Hic Sunt NATs: Uncovering Address Translation with a Smart Traceroute

Raffaele Zullo[†], Antonio Pescapé[†], Korian Edeline[‡], Benoit Donnet[‡]
[†]Università di Napoli Federico II, Italy - r.zullo@studenti.unina.it, pescape@unina.it
[‡]Montefiore Institute, Université de Liège, Belgium - firstname.name@ulg.ac.be

***Abstract*—Middleboxes are pervasive in today's Internet as they are deployed for an increasing number of reasons. An example is the network address translation (NAT), one of the first task to be performed to cope with the lack of IPv4 addresses. Recently the landscape for NATs has become even more crowded, especially in mobile networks, mainly due to the impossibility of IPv6 to be a large-scale solution to addressing issues.**

In this paper, we present a novel methodology for detecting NATs embodied in Mobile Tracebox, a measurement tool for Android smart devices that detects a wide range of middleboxes. It analyzes ICMP `time-exceeded` messages received during `traceroute` and points at IP and transport checksum inconsistencies in the embedded packets to uncover address translation along a path. We deployed Mobile Tracebox through a crowdsourcing approach and used the collected dataset to validate our methodology. Results showed that, in absence of middleboxes breaking `traceroute`, it can help to detect and locate NATs in the majority of the cases.

I. Introduction

The infrastructure use in the Internet's early days has radically evolved with the *end-to-end* principle making way to the presence of a variety of middleboxes along a path. Middlebox is a self-explanatory word for "an intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and a destination host" [1]. The term was coined in 1999 by Lixia Zhang but the concept of middlebox itself is even older: first examples of Network Address Translation (NAT) date back to 1994 [2] while first packet filters to late 1980s. The last years have witnessed an increase in middlebox usage [3], remarkably in mobile networks [4], [5], embracing a broadening range of functions (dealing with limited resources, improving performance, securing hosts, etc.). Unfortunately all these tasks come at the cost of a brake in protocols evolvability. Moreover, in some circumstances, middleboxes impoverish performance and alter end-to-end communications.

IP Address (and transport port) translation performed by NAT (NAPT) is among the first applications of middleboxes due to progressive IPv4 addresses depletion [6]. IPv6 should be the definitive solution but its adoption is still in its infancy [7]. In response to these delays, the picture of NATs deployment has gotten even more complicated with the possibility of multiple address translation at different stages (home network, provider's network) and all the complications that a multi-level sharing of an IP address can carry [8].

Recent papers have shed light on middleboxes, and notably NAT, deployment and their consequences. D'Acunto et al. [9] analyzed P2P applications and found that 88% of the participants in the studied P2P network were behind NATs. Sherry et al. [3] obtained configurations from 57 enterprise networks and revealed that they can contain as many middleboxes as routers. Wang et al. [4] surveyed 107 cellular networks and found that 82 of them used NATs. In parallel to the rise of middleboxes, a new practice for collecting network data appeared: *crowdsourcing* [10], [11]. Although it is a relatively new term [12], it refers to an approach already used in the past [13] and as "collaboration model enabled by people-centric web technologies" [14] it owes its success in the last years to the widespread availability of mobile devices and wireless Internet. Mobile-crowdsourcing approach where "mobile devices are used for data-collection tasks delegated to a larger number of people" [15] responds also to the necessity of the Internet measuring itself [16], [17]: probing networks from the user's point of view can lead to more effective measurements than running tests from fixed observation points. Beyond affecting performance, security and protocols evolvability, middleboxes also leave traces that allow us to reveal them. Unfortunately, although NATs are among the boxes most affecting end-to-end communications, they are usually transparent to network measurements: while using a controlled server can easily detect the presence of a NAT, measurement techniques based on `traceroute` and, more generally, techniques that require control on one end of the path fail to detect NATs.

In this paper, we present a novel methodology to deal with this limitation based on the analysis of ICMP `time-exceeded` messages generated by routers during `traceroute`, looking for inconsistencies in IP and transport checksums of embedded packets. We implemented this methodology in *Mobile Tracebox* [18], a measurement tool for Android smart devices belonging to the `tracebox` family [19], along with other techniques and procedures to detect different classes of middleboxes and cope with the innate limitations of mobile devices. We propose, validate, and evaluate Mobile Tracebox specifically with respect to NATs detection conducted through smart `traceroute` probes.

