Detecting and Measuring IPv4 and IPv6 NAT

Carlos Martinez-Cagnazzo

IEPG, Seoul, South Korea

November 2016

Measuring NAT from the browser

- Detecting if a browser is behind a NAT box by running a piece of JavaScript code
  - The usual approach for async resource fetching: **XMLHttpRequest**
    - Doesn’t support STUN requests
  - However **WebRTC** is now being implemented in major browsers and supports STUN requests.
    - Exposes the STUN response available to the JavaScript code that created the RTC Peer Connection.
    - Completely transparent to the end user.

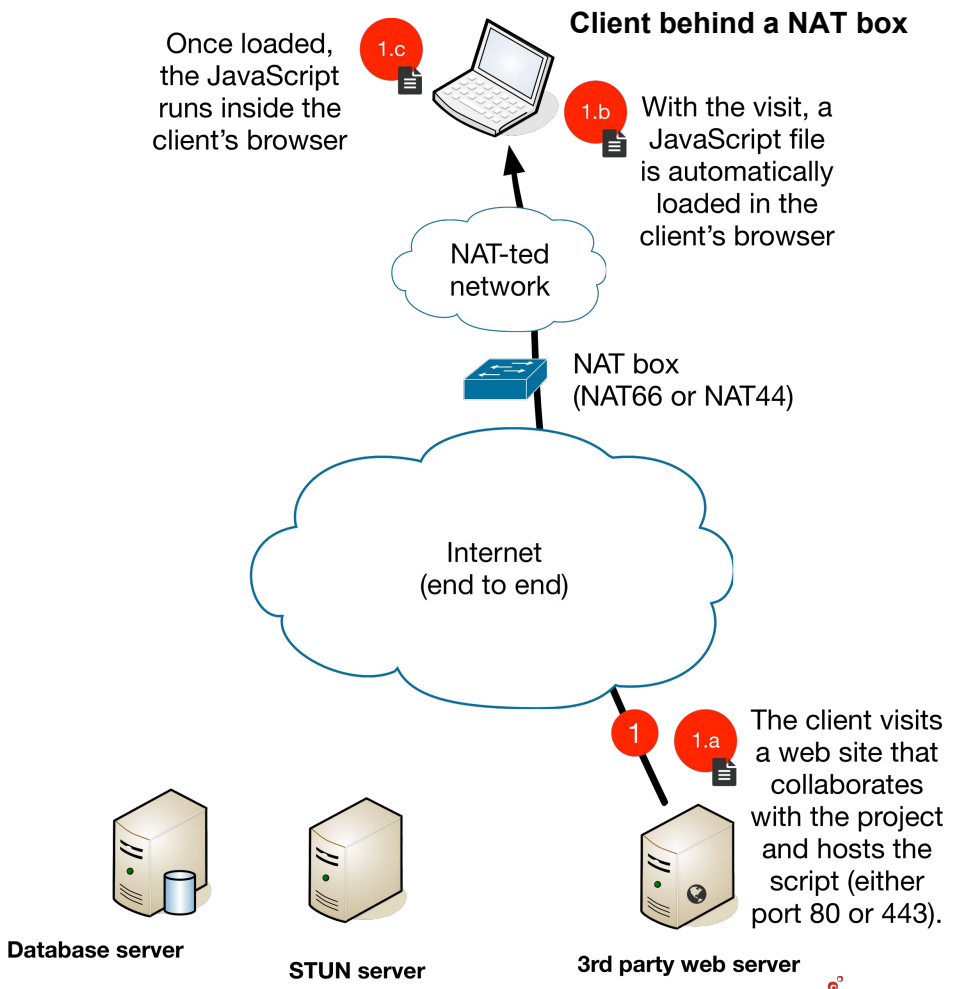
WebRTC & STUN

- **WebRTC** is a free, open specification that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs

- **STUN** (Simple Traversal of UDP through NATs (Network Address Translation)) is a protocol for assisting devices behind a NAT firewall or router with their packet routing
  - RFC 5389 redefines the term STUN as 'Session Traversal Utilities for NAT' (voip-info.org/wiki/view/STUN)

