Hic Sunt NATs: Uncovering Address Translation with a Smart Traceroute

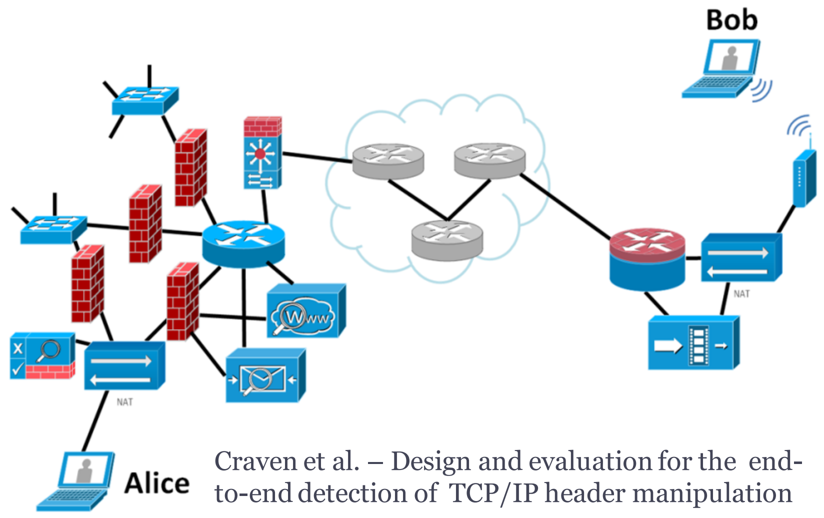
Raffaele Zullo, Antonio Pescapé, Università di Napoli Federico II, Italy
Korian Edeline, Benoit Donnet, Université de Liège, Belgium

<r.zullo@studenti.unina.it>, <pescape@unina.it>, <firstname.name@ulg.ac.be>

This work has been partially funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 688421. The opinions expressed and arguments employed reflect only the authors' views. The European Commission is not responsible for any use that may be made of that information. The work of Antonio Pescapé has been partially supported by the art. 11 DM 593/2000 for NM2 srl.

Roadmap

Middleboxes

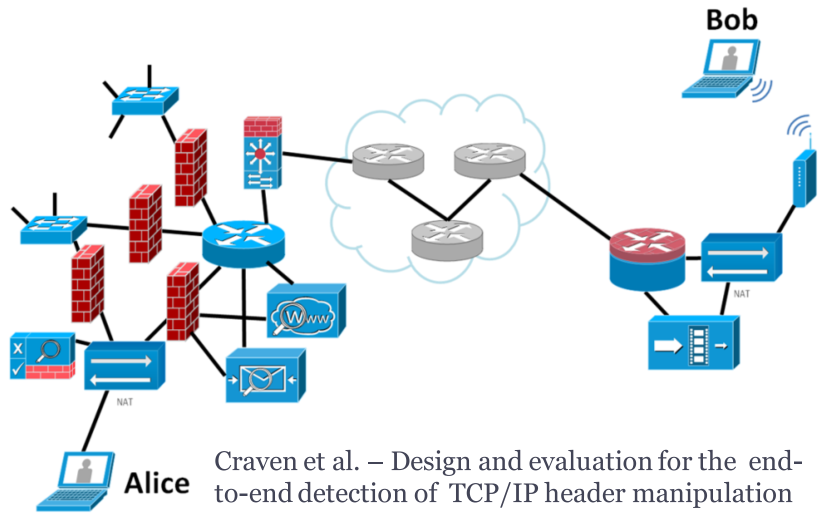


Craven et al. – Design and evaluation for the end-to-end detection of TCP/IP header manipulation

Middlebox
"Intermediary boxes performing functions apart from normal, standard functions of an IP router on the data path between a source host and a destination host" Lixia Zhang, 1999

- Purposes
  - dealing with limited resources
  - improving performance
  - securing hosts
- Drawbacks
  - brake protocols evolvability
  - impoverish performance
  - alter end-to-end communications

Middleboxes & NATs



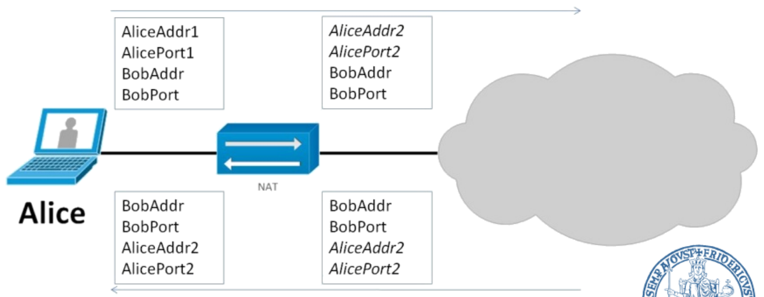
Craven et al. – Design and evaluation for the end-to-end detection of TCP/IP header manipulation

