Towards a Middlebox Policy Taxonomy: Path Impairments

Korian Edeline, Benoit Donnet
Université de Liège – Belgium

***Abstract*— Recent years have seen the rise of middleboxes, such as firewalls, NATs, proxies, or Deep Packet Inspectors. Those middleboxes play an important role in today's Internet, including enterprise networks and cellular networks. However, despite their huge success in modern network architecture, they have a negative impact on the Internet evolution as they can slow down the TCP protocol evolution and its extensions. Making available a summary of the potential middlebox network interferences is of the highest importance as it could allow researchers to confront their new transport protocol to potential issues caused by middleboxes. And, consequently, allowing again innovation in the Internet.**

This is exactly what we tackle in this paper. We propose a path impairment oriented middlebox taxonomy that aims at categorizing the initial purpose of a middlebox policy as well as its potential unexpected complications. Based on a measurement campaign on IPv4 and IPv6 networks, we confront our taxonomy to the real world. Our dataset is freely available.

I. Introduction

Nowadays, the standard and well-known description of the TCP/IP architecture (i.e., the end-to-end principle) is not anymore applicable in a wide range of network situations. Indeed, enterprise networks, WiFi hotspots, and cellular networks usually see the presence of *middleboxes* being part of the network architecture in addition to traditional network hardware [1]. A middlebox is a network device inspecting, filtering, or even modifying packets that traverse it. It performs actions on a packet that are different from standard functions of an IP router.

Recent papers have shed the light on the deployment of those middleboxes. For instance, Sherry et al. [1] obtained configurations from 57 enterprise networks and revealed that they can contain as many middleboxes as routers. Wang et al. [2] surveyed 107 cellular networks and found that 82 of them used NATs. D'Acunto et al. [3] analyzed P2P applications and found that 88% of the participants in the studied P2P network were behind NATs. Middleboxes may be deployed for several reasons, typically security (e.g., IDS, NATs, firewalls) and network performance (e.g., load balancer, WAN optimizer).

However, if there is a widespread usage of middleboxes, they come with important drawbacks. Indeed, it has been shown that middleboxes have a negative impact on the TCP protocol (and its extensions) evolution [4], [5]. Middleboxes may modify, filter, or drop packets that do not conform to expected behaviors. As a consequence, the Internet faces a kind of ossification due to the difficulties of proposing new transport protocols.

Researchers, when designing a new protocol, have thus to cope with a middlebox-full Internet. Each new mechanism has to be certified as middlebox-proof [5], [6]. For those researchers, a summary of the potential middlebox network interferences would be a valuable asset as they could easily confront their new protocol with potential issues caused by middleboxes.

This is exactly what we want to tackle here. In this paper, we propose a path-impairment oriented middlebox policy taxonomy, that aims at categorizing the initial purpose of a middlebox policy as well as its potential unexpected complications. We choose to classify middlebox *policies* rather than middleboxes themselves because the latter often combine multiple policies. Our taxonomy describes and catalogues policies under three different points of view: *Intended Function* (i.e., the policy purpose), *Action* (i.e., the fate of a packet crossing a middlebox implementing this policy), and *Complication* (i.e., the potential path connectivity deterioration).

Based on a `tracebox` [7] (a `traceroute` extension that is able to reveal the presence of middleboxes along a path) measurement campaign on IPv4 and IPv6 networks[1], we confront our taxonomy to the reality on the ground. We establish a dual-stack survey of middleboxes that implement rewrite, drop, and proxy policies that may harm the performance of regular traffic and affect protocol deployability. Then, we discuss our results in light of our taxonomy.

The remainder of this paper is organized as follows: Sec. II discusses several practical cases in which middleboxes are the source of problems; Sec. III presents our middlebox taxonomy; Sec. IV explains the measurement and data processing methodology we followed; Sec. V discusses the results we obtained, with respect to our middlebox taxonomy; Sec. VI positions this paper regarding the state of the art; finally, Sec. VII concludes this paper by summarizing its main achievements.

II. Understanding the Problems

Middleboxes are causing various malfunctions, especially with end-to-end protocols, by dropping packets that contain certain options or by stripping those options from TCP `SYN` segments to prevent them from being negotiated [4]. However, middlebox-related problems are not limited to this and can be much more subtle. In this section, we describe three different problems that middleboxes may cause.

