Reverse Traceroute

Valentin Heinrich

The problem

1 routerA.aug.net-a.com (10.10.10.10) 1ms 2ms 1ms
2 routerB.muc.net-a.com (20.20.20.20) 5ms 6ms 12ms
3 routerC.fra.net-a.com (30.30.30.30) 11ms 21ms 14ms
4 routerD.fra.b-net.com (40.40.40.40) 340ms 320ms 350ms
5 www.example.com (50.50.50.50) 345ms 310ms 360ms

Hops traceroute shows
The likely interface IPs you'll see
Packets on the forward path
Packets on the reverse path
Routers on the reverse path
One past attempt

- "Traceroute Using an IP Option", RFC 1393, January 1993
  - A special IPv4 option is added to TR packets (incl. the IP address of the originator)
  - Causes a router to send a special TR message to the originator
  - Packet with the option is simply forwarded
  - The receiver also sends a packet incl. above option with the originators address

- Why don’t we have this yet?
  - Well, likely the need for router support and the use of IP options
  - Could spoof originator’s IP / Amplification attack vector
  - It teaches us to be careful with design choices
  - RFC 1393 was obsoleted in 2012

- Our design goals: https://youtu.be/Y7NtqLEtgjU
Meet reverse traceroute

- Uses a new ICMP request to trigger a reverse traceroute
- One request per TR packet
Meet reverse traceroute

- A regular TR packet is sent (UDP, ICMP or TCP)
- Fields for load-balancing can be controlled

The likely interface IPs you'll see

- Routers reverse traceroute shows
- Routers on the forward path
Meet reverse traceroute

- For that single probe, an ICMP response is sent back
• Reverse Traceroute is defined for both ICMP and ICMPv6
• ICMP messages typically start like this:

```
|     Type      |     Code      |          Checksum             |
+---------------+---------------+-----------------------------|
```

● Question, which Type and Code to use:
   ○ Option A: New types and codes
   ○ Option B: Existing type and new codes

● But, which ones work on today’s internet?
What about middleboxes?

- The internet is ossified, mainly thanks to middleboxes
  - NATs e.g., are a pretty common middlebox
- Question: which packets go through NATs
- Tested 12 NAT implementation:
  - We sent two packets with type 8 (used by ping request) and codes 1 and 2 (standard ping uses 0), replies matched the code but used type 0
  - And two unassigned types (7 and 252) with code 0 each

<table>
<thead>
<tr>
<th>ICMP request</th>
<th>forwarded</th>
<th>filtered</th>
<th>bypassed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 8, code 1</td>
<td>11</td>
<td>1 a)</td>
<td>0</td>
</tr>
<tr>
<td>Type 8, code 2</td>
<td>11</td>
<td>1 a)</td>
<td>0</td>
</tr>
<tr>
<td>Type 7, code 0</td>
<td>1</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Type 252, code 0</td>
<td>1</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

1) Response dropped
But what happens to those packets on the internet?

- We picked ten million IPv4 addresses at random and send an ICMP Echo request there (good old Ping)
- For each host that responded, we sent an ICMP Packet with the Echo type but a different code (code 1)

<table>
<thead>
<tr>
<th>Filtered</th>
<th>Reflective</th>
<th>Unreflective</th>
<th>Erroneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>39.993</td>
<td>931.427</td>
<td>32.478</td>
<td>659(^a)</td>
</tr>
</tbody>
</table>

\(^a\) mostly dest. unreach.
Conclusion

● Presented at operator event (DENOG14)
  ○ Operational community quite positive about the proposal
● Running code
  ○ Github:  https://github.com/HSAnet/reverse-traceroute
  ○ Debian package available for both server (eBPF) and client (Python)
  ○ Contact:  valentin.heinrich@hs-augsburg.de
● Next steps
  ○ Finish the protocol design (mostly done, missing feedback)
  ○ Ask for WG adoption