Middlebox
"Intermediary boxes performing functions apart from normal, standard functions of an IP router on the data path between a source host and a destination host" Lixia Zhang, 1999

•Purposes
- •dealing with limited resources
- •improving performance,
- •securing hosts

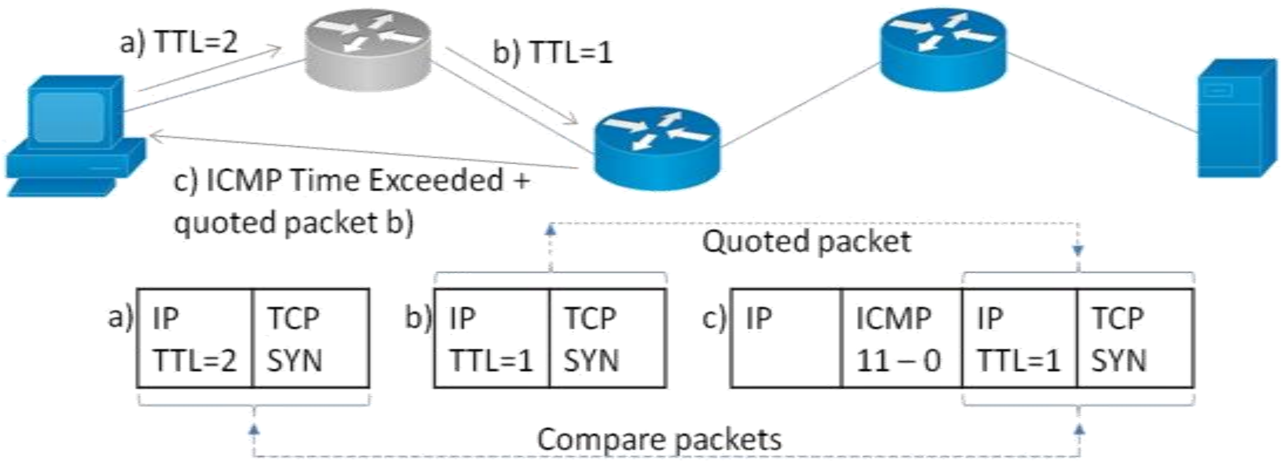
•Drawbacks
- •brake protocols evolvability
- •impoverish performance
- •alter end-to-end communications

NAT (NAPT)
Network Address (and Port) Translation
•First examples date back to 1994
•Progressive IPv4 addresses depletion
•IPv6 adoption still in its infancy



Tracebox

- Extension to **`traceroute`**, Developed by Detal et al. (2013)
- Tracebox extracts the original packet inside the ICMP Time Exceeded message and compares it to the packet that has been sent
- Middlebox **detection** and **location**
- Technique made possible by RFC792 and RFC1812