[1]Our dataset is freely available. See `http://queen.run.montefiore.ulg.ac.be/~edeline/`

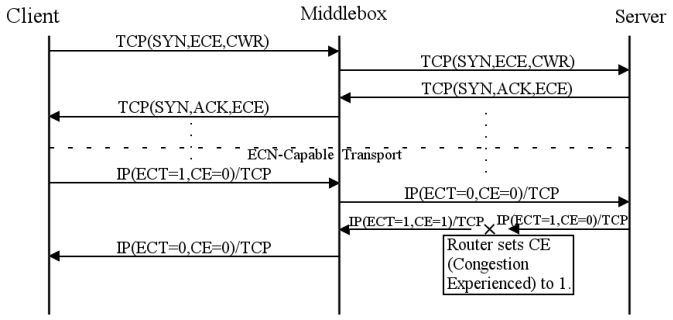


Fig. 1: Clearing IP ECN bits.

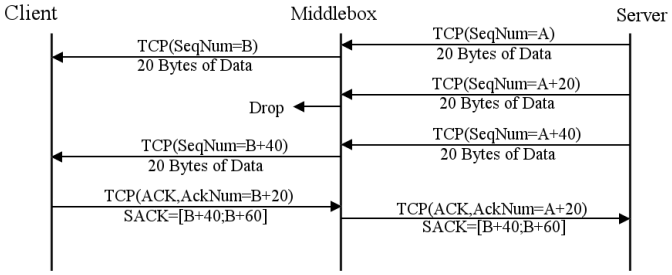


Fig. 2: TCP Initial Sequence Number re-shuffling middlebox and Selective ACKnowledgement.

A. Explicit Congestion Notification

In this setup, shown in Fig. 1, the middlebox is allowing both ends to negotiate the use of *explicit congestion notification* (ECN – a TCP/IP extension allowing to signal network congestion before packet losses occur) but is clearing the ECN bits in the IP header, rendering it unable to report any congestion.

The client tries to establish an ECN-Capable connection with a remote server by sending a TCP ECN-setup `SYN` segment (ECN Echo – `ECE` – and Congestion Window Reduced – `CWR` – flags set). The server sends back a TCP `SYN+ACK` segment with the `ECE` flag set. Both packets are forwarded unmodified by the middlebox. Following packets are marked as ECN-Capable Transport (`ECT`) by both ends but the middlebox is systematically clearing both the `ECT` and `CE` (Congestion Experienced) bits in the IP header. If an intermediate router sets the `CE` bit (Congestion Encountered) of a packet, it will be cleared by the middlebox afterward [8].

B. TCP Sequence Number

Fig. 2 illustrates a situation where a middlebox is applying modifications to TCP segments that leads them to be discarded by the server.

The server is sending three TCP segments with 20 bytes of data each through an already established TCP connection within which both ends agreed on the use of the `SACK` option. In the path between the client and the server, there is a middlebox rewriting the Sequence Number and ACKnowledgement Number of any packet of this connection as it has re-shuffled the Initial Sequence Number to counter prediction attacks. When the second packet with Sequence Number $A + 20$ is dropped, the receiving side is notifying the sending side by acknowledging the first and the third data segment using the `ACK` number and the `SACK` option. The middlebox is modifying the sequence and `ACK` number of this packet, but not the sequence numbers of the `SACK` block. If those unmodified sequence numbers are out of window, Linux's TCP stack discards the whole packet [5].

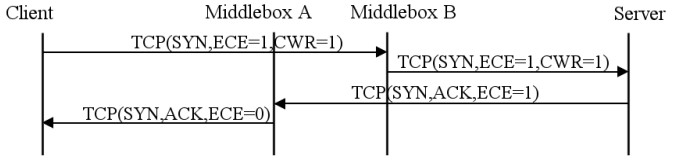


Fig. 3: State announcement and asymmetric paths/load balancing.

C. State Announcement

In this example, shown in Fig. 3, a middlebox tries to conceal attempts of state announcement, but with the presence of asymmetric paths, it leads to inconsistencies.

Both ends are trying to share state related data (i.e., `ECE` bit). The path between them is asymmetrical and ingress and egress traffic are crossing different middleboxes, A and B respectively. Middlebox A is clearing the `ECE` bit and middlebox B is not. The result of the state share is inconsistent because the server has sent `ECE` = 1 and the client has received `ECE` = 0. The client thinks that the connection is not ECN-Capable while the server does. Solutions to fix this issue (i.e., a fallback mechanism) have been proposed [9], but not widely deployed [10].