The remainder of this paper is organized as follows: Sec. II describes how Mobile Tracebox works; Sec. III introduces our methodology for detecting NATs; Sec. IV analyzes crowdsourced dataset to outline the extent of the methodology; Sec. V positions Mobile Tracebox regarding the state of the art; finally, Sec. VI concludes and discusses further work.

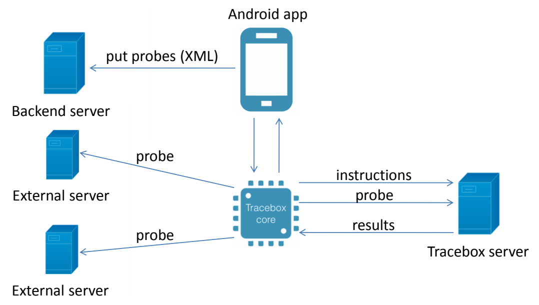

Fig. 1. General overview of the Mobile Tracebox architecture.

II. Mobile Tracebox

Methods to detect middleboxes are diverse and usually involve active measurements. Detection through passive measurement is also possible [20] but, as middleboxes are usually designed to be transparent, methods often resort to active probing to force the box to reveal its presence. Mobile Tracebox relies on an active measurement paradigm by sending specially crafted packets and retrieving them after modifications occur. There are two main methods implemented in our tool to highlight middleboxes along a path: the first is based on the `traceroute` mechanism (i.e., `tracebox` [19]), demanding control of a single endpoint (the mobile device, indeed – see Sec. II-A and Sec. II-B), the second method requires the presence of a a controlled server that cooperates with the device (Sec. II-C). The general overview of Mobile Tracebox is illustrated in Fig. 1. Both methods infer middleboxes from modifications explicitly observed on the packets. The main difference between the two methods is how the potentially altered copy of the sent packet is captured: *à la* `traceroute` method relies on ICMP `time-exceeded` messages quoting the expired packet, while the server-based method takes advantage of a controlled server to retrieve the packet exactly as it is received. Both methods have their pros and cons. The first method can detect and locate middleboxes but is prone to middleboxes breaking `tracebox` (e.g., proxies or firewalls affecting ICMP traffic). On the contrary, the server-based method embraces TCP-terminating proxies but fails at locating and can highlight only the last modification on every field, if multiple modifications occur. The latter method is also limited to specific paths as the two ends must be under control, while `tracebox` can be performed to any destination. Our work relies on `tracebox` to detect NATs and make use of server-based probes for validation.

A. Standard `tracebox`

The first method is perfectly embodied by `tracebox` [19], an extension to the widely used `traceroute`. `tracebox` borrows `traceroute` incremental approach, sending packets with increasing TTL and receiving ICMP `time-exceeded` replies from routers. While `traceroute` just takes note of the sent packet's TTL and the source address of ICMP reply to draw a path in terms of hops from source to destination, `tracebox` goes deeper extracting the original packet data inside the ICMP message and comparing it to the packet that

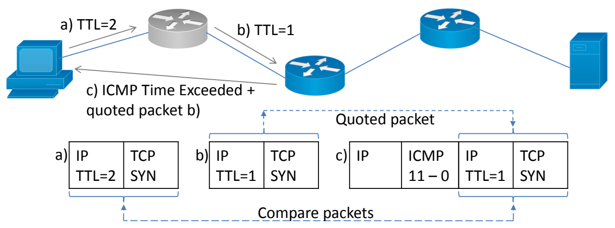

Fig. 2. Middlebox detection and localization with `tracebox`.

has been sent. This lets tracebox highlight which modifications were possibly performed on the packet (*middleboxes detection*) and between which hops they took place (*middlebox location*). This technique is made possible by RFC792 [21] and RFC1812 [22] stating that ICMP `time-exceeded` replies should encapsulate the full IP header plus respectively the first 8 bytes of IP payload or the full IP payload of the packet whose TTL has expired in transit.

In Fig. 2 between the source performing `tracebox` and the second hop lies a middlebox altering TCP Initial Sequence Number (ISN): it receives the original packet sent (`a`) and forwards the altered packet (`b`) to the next hop; as the TTL expires an ICMP message (`c`) is returned containing `b` as payload. By comparing `b` to `a`, `tracebox` will show a TCP Sequence Number modification observed at hop 2 allowing the user to detect and locate the middlebox.