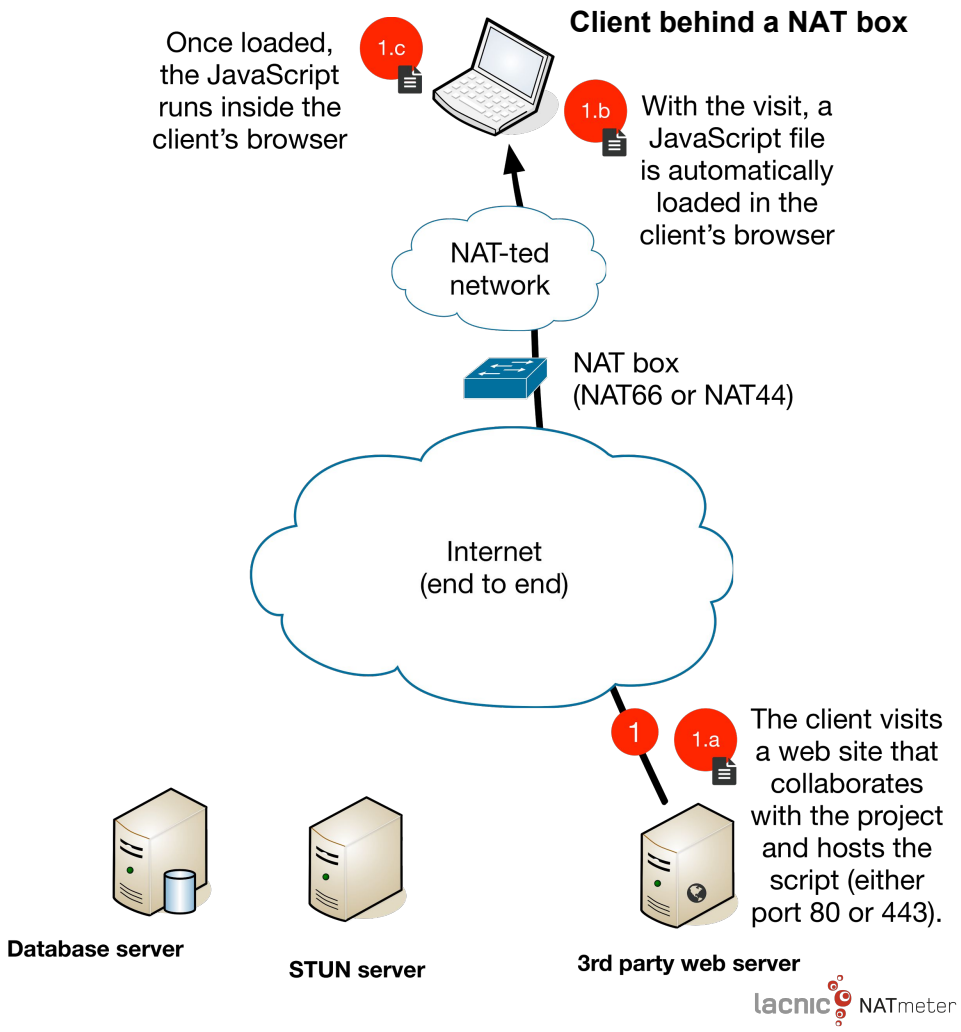
How does it work?

1. A Javascript testing probe is hosted in a participating web property
    a. A user visiting any one of these websites triggers the JS script, which is loaded and executed by the browser



How does it work?