In the next section, we present a characterization guideline to examine those problems under different points of view and we extract root causes that we believe are common to all middlebox-related path impairments.

III. Taxonomy

In this section, we propose a path-impairment oriented middlebox policy taxonomy, that aims at categorizing the initial purpose of a middlebox policy as well as its potential unexpected complications.

We chose to classify middlebox *policies* rather than middleboxes themselves because the latter often combine multiple policies. Our taxonomy focuses on packet mangling middleboxes aspects, the initial purpose of such an action, and the network interferences that may result.

We describe each policy implemented in a single (not multi-hop) middlebox in three ways (i.e., meta-categories), each one including several taxa (i.e., categories); (i) **Intended Function**, *what* the policy expects to achieve, its purpose; (ii) **Action**, *how* the policy is trying to achieve its goals, the fate of a packet crossing a middlebox that implements this policy; (iii) **Complication**, the possible resulting path connectivity *deterioration*. Fig. 4 illustrates our path-impairment oriented middleboxes policy taxonomy. Each middlebox policy has to fall in *at least* one taxon for each of the three point of view in order to be consistent with the taxonomy.

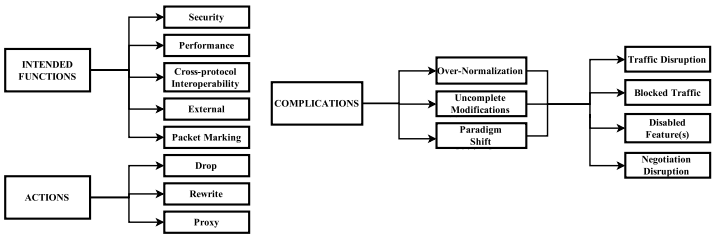


Fig. 4: A path-impairment oriented middlebox policy taxonomy.

As we aim at characterizing middlebox-related network interferences rather than establishing an exhaustive middlebox taxonomy [11], we focus on single-hop modifications and ignore other aspects.

While examining the possible complications involved by middlebox policies, we deliberately narrow our horizon to performance worsening and feature's inability of use, omitting the multiple and various reported security flaws created by middlebox policy implementations (see, for instance, Qian and Mao [12]).

Fig. 4 displays the taxonomy and shows its three facets: *Intended Functions*, *Actions*, and *Complications*. These meta-categories are described in the subsequent sections. We apply this classification to the dataset we collected in Sec. V. This taxonomy is distinct from the middlebox taxonomy presented in RFC3234 [11] as we aim at empirically classifying middlebox policies to regroup causes of performance worsening and to take common design decision afterwards, as an up-to-date path-impairment oriented extension of RFC3234.

A. Intended Functions

The *Intended Function* meta-category describes the purpose of a middlebox policy (e.g., what it is intending to achieve). As we already noticed earlier in this section, a policy can be classified differently if we consider the policy in itself or the middlebox set it is part of (e.g., a tunnel endpoint with respect to the whole tunnel). We consider five kinds of Intended Functions: *Security*, *Performance*, *Cross-Protocol Interoperability*, *External*, and *Packet Marking*.

Security-motivated middlebox policies are implemented in Security-oriented middleboxes, dedicated hardware deployed in enterprise, and home network for improving network security such as firewalls and IDS/IPSes. They may serve purposes like providing an authentication mechanism (e.g., application-level gateways), defending against attacks such as DDoS, attacks on protocols (e.g., IP, TCP), providing access control, normalizing the TCP features to prevent the use of features considered unknown and/or unsafe, or separating similar networks into different security zones.

It has been shown that such a type of middleboxes is becoming more and more popular [1]. This increasing popularity raises concerns regarding overly restrictive middlebox policies which may either block benign traffic matched as unsafe, causing blackholes, or apply modifications to transport and network header fields, precluding the use of protocol features or hampering their proper functioning, indirectly reducing network performance and/or functionalities (e.g., forbidding ECN or MultiPath TCP [13]).

Performance-enhancing middlebox policies aim at improving the network performance regarding a link along the path, a connection between both path's ends, or the middlebox itself (e.g., TCP performance enhancing proxies, caches, schedulers). It may aim at improving pure effective bandwidth performance through transport layer engineering as well as solving box-relative performance issues.

This category of middlebox policies may apply semantically wrong legacy modifications which may cause traffic disruption and protocol inconsistencies. They might as well introduce TCP errors and unforeseen changes in TCP adaptive behavior, leading to various network interferences.