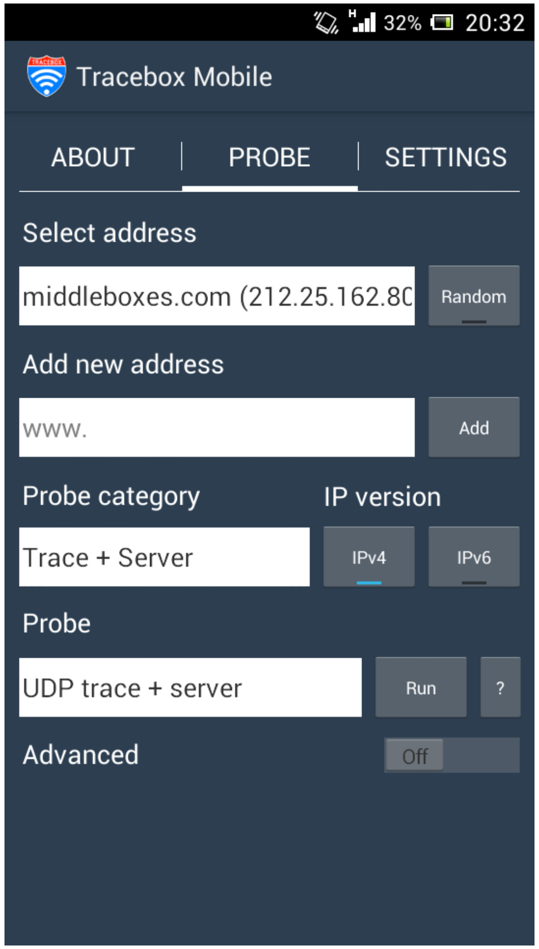


Mobile Tracebox

- TraceboxAndroid
  - Porting of `tracebox` to mobile devices
  - Developed by Thirion et al. (2015)
  - Proof-of-concept version
  - Rooted devices only
    - CAP_NET_RAW POSIX capability
- Mobile Tracebox
  - All Android platforms (arm, x86, mips, 32 and 64 bit)
  - All possible TCP and UDP probes over IPv4 or IPv6
  - Full customization of every single field and parameter
  - Refinements to standard tracebox mode
  - Server-based mode (also available on non-rooted devices)
- Data collection
  - Crowdsourcing
    - Instant probing
    - Backward probing

Mobile Tracebox

https://play.google.com/store/apps/details?id=be.ac.ulg.mobiletracebox



Select address: middleboxes.com (212.25.162.80

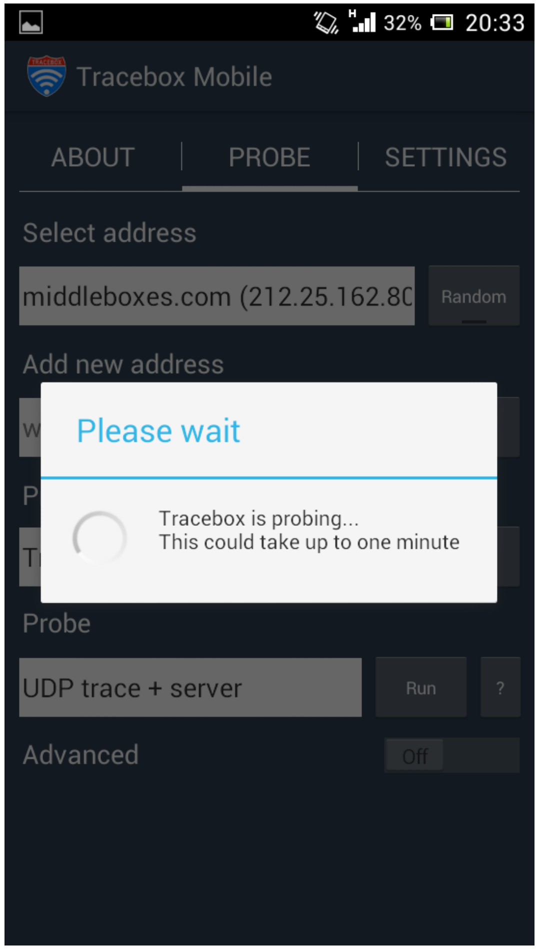
Add new address: www.