B. Smart Devices

A proof of concept version of `tracebox` for mobile devices has been developed by Thirion et al. [23] as *TraceboxAndroid*. It included a porting of classic `tracebox` mode to Android (core) in conjunction with a GUI for users to perform and review probes (frontend) and a backend server to collect data. In the new version, more pretentiously named *Mobile Tracebox*, we tried to deal with the limitations of the first version: we extended the support to all Android platforms (arm, x86, mips, both 32 and 64 bit architectures), enhanced the core to handle all possible TCP and UDP probes over IPv4 or IPv6 and provided full customization capabilities for every single field or parameter of the probe. With server-based mode we also extended operations to non-rooted devices but, as this methodology is based on `traceroute` and access to content of ICMP `time-exceeded` messages requires `CAP_NET_RAW` POSIX capability, the audience of this work is still restricted to rooted Android devices. Core has undergone the most of the changes as it is the key section of the app. Further, we opted for a more functional and user-friendly frontend, and extended backend to match new requirements and to support a preliminary screening of data as soon as they were collected. We completed the architecture with a `tracebox` server (further detailed below) as shown in Fig. 1 to include server-based probing and to cope with lack of privileges of most mobile devices. Probes are sent in two ways: (*i*) the user selects a destination and runs the probe (*instant probing*) and (*ii*) a scheduled task executed in background is responsible for probing the network (*background probing*), if the user has enabled this feature.

We adopted some further refinements with respect to standard `tracebox`. Our core keeps record of the modifications observed: if a certain modification is detected at hop i it will be listed at a hop $> i$ only if the field changes again from last value observed. This output is more friendly for the user allowing him to better locate where modifications actually occur. Obviously this carries the assumption that packets with increasing TTL are flowing through the same path, or at least are undergoing the same modifications along different paths (including load balancing), which is not unreasonable.

A special treatment is given to IPv4 TTL (as well as to IPv6 Hop Limit) and Checksum: modifications of those fields that are entirely in line with routing (i.e., decreasing TTL by one and updating IPv4 Checksum accordingly) are not shown in Mobile Tracebox since they are not related to middleboxes. This allows our tool to better highlight subtle boxes interference, e.g., expired packets with TTL higher than 1 (even 4, 5) or packets with IP Checksum altered not accordingly to other modifications, that should have passed unnoticed in a sea of other legitimate TTL/Checksum modifications. Another special treatment is given to TCP and UDP Checksums along with already quoted IPv4 Checksum. Mobile Tracebox doubles the markers for Checksum fields: one is just the value of the field, the other is the value returned after checking the integrity of the packet (it is supposed to be zero). In the remainder of this paper, we will see that it is not infrequent to receive ICMP messages carrying quoted packets with wrong IP, TCP, UDP Checksum and we will give the offset from correct Checksum a specific meaning.

C. Server-Based `tracebox`

Not all middlebox interferences are brought to light by the `tracebox` method. As well as modifications can occur on outbound packet, there is nothing to prevent other modifications on the incoming packet – even if carried inside of ICMP enclosure. Boxes that change some fields on the way out can restore – in whole or in part – those fields to the original values making themselves completely transparent to standard `tracebox` test. A server-based approach will not fail in this scenario since controlling the server gives us access to the packet in the exact shape as it is received at the destination. Mobile Tracebox includes a server-based mode and can also combine the two modes: the packet is sent to the server in a `traceroute` fashion in order to show how middleboxes interference appears from both perspectives. We will take advantage of this mode in Sec. III-A.

III. Detecting NATs

Standard `tracebox` methodology is able to detect a wide spectrum of modifications performed by middleboxes. Unfortunately modifications on source address and port – indeed those performed by NAT – are not among them. The reason is given in RFC 5508 [24], stating NAT 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 the embedded payload, RFC5508 prescribes the NAT to 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.