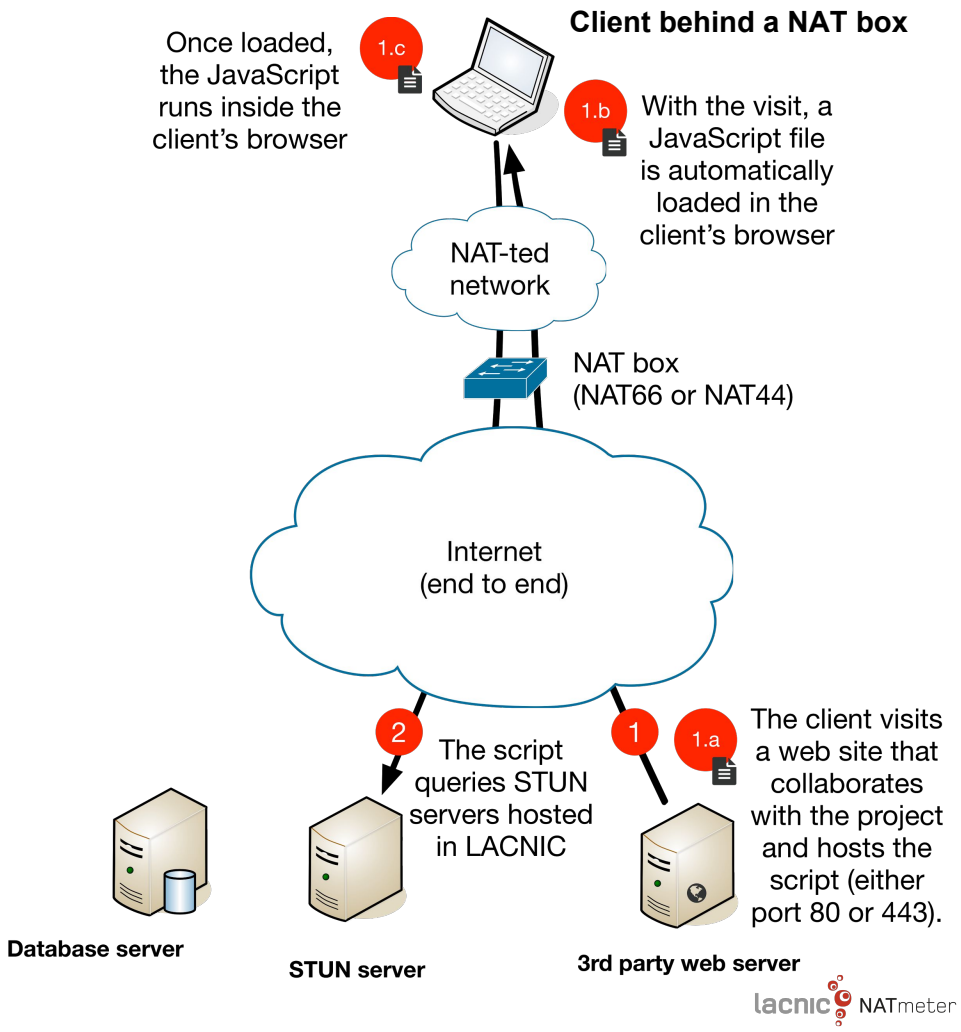
1. A Javascript testing probe is hosted in a participating web property
    a. A user visiting any one of these websites triggers the JS script, which is loaded and executed by the browser
    b. Runs silently in background



How does it work?