Probe category: Trace + Server

IP version: IPv4

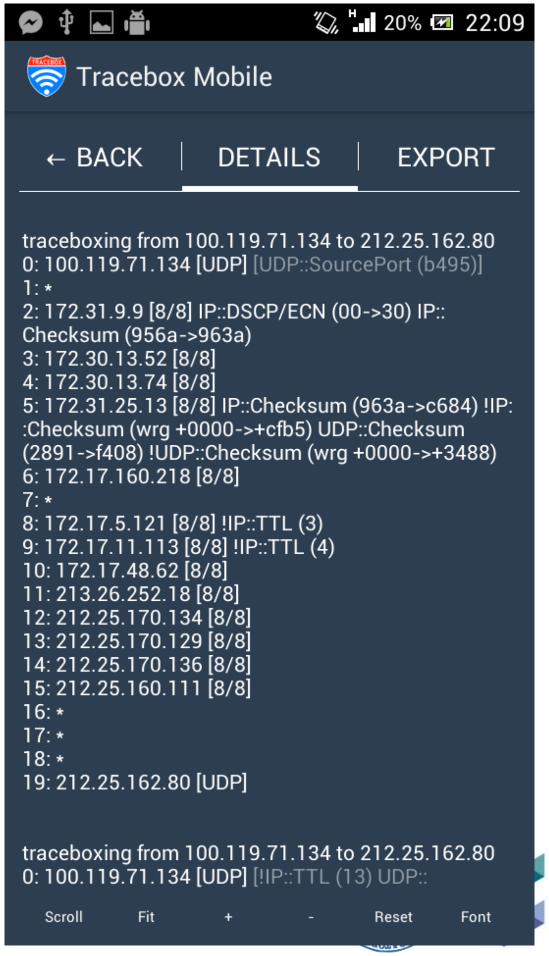
Probe: UDP trace + server

Advanced: Off



Please wait

Tracebox is probing...
This could take up to one minute

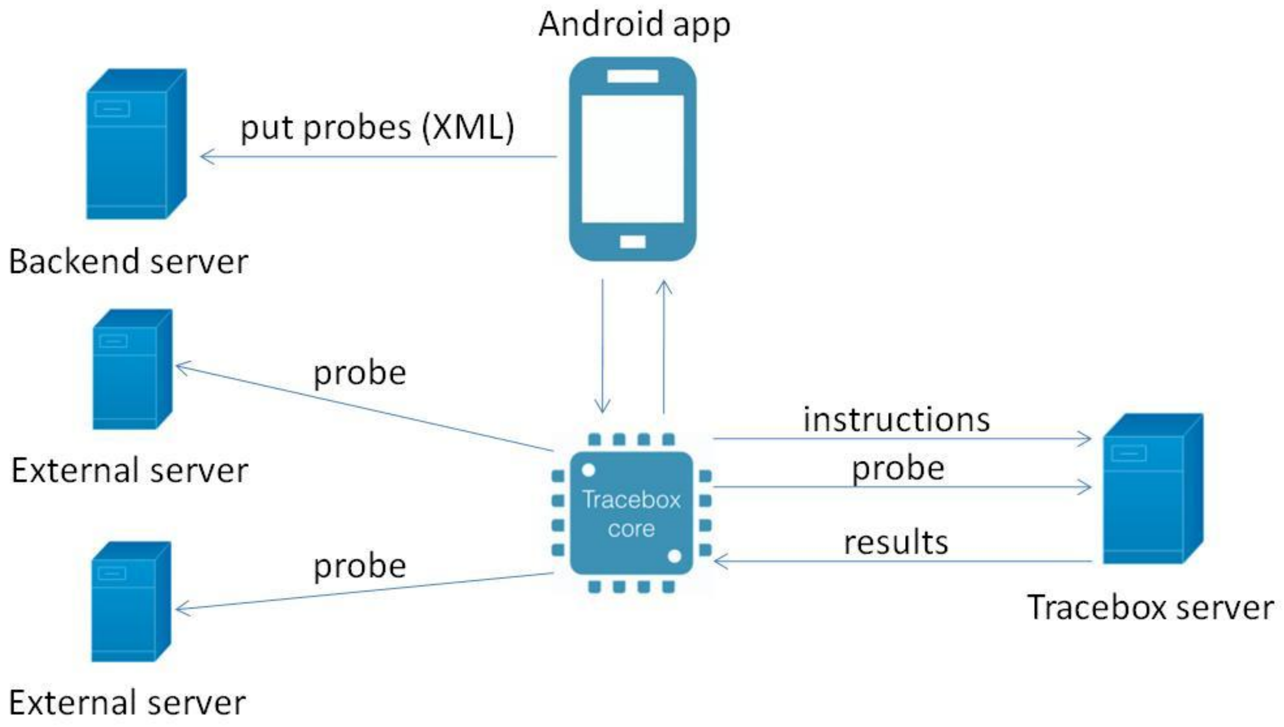


```
traceboxing from 100.119.71.134 to 212.25.162.80
0: 100.119.71.134 [UDP] [UDP::SourcePort (b495)]
1: *
2: 172.31.9.9 [8/8] IP::DSCP/ECN (00->30) IP::
Checksum (956a->963a)
3: 172.30.13.52 [8/8]
4: 172.30.13.74 [8/8]
5: 172.31.25.13 [8/8] IP::Checksum (963a->c684) !IP:
:Checksum (wrg +0000->+cfb5) UDP::Checksum
(2891->f408) !UDP::Checksum (wrg +0000->+3488)
6: 172.17.160.218 [8/8]
7: *
8: 172.17.5.121 [8/8] !IP::TTL (3)
9: 172.17.11.113 [8/8] !IP::TTL (4)
10: 172.17.48.62 [8/8]
11: 213.26.252.18 [8/8]
12: 212.25.170.134 [8/8]
13: 212.25.170.129 [8/8]
14: 212.25.170.136 [8/8]
15: 212.25.160.111 [8/8]
16: *
17: *
18: *
19: 212.25.162.80 [UDP]


traceboxing from 100.119.71.134 to 212.25.162.80
0: 100.119.71.134 [UDP] [!IP::TTL (13) UDP::
```

