)</td>
<td>8</td>
</tr>
<tr>
<td>Source IP</td>
<td>5</td>
</tr>
<tr>
<td>Total Bits</td>
<td>47</td>
</tr>
</tbody>
</table>

Can we configure our DNS with say 32 IPv6 addresses and let the server pick IP-address randomly?

- 2 = 1 bit
- 4 = 2 bit
- 8 = 3 bit
- 16 = 4 bit
- 32 = 5 bit
- 64 = 6 bit
- 128 = 7 bit
- 256 = 8 bit
Questions?

Mark Beckett
Secure64 Software Corporation
mark.beckett@secure64.com
(303) 242-5899