Point 3 is needed for the host whose packet has expired in transit to receive the ICMP error message while Point 1 is needed for the same host to understand which connection (in other words which socket) the error is related to. This means that the NATted address and port of the quoted packet – inside the ICMP message – are restored to the original values making the NAT transparent. To leave no stone unturned, we used at first an experimental version of Mobile Tracebox where the core was set to receive any incoming ICMP `time-exceeded` message. In this way, `tracebox` was able to detect any modification on packets, including source address and port if not properly restored by NAT but, unfortunately, it was not of any help, resulting only in a large amount of inconclusive probes.

After this preliminary survey, we set the `tracebox` core to correctly retrieve only ICMP packets where the embedded packet matches sent packet 5-tuple (source address, destination address, source port, destination port, transport protocol), and we switched to a smarter methodology. Middleboxes changing a specific 3rd or 4th layer field do not operate solely on that field. For instance, a box modifying IP DSCP has to update IP checksum since the IP header has been changed. NATs, besides source address and port, have to update both IP and TCP/UDP checksum: even if the NAT does not alter port, transport checksum has to be updated since IP source address is part of the *pseudoheader*. We have explained earlier how NATs restore the initial values of source address and port, the next question is: are they restoring also IP and transport checksum? Looking back to RFC5508 [24], it states that NAT should "revert the IP and transport headers of the embedded IP packet to their original form", thus including checksums even if not explicitly asserted. On the other hand, a first deployment of Mobile Tracebox showed evidence of ICMP messages carrying an embedded packet with wrong IP or transport checksum, especially the latter. There is a good reason for tolerating wrong transport checksum inside ICMPv4 messages and it is related to quoting mechanism: although RFC1812 [22] recommends to include as much as possible of the original packet, the previous RFC792 [21] recommended only 8 bytes of Layer-4 data to be included and, in most cases, only a part of the transport data is reported: in this cases the checksum is, of necessity, inconsistent. RFC5508 suggests incorrect IP and transport checksums to be treated differently: IP checksum should be validated and a packet with wrong IP checksum should be dumped while transport checksum should not be validated. While analyzing data as

TABLE I
CHECKSUM ERROR COHERENCE WITH ADDRESS/PORT OFFSET.

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

they were collected from users (see Sec. IV-B for details about data collection), we realized that about 5% of routers crossed by `tracebox` showed checksum inconsistencies in Layer-4 checksum and about 1% in IP checksum. The reasons behind these errors can be diverse: from malfunctioning in forwarding packets to any mix of modifications occurring on downstream and upstream that do not compute the checksum accordingly. Mobile Tracebox abstains from evaluating transport checksum when the entire packet is not available, thus partial quoting can be excluded as a cause for detected checksum inconsistencies. Inspecting those errors, we saw evidence that they can be related to NATs:

- 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.

A. Validation

To further validate correlation between checksum errors and NATs, we resort to `tracebox` and server-based probes. We analyzed probes from 22 networks, both cellular and Wi-Fi, and checked for coherence between IP and transport checksum errors detected through `tracebox` and modifications on source address and port detected at the server.

Results presented in Table I show how checksum errors can be linked to NATs. In these cases, the error detected via `tracebox` matches exactly the address (and port) translation offset. Surprisingly, a few transport checksum errors (especially for UDP) match only the offset between private address and NATted address, even when the probe to our server confirmed the presence of a NAPT. IP checksum errors can only match addresses offset because IP checksum does not include port information. In Fig. 3 a typical scenario where our methodology works is shown. The NAT is translating a client's private address and port and updating the transport checksum on the client's outgoing packet. The packet is received at the server with the fields `Addr2`, `Port2`, `Checksum2`. When an incoming ICMP `time-exceeded` message reaches the NAT, it restores `Addr1` and `Port1` in the embedded packet but does not update the checksum: the packet retrieved through

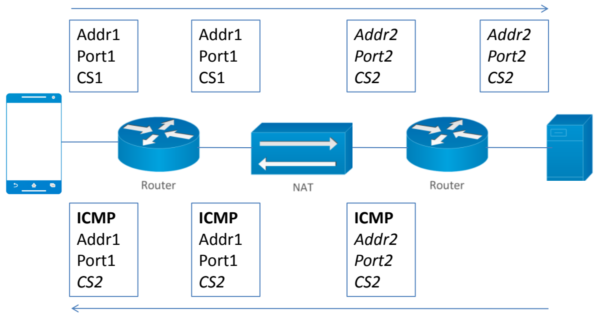

Fig. 3. Detecting NATs through checksum errors.