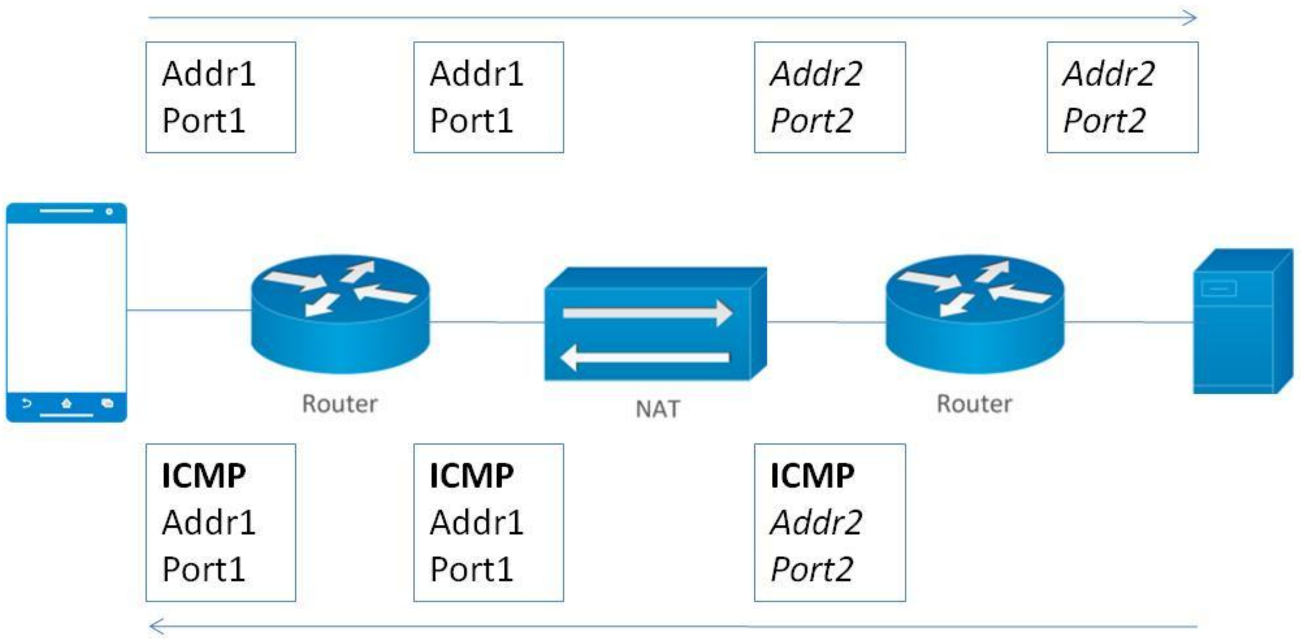
Mobile Tracebox

- Architecture



NAT detection

- Server-based mode succeeds



- Tracebox mode fails
  - Source Address and Source Port of ICMP embedded packet are restored by NAT to original values

NAT detection: RFC5508

Best current practice with respect to ICMP Error Packet Translation

When a NAT device receives an ICMP Error packet from the external realm, if the NAT has an active mapping for the embedded payload, then the NAT MUST do the following prior to forwarding the packet:

1. *Revert the IP and Transport headers of the embedded IP packet to their original form, using the matching mapping;*
2. *Leave the ICMP Error type and code unchanged;*
3. *Modify the destination IP address of the outer IP header to be same as the source IP address of the embedded packet after translation.*

NAT detection: RFC5508

Best current practice with respect to ICMP Error Packet Translation

When a NAT device receives an ICMP Error packet from the external realm, if the NAT has an active mapping for the embedded payload, then the NAT MUST do the following prior to forwarding the packet:

1. *Revert the IP and Transport headers of the embedded IP packet to their original form, using the matching mapping;*
2. *Leave the ICMP Error type and code unchanged;*
3. *Modify the destination IP address of the outer IP header to be same as the source IP address of the embedded packet after translation.*

OK, but what happens to the Checksum?