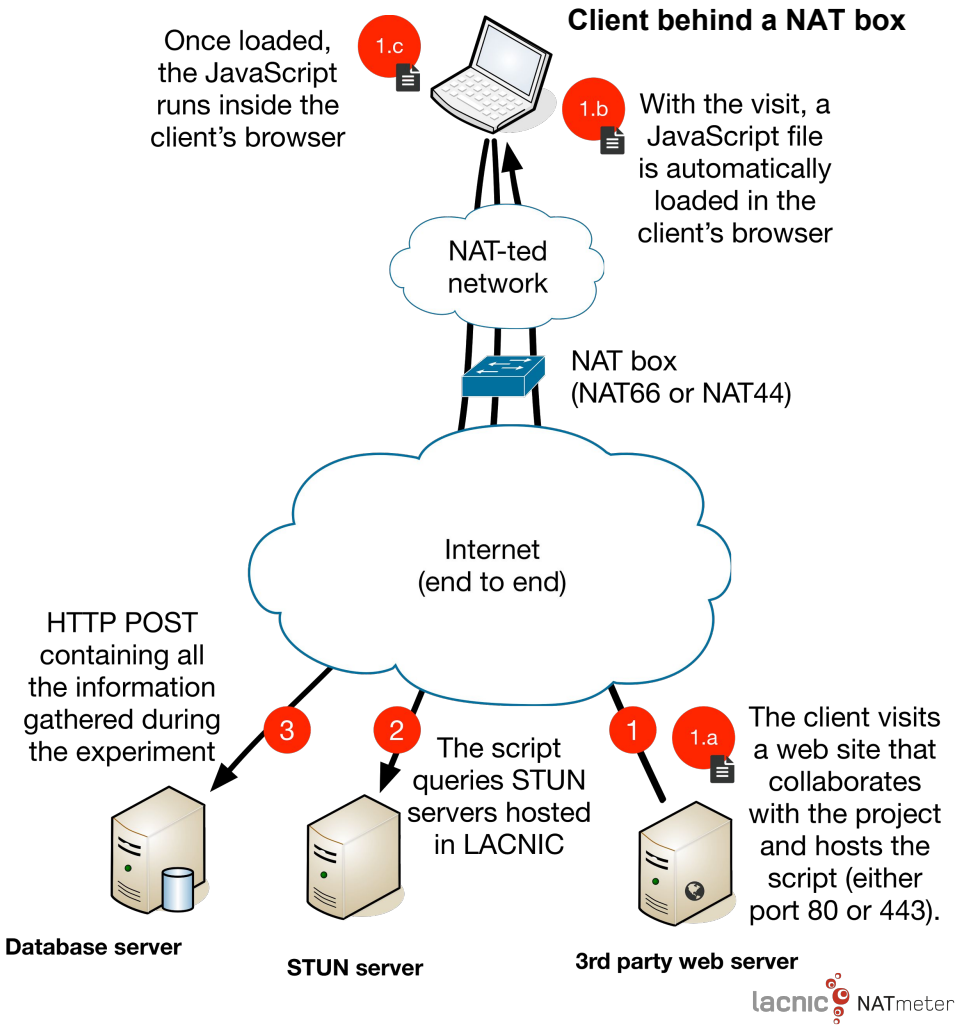
1. A Javascript testing probe is hosted in a participating web property
   a. A user visiting any one of these websites triggers the JS script, which is loaded and executed by the browser
   b. Runs silently in background

2. The hosts talks with the STUN Server



How does it work?

1. A Javascript testing probe is hosted in a participating web property
   a. A user visiting any one of these websites triggers the JS script, which is loaded and executed by the browser
   b. Runs silently in background
2. The hosts talks with the STUN Server
3. POST the results back to a central collector



How does it work? (in short)

- The JS script will instance two (or three) "`new RTCPeerConnection`" targeting
  - localhost
  - v4-only (and v6-only) STUN servers hosted by LACNIC.
- Localhost or the STUN servers answer back with information regarding the client's host addresses and the client's perceived addresses from the the public Internet
  - When the responses do not match, the user is behind NAT
- Results are posted to a central collector database

Note: Currently running Stuntman version 1.2.8 (http://www.stunprotocol.org/) on Ubuntu 13.04. Two separate servers, one for IPv4 and one for IPv6

Some results

Metric	Value
NAT 44	95.1 %
NAT 66	0.8 %
V6-only hosts	0 %
Dual stack hosts	22.5 %
NPT usage	0 %
Amount of v4 addresses p/host	Avg.: 1.1; Max.: 11
Amount of v6 addresses p/host	Avg.: 1.1; Max.: 9
The two most used IPv4 prefixes behind NAT	1. 192.168.1.0 2. 192.168.0.0

NAT66 example output

```
alejandro@simon:~$ ./nat_measurements.py NAT66
('Natted IPv6 Host', ['2800:XX::2'], 'IPv6 private addresses: ', [['fd00:88aa:cafe::3']])
('Natted IPv6 Host', ['2001:XX:abdc'], 'IPv6 private addresses: ', [['fc00:XX:abcd']])
('Natted IPv6 Host', ['2a03:XX::9e'], 'IPv6 private addresses: ',
[['fdd8:a2de:468c:72::107e']])
('Natted IPv6 Host', ['2001:XX:c44c'], 'IPv6 private addresses: ', [['2001:XX:ff31']])
('Natted IPv6 Host', ['2001:XX:ce0d'], 'IPv6 private addresses: ',
[['4006:e024:680:ce0c:3435:ed62:b2a9:5f60']])
('Natted IPv6 Host', ['2001:XX:ce0d'], 'IPv6 private addresses: ',
[['4006:e024:680:ce0c:3435:ed62:b2a9:5f60']])
('Natted IPv6 Host', ['2001:XX:3ad5'], 'IPv6 private addresses: ', [['2001:XX:3ad5']])
('Natted IPv6 Host', ['2001:XX:8678'], 'IPv6 private addresses: ', [['2001:XX:fe99']])
('Natted IPv6 Host', ['2001:XX:77d8'], 'IPv6 private addresses: ', [['2001:XX:fedc']])
('Natted IPv6 Host', ['2001:XX:1005'], 'IPv6 private addresses: ', [['2001:XX:fffb']])
```