Cross-protocol interoperability middlebox policies are performing protocol translation. They aim either at connecting dissimilar networks (e.g., NAT 6to4) or at translating protocols over the network layer (e.g., tunnel endpoints, transcoders). The main problem with this kind of policies is to be consistent with every protocol version and feature. Indeed, the input protocol language may evolve, and middleboxes may then apply outdated translations to it or may fail to ensure completeness of its modifications. This can lead either to the inability to use certain features or to network performance degradation.

Packet marking middleboxes classify packets according to policies and select or mark them for differentiated services (via IP DSCP's field). Their action is limited to network layer modifications. They are packet classifiers, or ECN-capable routers. Their policies are not likely to degrade network performance.

External middlebox policies have purposes that are external to the paths to which they belong (e.g., IPv4 address exhaustion) or not related to Internet technical aspects (e.g., economic purposes). External middlebox policies are heterogeneous and have to be analyzed independently.

B. Actions

The *Action* meta-category describes the actual action of a middlebox on a matched packet, defined by middlebox policies, performed to achieve its intended functions. We consider three basic kinds of middlebox policy actions: *Drop*, *Rewrite*, and *Proxy*. This aspect is decisive because middlebox policies that are applying different actions will more likely cause different types of network dysfunctions.

Drop policies are common features whose purposes vary from security to performance optimization concerns. Depending on how both ends react to this type of failure, the outcome may also vary from minor traffic disruptions, such as bandwidth reduction or the inability to use specific TCP options, to the inability to establish a TCP connection.

Rewrite policies are also common among middleboxes. Their initial purposes are covering the entire Sec. V-A. As they break the TCP/IP end-to-end principles, they may cause various problems to protocol end-to-end functions which assume unmodified layers over the third.

Proxy policies middleboxes are relay agents between clients and servers of an application. They vary from the *Rewrite* policies middleboxes by not simply rewriting the forwarded packets but rather by receiving client data from a connection, then establishing a second connection to send data to the server and vice versa.

C. Complications

We describe here the potential *Complications* caused by middleboxes policies by examining them under two points of view: (i), their technical causes, which are directly related to their initial purposes and, (ii), the associated actions (respectively Sec. V-A and Sec. V-B), and their unfortunate consequences, i.e., causes and consequences.

1) Causes: The *Causes* of the network interferences created by middleboxes aim at classifying the technical problems origin. It regroups manufacturers and policy designers fundamental errors or deliberated choices leading, from a path-impairment perspective, to network interferences.

Over-normalization refer to a middlebox policy that limits protocol features and options, as a blacklist or whitelist filter, to a restricted subset of the protocol. The problem of this type of middlebox behavior constraining the design of new extensions has already been addressed [4]. It may limit protocol performance as well by preventing the usage of the entire protocols capabilities, or simply by taking drop decisions.

The middlebox clearing IP ECN bits that is displayed in Fig. 1 falls within this category.

Incomplete modifications refers to middlebox policies that fail to ensure completeness of their modification(s). This type of network inconvenience is caused by middleboxes modifying a specific protocol field and not modifying semantically related fields, allowing translated/modified data alongside untranslated/unmodified data. They may fail to identify all related fields for legacy reasons or simply neglecting them for performance concerns (e.g., refusing to parse TCP options).

In Fig. 2, the middlebox translates TCP sequence numbers of the header but not those of the options, the modification being therefore incomplete.

A *Paradigm shift* (2-way to `n`-way) happens when both ends running a protocol are assuming 2-way peering relationships. Middleboxes, by applying modifications *in the middle of the path*, are breaking the TCP/IP end-to-end principles, and thus, are causing both ends to undergo a paradigm shift *de facto* to `n`-way peering relationships [14]. As many mechanisms are not designed to handle this new paradigm, errors may occur. When both ends are trying to share state related data or to negotiate capabilities, this phenomenon may, in certain scenarios, put both ends in conflicting states or, combined with an unfortunate load balancing, distort protocol negotiations [5].

An example of such inconsistencies is shown in Fig. 3. End state announcement is in fact path state announcement, which is incompatible with asymmetrical paths and/or load balancers.

2) Consequences: The *Consequences* are the network complications final outcome, what both ends actually experience.

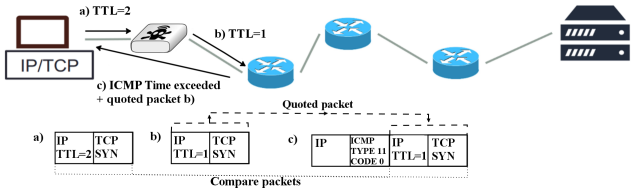


Fig. 5: Middlebox detection and localization with `tracebox`.