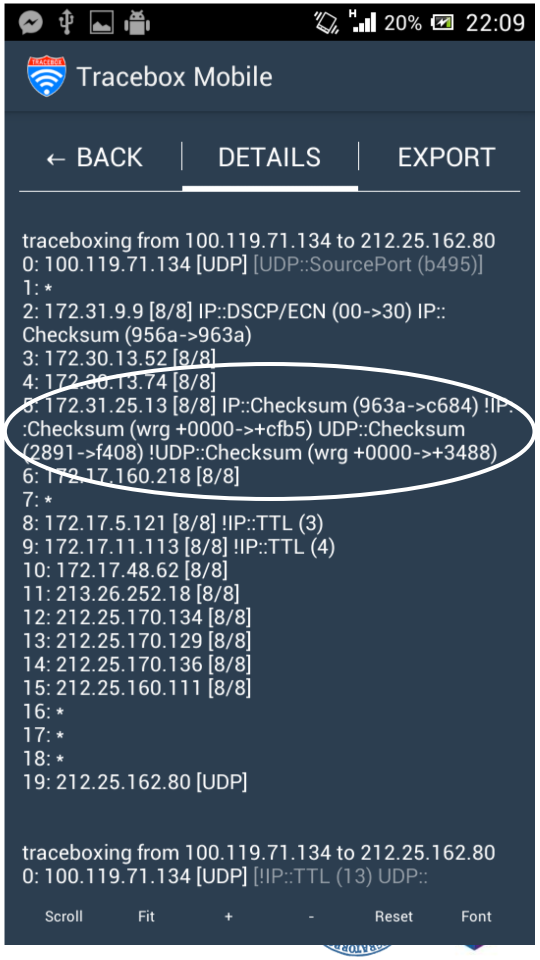
Uncovering NAT: Checksum errors

- A first survey of data collected
  - About 5% of routers crossed showed Layer-4 Checksum inconsistency
  - About 1% of routers crossed showed IP Checksum inconsistency

- Other clues
  - errors appear at some hop and persist after that hop until the destination
  - for repeated probes, errors always show up at the same hop (even with different values)
  - packets within a single connection show the same offset

    for instance TCP SYN and the following ACK packet, although having different sent and received transport checksums, turn out to have the same offset from the respective correct values
  - packets belonging to different connections show the same IP checksum offset

- All these observations are compatible with NATs.



```
traceboxing from 100.119.71.134 to 212.25.162.80
0: 100.119.71.134 [UDP] [UDP::SourcePort (b495)]
1: *
2: 172.31.9.9 [8/8] IP::DSCP/ECN (00->30) IP::
Checksum (956a->963a)
3: 172.30.13.52 [8/8]
4: 172.30.13.74 [8/8]
5: 172.31.25.13 [8/8] IP::Checksum (963a->c684) !IP:
:Checksum (wrg +0000->+cfb5) UDP::Checksum
(2891->f408) !UDP::Checksum (wrg +0000->+3488)
6: 172.17.160.218 [8/8]
7: *
8: 172.17.5.121 [8/8] !IP::TTL (3)
9: 172.17.11.113 [8/8] !IP::TTL (4)
10: 172.17.48.62 [8/8]
11: 213.26.252.18 [8/8]
12: 212.25.170.134 [8/8]
13: 212.25.170.129 [8/8]
14: 212.25.170.136 [8/8]
15: 212.25.160.111 [8/8]
16: *
17: *
18: *
19: 212.25.162.80 [UDP]


traceboxing from 100.119.71.134 to 212.25.162.80
0: 100.119.71.134 [UDP] [!IP::TTL (13) UDP::
```

Uncovering NAT: Checksum errors

```
0 100.119.71.134 [UDP] UDP::SourcePort (b495)
1 *
2 172.31.9.9 [8/8]       IP::DSCP/ECN (00->30) IP::Checksum (956a->963a)
3 172.30.13.52 [8/8]
4 172.30.13.74 [8/8]
5 172.31.25.13 [8/8]    IP::Checksum (963a->c684) !IP::Checksum (wrg +0000->+cfb5)
                        UDP::Checksum (2891->f408) !UDP::Checksum (wrg +0000->+3488)
6 172.17.160.218 [8/8]
7 *
8 172.17.5.121 [8/8]     !IP::TTL (3)
9 172.17.11.113 [8/8]    !IP::TTL (4)
10 172.17.48.62 [8/8]
11 213.26.252.18 [8/8]
12 212.25.170.134 [8/8]
13 212.25.170.129 [8/8]
14 212.25.170.136 [8/8]
15 212.25.160.111 [8/8]
16 212.25.162.80 [UDP]
```