`traceroute` has the fields `Addr1`, `Port1`, `Checksum2`. The client then computes the offset between `Checksum1` and `Checksum2`.

Errors that do not match NAT offsets can be due to other middeboxes modifications that are performed by upstream and restored downstream (without restoring checksum) but we did not find evidence of correlation with other fields alterations detected at the server. They can be still related to NAT in a more subtle way and must be further investigated. Two networks showed at the same hop a transport checksum error matching the offset and an IP checksum error not matching the offset: we cannot exclude that even the IP checksum error is due to NAT. We must clarify that the test above should not succeed when there is more than one NAT and only one is not updating checksums. In this case, the error is related to one NAT while the offset is due to multiple address translations. We had physical access to one of the networks that showed no coherence between checksum error and address/port offset and verified that it had a double level of NAT.

IV. DEPLOYMENT

A. Dataset

In Sec. III we used a small number of networks to show that checksum errors in ICMP embedded packet can be linked to NATs. Now we analyze the dataset collected through crowdsourcing to outline the extent of this methodology. We started from all probes executed in `traceroute` mode by 124 users and collected from February 2016 to February 2017 and performed a sanitization. We eliminated 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) during the execution. With regard to IP checksum, we excluded all errors matching the pattern 0x0*00: these turned out to be quite frequent but are more probably due to a mismatch between checksum and TTL than related to NATs. The sanitized dataset includes about 8,000 probes belonging to 40 cellular networks and 72 Wi-Fi networks.

B. Results

We show now results of the analysis of IPv4 probes. We will analyze IPv6 probes in the last part of this section.

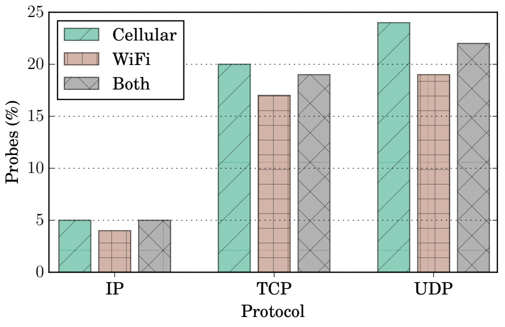

Fig. 4. Checkum errors raw percent on probes.

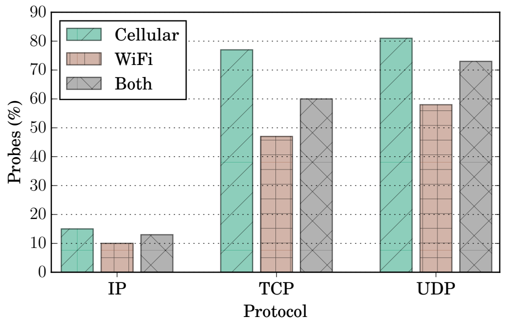

Fig. 5. Checkum errors refined percent on probes.

Fig. 4 displays how many probes are showing IP and transport checksum errors: a significant portion of paths presents TCP or UDP checksum errors. IP checksums errors are less frequent as NATs are probably less negligent in manipulating IP header of embedded packets.

Results in Fig. 4 can underestimate the extent of this methodology as they are based on the entirety of probes: not all probes in fact are supposed to exhibit that previous behavior. To gain a better perspective we have to refine the dataset with respect to three features: the address owned by the host, the presence of other middleboxes that break `traceroute`, and the partial embedding of Layer-4 data in ICMP error messages. First of all, we have to differentiate probes in which a host has a private address and probes in which the host has a public address: while in the first case the presence of NAT is implied, in the second instead it is quite unlikely. Then, we have to exclude the probes where other middleboxes break `traceroute`. Due to the presence of a proxy or a firewall blocking ICMP messages many probes exhibits no or just a few intermediate hops: as we linked errors to NAT, a probe should show in the path at least one router with a public address, thus residing outside the private network. Another consideration concerns the amount of Layer-4 data encapsulated in ICMP messages. Although RFC1812 recommends to quote the entire packets, many IPv4 routers quote only the first 8 bytes of transport layer, as originally required by RFC792: Mobile Tracebox cannot validate a Layer-4 checksum if the packet is not available in its entirety. This leads us to restrict the eligible probes 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. Fig. 5 displays the refined statistics for checksum errors. It shows how, under the circumstances discussed before, the majority of probes that are supposed to cross a NAT in `traceroute` path show checksum inconsistencies.