NAT44 example output

```
alejandro@simon:~$ ./nat_measurements.py NAT44
('Natted IPv4 Host', [['172.16.29.52']], 'public', [['196.XX.114']])
('Natted IPv4 Host', [['10.200.41.45']], 'public', [['200.XX.253']])
('Natted IPv4 Host', [['10.181.28.199']], 'public', [['201.XX.37']])
('Natted IPv4 Host', [['10.0.80.227']], 'public', [['208.XX.64']])
('Natted IPv4 Host', [['192.168.177.1'], ['192.168.224.1'], ['192.168.0.11']], 'public',
[['186.XX.95']])
```

Let’s review the results of NAT66 (just for fun)

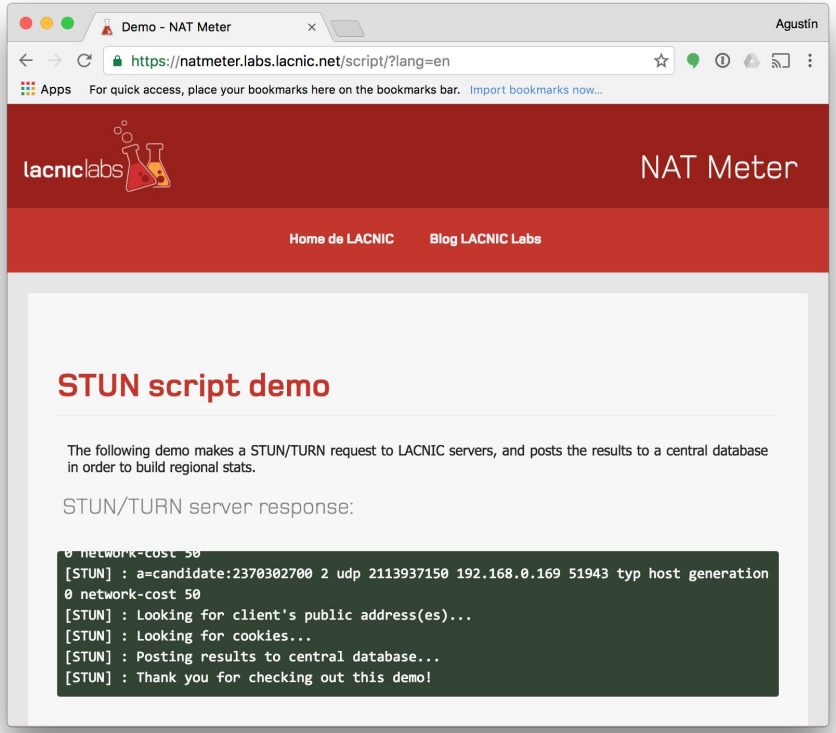
- Looks like the “private” address the people tends to use is ULA, good!

- But the squatters are there, just like in old IPv4-land: *'4006:e024:680:ce0c:3435:ed62:b2a9:5f60', not so good!*

- We also found cases testing positive for NAT66 using the same addresses within the same /64

Want to try? https://natmeter.labs.lacnic.net/script/

- Only Chrome is supported at the moment
- Want to contribute??
  - Will be distributing the script by the end of year. Please do contact {carlos | alejandro | agustin}@lacnic.net
  - More info coming soon!



Some final notes

- The dataset
  - Populated by lacnic.net visitors and some other regional blogs.
  - Normalization using the % of advertised IP addresses in the global routing table at a country level (1 sample from Brazil weighs more than 1 sample from Guyana)
  - Some results ignored: those from inside LACNIC itself for example
  - Started on Sep. 9th 2016 (still ongoing, no finish date set)
  - 25 K samples so far
- Geolocation
  - Using Maxmind
- Code at github.com/LACNIC/natmeter, based on github.com/diafygi/webrtc-ips

Thanks! Questions?

natmeter.labs.lacnic.net