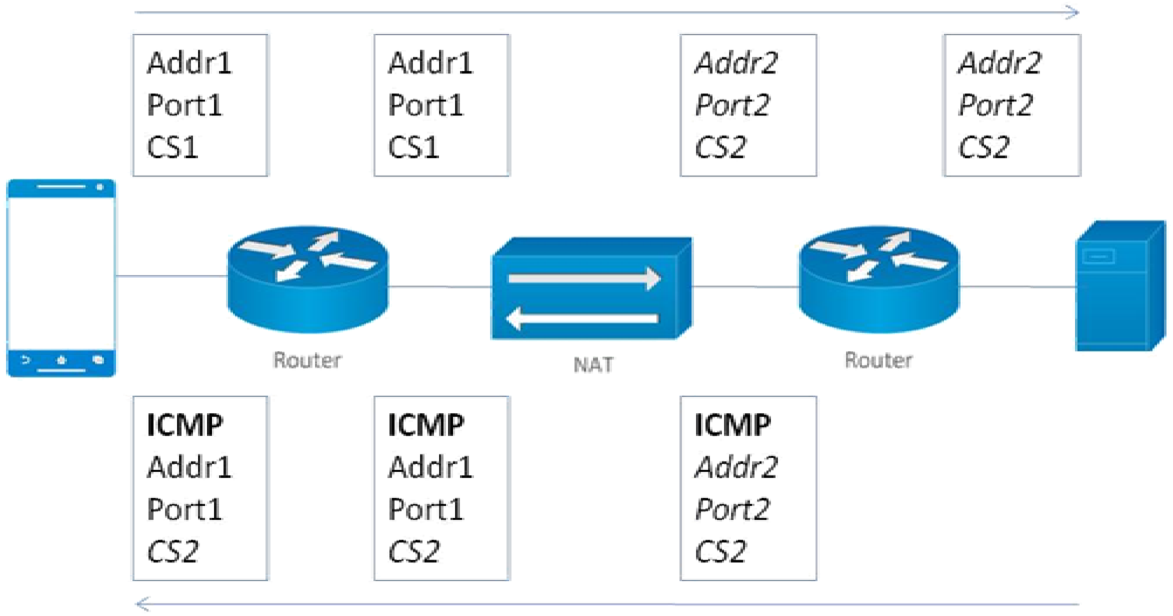
Uncovering NATs: Validation

- Tracebox + Server-based probes
  - A packet is sent to the controlled server in a traceroute fashion
- 22 networks (cellular and Wi-Fi) tested
- Checked for match between
  - IP and Transport Checksum errors detected through tracebox
  - Address and Port translation offset detected at the server

	Address+Port	Address only	None	All
IP	-	2	5	7
TCP	7	2	3	12
UDP	3	5	3	12

Uncovering NATs: Validation

- Checksum errors match address and port translation offset in most cases



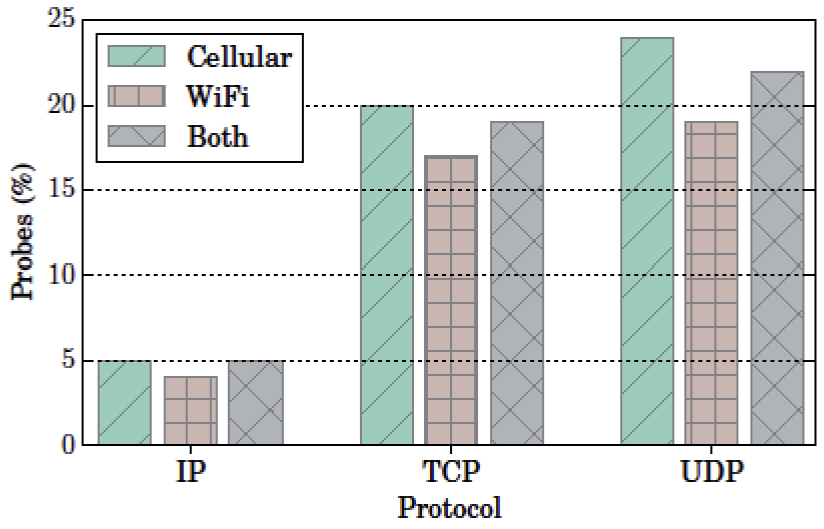
- Some Transport Checksum errors match addresses offset even if there's a NAPT (especially for UDP)
- NB. Errors not matching address translation offset

Dataset

- Probes performed by 124 users in tracebox mode
  - From February 2016 to February 2017
- Sanitization
  - Probes that presented errors or inconsistencies in the output
  - Probes where it was not possible to retrieve information about the network through Android APIs
  - Probes exhibiting a network switch (cellular to Wi-Fi or vice versa)
  - IP Checksum
    - errors matching the pattern 0x0*00 are excluded
    - quite frequent but are more probably due to a mismatch between checksum and TTL than related to NATs.
- Sanitized Dataset
  - 8,000 probes
  - 40 cellular networks
  - 72 Wi-Fi networks