We excluded probes where the host has a public IP address from last statistics. We analyzed separately 320 probes collected from 8 networks but did not find any evidence of checksum errors. This is not surprising: even if still possible, the presence of a NAT is quite unlikely in those cases.

All the statistics provided till now are based only on IPv4 probes. We explored also IPv6 probes in the dataset looking for checksum errors: 160 probes belonging to 5 networks did not show any inconsistency in transport checksum (IPv6 has no checksum at IP layer). Again it is entirely in line with the relation between checksum errors and address translation since a host owning an IPv6 public address is not supposed to cross a NAT.

We now briefly summarize some other possible uses of this methodology: (*i*) 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 corroborating the suspicion of two of more NATs; (*ii*) 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 [25] server); (*iii*) NAT properties: in the same scenario as before, transport checksum errors on different connections can help outlining some NAT properties, e.g., if it is symmetric or not; (*iv*) Other methodologies: our technique can be integrated with other methodologies to improve NAT detection.

V. Related Work

`traceroute`, despite its well-known limitations [26], has been used for revealing IP interfaces along the path between a source and a destination. It has been extended in a variety of ways, for instance to face load balancing [27], the reverse path [28], or detect hidden routers [29] and middleboxes [19]. In the last years, middleboxes inference has become an important topic of interest. `TCPExposure`, developed by Honda et al. [30], is close to `tracebox`: it resorts to specially crafted packets to test for middlebox interference. Craven et al. [31] proposed `HICCUPS` to reveal packet header manipulation to both endpoints of a TCP connection: it works by hashing a packet header and by spreading the resulting hash into three fields. The `Netalyzr` [11] tool has provided a survey on several features of Internet's edge including interference by middleboxes. Considering NATs, Wang et al. [4] deployed `net-piculet` on 107 cellular networks and found that 82 of them used NATs. Müller et al. proposed `NATAnalyzer` [32], an algorithm to discover previously unknown cascaded NAT configurations by controlling both ends of the path. More recently, Lutu et al. introduced `NAT Revelio` [8] a novel test suite and methodology for detecting NAT deployments beyond the home gateway, also known as NAT444. All these tools provide great results but none of them provides a methodology to detect address translation along a `traceroute` path to any destination.

VI. Conclusions and Future Works

NAT detection along a path can be complicated when both ends are not under control as this kind of middlebox can be completely transparent to certain measurements, specifically those based on `traceroute`. We addressed this issue with a new methodology that detects address translation from checksum inconsistencies found in embedded packets inside ICMP `time-exceeded` messages received during `traceroute`. We implemented the methodology in Mobile Tracebox, a `tracebox` extension for Android devices. Based on data collected through crowdsourcing from 40 cellular networks and 72 Wi-Fi networks from all around the world we showed the extent of the new technique. Results demonstrate that, when no other middleboxes (e.g., proxies, firewall blocking ICMP, etc) are obstructing `traceroute`, it can detect and locate NATs in the majority of cases. Finally we have briefly summarized some other possible uses of this methodology.

A larger scale deployment, including also wired networks, would be of help to outline more clearly the extent and possible usage of this methodology.

Acknowledgments

The research described in this paper 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.

References

[1] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and issues," Internet Engineering Task Force, RFC 3234, February 2002.

[2] K. Egevang and P. Francis, "The IP network address translator (NAT)," Internet Engineering Task Force, RFC 1631, May 1994.

[3] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proc. ACM SIGCOMM*, August 2012.

[4] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *Proc. ACM SIGCOMM*, August 2011.

[5] A. Botta and A. Pescapé, "Monitoring and measuring wireless network performance in the presence of middleboxes," in *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*. IEEE, 2011, pp. 146–149.

[6] G. Huston, "IPv4 address report," April 2017, https://ipv4.potaroo.net.

[7] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey, "Measuring IPv6 adoption," in *Proc. ACM SIGCOMM*, August 2014.

[8] A. Lutu, M. Bagnulo, A. Dhamdhere, and k. claffy, "NAT revelio: Detecting NAT444 in the ISP," in *Proc. Passive and Active Measurement Conference (PAM)*, March 2016.

