IPv6 tunnel discovery

Lorenzo Colitti
Roma Tre University
RIPE NCC
IPv6-in-IPv4 tunnels

- Point-to-point link between two routers
- IPv6 uses IPv4 as its “link layer”
- IPv6 packets are encapsulated in raw IPv4 packets with Protocol set to 41
Problems with tunnels

Easy migration from IPv4, but:
- Low performance
  - Place heavy load on routers
  - May lead to inefficient routing
- Difficult to troubleshoot
- Pose security problems

To avoid them we must know they’re there
- Transparent to IPv6, “single-hop”
- What can we do?
- (What we can’t do: DNS)
Objective

- Find all the v6-in-v4 tunnels in the Internet
  - How?
- Start by finding tunnels on one path (A->B)
  - If this can be done, solve the problem (in principle) by iterating over all possible paths
  - For each path, find out:
    1. If the path is native or tunneled (easy)
    2. Which hops are tunnels (more difficult)
    3. The IPv4 endpoints of the tunnels (very difficult)
Approach

- **Basic tools:**
  - Path MTU discovery
  - Query Internet data sources (DNS, AS, registries)
  - Tunnel endpoint spoofing
  - Hop Limit (= TTL) manipulation
  - Routing Header (= source routing)

- Combine these to form rules

- Each rule has one or more of the following objectives:
  - Infer the existence of tunnels
  - Confirm their existence
  - Collect information about tunnel endpoints
  - Perform “third-party exploration”
Path MTU discovery

- MTU = largest packet a link can carry
- Path MTU discovery reports MTU decreases along path
- Allows us to find first tunnel in a path
  - MTU of tunnel usually lower than native links
  - Certain MTU values typical of tunnels
- Enough if we only want to know if path is native or not

MTU=1500  MTU=1480  ⇒ Tunnel?  MTU=1500

```plaintext
colitti@giga:~$ findmtu www.6net.org
colitti@giga:~$ findmtu orange.kame.net
colitti@giga:~$
```
Useful data sources

- **DNS**
  - Names are protocol-independent
  - Look up v4 address given v6 address (or vice versa)
  - If we suspect a tunnel, DNS can tell us the endpoint

- **AS lookups**
  - Many tunnels are between two different ASs
  - IPv4 address of an interface may be in different AS from IPv6 address of the same interface

- **Registry queries**
  - The 6bone registry contains information about tunnels
Endpoint spoofing

- Tunnels provide no authentication mechanism
- Any host Z that knows the IPv4 endpoints can "inject" packets into the tunnel
- We can send IPv6 packets as if connected to B
  - Similar to source routing, but Hop Limit is untouched
- Allows us to:
  - Confirm the presence of a tunnel (send a packet to ourselves)
  - Find more tunnels from B with MTU discovery (fragment encapsulating IPv4 packets)
  - Find IPv6 addresses of A and B using Hop Limit manipulation and source routing

A = IPv4 address of A
A = IPv6 address of A
A possible algorithm to find tunnels in a path:
- Look for one tunnel at a time
- Use Path MTU discovery to find next tunnel
- Use DNS to discover tunnel endpoints
- Use endpoint spoofing to confirm tunnel
- Use endpoint spoofing to carry on discovery process from last tunnel
- Repeat until end of path reached

Unfortunately, DNS does not provide enough information to make this feasible
- Information in DNS is incomplete, inaccurate, or old

Is there another way?
What else can we do?

- Endpoint spoofing tells us if a tunnel is working or not
  - Given two addresses see if there’s a tunnel between them
  - Use this to verify the quality of data in the 6bone registry
    - How many tunnels in the registry are actually working?
- Path MTU discovery tells us if a path is native or not
  - What percentage of the Internet is native?
  - For example, measure MTU to every prefix in the BGP table
    - This gives us one view of the Internet
    - Endpoint spoofing lets us to do this from any tunnel we know
    - For example, from all the tunnels in the 6bone database
How many tunnels?

- The 6bone registry contains >1000 “working” tunnels (~25%)
- The rest nonexistent (~50%), down or filtered
- We measured the MTU from each endpoint to all prefixes in the global IPv6 BGP table
- Result: native paths only ~ 8% of total

<table>
<thead>
<tr>
<th>MTU</th>
<th># paths</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1480</td>
<td>150946</td>
<td>39.4</td>
</tr>
<tr>
<td>1280</td>
<td>138358</td>
<td>36.1</td>
</tr>
<tr>
<td>1476</td>
<td>44404</td>
<td>11.6</td>
</tr>
<tr>
<td>1500</td>
<td>31525</td>
<td>8.2</td>
</tr>
<tr>
<td>1428</td>
<td>13619</td>
<td>3.6</td>
</tr>
<tr>
<td>Other</td>
<td>4104</td>
<td>1.1</td>
</tr>
<tr>
<td>Total</td>
<td>382956</td>
<td>100.0</td>
</tr>
</tbody>
</table>
How native is “native”? 

Percentage of BGP prefixes reached through tunnel(s) from GARR and RIPE NCC / testboxes. Even the “best” are ≥ 62%
The TTM service will soon use tunnel detection to qualify IPv6 data.

Every testbox measures MTU to every other testbox and results are displayed in a matrix.

Historical data will provide information on routing changes.

Most testboxes are in native IPv6 networks.
Conclusions

- **Tunnel detection**
  - Native / tunneled path detection is easy
  - Finding more than one tunnel in a path is harder
  - Finding the endpoints is very difficult
    - Problem: incomplete and/or inaccurate DNS information

- **6bone database**
  - 50% of tunnels nonexistent
  - 25% working

- **IPv6 largely relies on tunnels**
  - In total, 8% of paths native
  - Even “native” networks are not better than 40% native