Results

- Checksum errors raw statistics on IPv4 probes



- A significant portion of paths presents TCP or UDP checksum errors.
- IP checksum errors are less frequent
  - NATs are probably less negligent in manipulating IP header

Refining dataset

- Previous results can underestimate the extent of this methodology
  - Not all probes are supposed to show the requested behavior
- Dataset must be refined with respect to
  - **Address owned by the host**
    - Private address: NAT presence is implied
    - Public address: NAT presence is not likely
  - **Presence of middleboxes obstructing traceroute**
    - Proxies, Firewall blocking ICMP messages, etc
    - No or a few intermediate hops
  - **ICMP embedded packet quoted bytes**
    - Mobile Tracebox can't validate transport checksum
- We restrict dataset to:
  - probes with at least a public address router with regard to IP checksum
  - probes with at least a public address router quoting the full original packet with regard to transport checksum

Refining dataset

```
1.1.1.1 [TCP SYN]
*
*
*
*
```

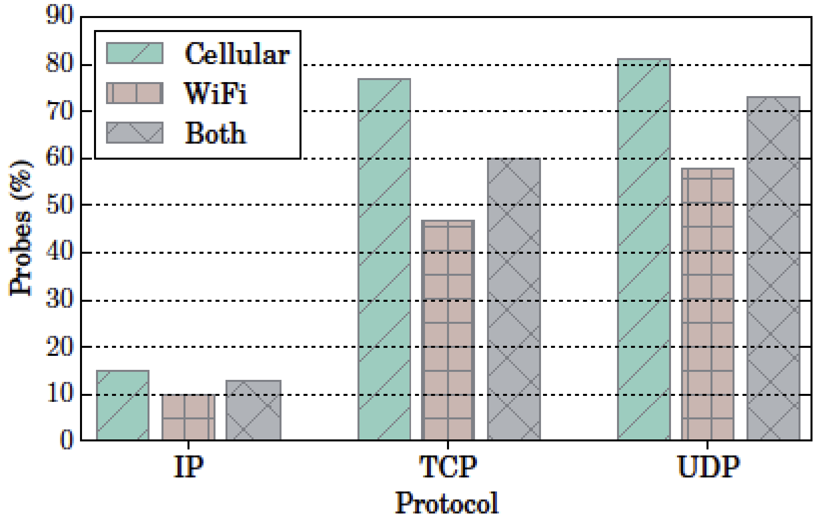
```
1.1.1.1 [TCP SYN]
2.2.2.2 [8/40]
*
4.4.4.4 [8/40]
5.5.5.5 [TCP SYN ACK]
```

```
1.1.1.1 [TCP SYN]
*
5.5.5.5 [TCP SYN ACK]
```

```
1.1.1.1 [TCP SYN]
2.2.2.2 [40/40]
*
4.4.4.4 [40/40]
5.5.5.5 [TCP SYN ACK]
```

Refined results

- Checksum errors refined statistics on IPv4 probes



- Under the circumstances discussed before, the majority of probes that are supposed to cross a NAT in traceroute path show checksum inconsistencies

Other uses

- **Double NATs:** mismatch between IP and transport checksum errors detected along the path and address and port translation offset detected at tracebox server can help to corroborate the suspicion of two of more NATs
- **Revealing NATted port:** when transport checksum error matches Address+Port offset, NATted port on outgoing connections can be revealed by combining private port, transport checksum error and offset between private address and NATted address (as discovered through tracebox server or similarly through a STUN server)
- **NAT properties:** in the same scenario as before, transport checksum errors on different connections can help to outline some NAT properties, e.g., if it is symmetric or not
- **Other methodologies:** our technique can be integrated with other methodologies to improve NAT detection

Conclusions and future work

- A novel methodology to detect NAT through traceroute
  - Based on IP or Transport Checksum errors found in ICMP Time-Exceeded embedded packet
  - Implemented in **Mobile Tracebox**
  - Deployed through crowdsourcing on 40 cellular and 72 Wi-Fi networks

When no other middleboxes are obstructing traceroute, it can detect NATs in the majority of cases

- A larger scale deployment – including wired networks – should outline more clearly the extent and possible usage of the methodology

Thank you

Mobile Tracebox

https://play.google.com/store/apps/details?id=be.ac.ulg.mobiletracebox

Questions, comments, etc

Raffaele Zullo

<r.zullo@studenti.unina.it>