[9] L. D'Acunto, N. Chiluka, T. Vinò, and H. J. Sips, "Bittorrent-like P2P approaches for VoD: a comparative study," *Computer Networks (COMNET)*, vol. 57, no. 5, pp. 1253–1276, April 2013.

[10] A. Faggiani, E. Gregori, L. Lenzini, S. Mainardi, and A. Vecchio, "On the feasibility of measurement the Internet through smartphone-based crowdsourcing," in *Proc. IEEE International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks*, May 2012.

[11] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the edge network," in *Proc. ACM Internet Measurement Conference (IMC)*, 2010.

[12] J. Howe, "The rise of crowdsourcing," *Wired Magazine*, vol. 14, no. 06, pp. 2–6, June 2006.

[13] A. Tarrell, N. Tahmasbi, D. Kocsis, A. Tripathi, J. Pedersen, J. Xiong, O. Oh, and G.-J. de Vreede, "Crowdsourcing: A snapshot of published research," in *Proc. Conference on Information Systems*, August 2013.

[14] J. Pedersen, D. Kocsis, A. Tripathi, A. Tarrell, A. Weerakoon, N. Tahmasbi, J. Xiong, W. Deng, O. Oh, and G.-J. de Vreede, "Conceptual foundations of crowdsourcing: A review of IS research," in *Proc. 46th Hawaii International Conference on System Sciences*, January 2013.

[15] F. Fuchs-Kittowski and D. Faust, "Architecture of mobile crowdsourcing systems," in *Proc. 20th International Conference on Collaboration Technology (CRWIG)*, September 2014.

[16] Y. Shavitt and E. Shir, "DIMES: Let the internet measure itself," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 71–74, October 2005, see http://www.netdimes.org.

[17] M. Molinari, M.-R. Fida, M. K. Marina, and A. Pescape, "Spatial interpolation based cellular coverage prediction with crowdsourced measurements," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*. ACM, 2015, pp. 33–38.

[18] R. Zullo, K. Edeline, A. Pescapé, and B. Donnet, "Mobile tracebox," 2016. [Online]. Available: http://play.google.com/store/apps/details?id=be.ac.ulg.mobiletracebox

[19] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2013.

[20] U. Goel, M. Steiner, M. P. Wittie, M. Flack, and S. Ludin, "Detecting cellular middleboxes using passive measurement techniques," in *Proc. Passive and Active Measurement Conference (PAM)*, March 2016.

[21] J. Postel, "Internet control message protocol," Internet Engineering Task Force, RFC 792, September 1981.

[22] F. Baker, "Requirements for IP version," Internet Engineering Task Force, RFC 1812, June 1995.

[23] V. Thirion, K. Edeline, and B. Donnet, "Tracking middleboxes in the mobile world with traceboxandroid," in *Proc. 7th International Workshop on Traffic Monitoring and Analysis (TMA)*, April 2015.

[24] P. Srisuresh, B. Ford, S. Sivakumar, and S. Guhg, "NAT behavioral requirements for ICMP," Internet Engineering Task Force, RFC 5508, April 2009.

[25] P. Matthews, J. Rosenberg, D. Wing, and R. Mahy, "Session Traversal Utilities for NAT (STUN)," RFC 5389, Oct. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5389.txt

[26] P. Marchetta, V. Persico, A. Pescapé, and E. Katz-Bassett, "Don't trust traceroute (completely)," in *Proc. ACM CoNEXT Student Workshop*, December 2013.

[27] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2006.

[28] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson, "Reverse traceroute," in *Proc. USENIX Symposium on Networked Systems Design and Implementations (NSDI)*, June 2010.

[29] P. Marchetta and A. Pescapé, "DRAGO: Detecting, quantifying and locating hidden routers in traceroute ip paths," in *Proc. IEEE Conference on Computer Communications Workshops*, April 2013.

[30] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2011.

[31] R. Craven, R. Beverly, and M. Allman, "Middlebox-cooperative TCP for a non end-to-end Internet," in *Proc. ACM SIGCOMM*, August 2014.

[32] A. Müller, F. Wohlfart, and G. Carle, "Analysis and topology-based traversal of cascaded large scale NATs," in *Proc. Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox)*, December 2013.