We focus exclusively on path performance related issues, leaving aside security and processing performance considerations.

Traffic disruption policies produce unwanted consequences such as interferences with control data rendering it useless, bandwidth reductions or others path performance impairments.

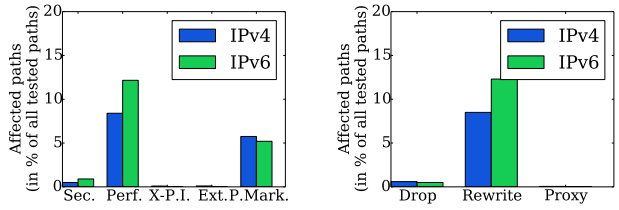
Middlebox policies may cause *Blocked traffic* either explicitly (sending TCP `RST` packet) or implicitly (dropping packets). It may not be the final outcome of a connection. If a specific option/feature is blocked by a middlebox, the client could be configured to retry establishing the connection without the undesirable options/features, but if it is not, no connection at all is possible.

Middlebox policies may aim at preventing the use of features considered unknown and/or unsafe by modifying, stripping them or by preventing them from being negotiated. If it is achieve symmetrically, the consequences are limited to the inability to use the restricted features; It is a *Feature-disabling* policy. If the modifications are done asymmetrically and the negotiation is not resilient enough, the policy may fail to disable the feature and lead to inconsistent protocol states [5]. Policies resulting in the latter consequences are categorized as *Negotiation disruption* policies.

IV. Measurement Methodology

To reveal the presence of middleboxes along a path, we use `tracebox` [7], an extension to the widely used `traceroute`. `tracebox` mechanism is displayed in Fig. 5. It relies on RFC1812 and RFC792 stating that the returned ICMP `time-exceeded` message should quote the IP header of the original packet and respectively the complete payload or the first 64 bits. `tracebox` uses the same incremental approach as `traceroute`, it sends packets with different IP, UDP or TCP fields and options with increasing TTL values. By comparing the quoted packet to the original, one can highlight the modifications and the initial TTL value allows us to localize the two or more hops between which the change took place. Readers interested in details and extended evaluation of `tracebox` should refer to Detal et al. [7].

We observed the following methodology: We started from the top one million web sites from Alexa. We performed A and AAAA DNS queries to obtain the corresponding IP addresses. Out of the whole set of domains, 94.7% had an A record and 4.5% had an AAAA record registered in the servers we probed. We kept only the dual stacked domains in order to conduct IPv4/IPv6 comparative analysis. The complete address set contained 14,373 different address pairs. Second, we deployed `tracebox` on CAIDA's active measurement infrastructure, Archipelago [15]. We made use of 20 monitors with IPv6 connectivity scattered over four continents: eight



(a) Policies per purposes (b) Policies per actions

Fig. 6: Middleboxes observed during our measurement campaign with respect to our taxonomy.

Options	*Drop (%)*	
Field:	IPv4	IPv6
ECN-setup SYN	0.37	0.35
MultiPathTCP	0.21	0.18

TABLE 1: Observed drop rates, in % of all tested paths.

monitors in Europe, seven in America, two in Asia, and three in Oceania. Third, we performed our measurements from each vantage points to each IPv4 and IPv6 address in parallel. Each monitor of the Archipelago infrastructure probed each IPv4 address and its corresponding IPv6 address (in the DNS sense) at the same time. Fourth, each sent and received packet during a `tracebox` run was saved and identified by a $(run\ \#, saddr, daddr, key)$-tuple where the key is used to match concurrent IPv4 and IPv6 probing runs. Finally, we analyzed the collected dataset to visualize how middleboxes apply their policies in the middle of Internet paths and cause traversal issues.

V. Results

The main results are provided in Fig. 6. It displays the percentage of routers that were observed to implement different types of policies. They are discussed in Sec. V-A and Sec. V-B. Table 1 contains the observed drop rates for a few TCP options. Table 2 shows the modification (*Rewrite*) rate of TCP options and IP ECN bits over IPv4 and IPv6. We localized the modification source using `tracebox` location ability and we used normalized distance (in number of hops) to distinguish three different locations; Close to the vantage point (from the source to hop 3 included), in the Network core (from hop 4 to hop 10 included) and close the to destination (after hop 10).

Note that the basic technique we used to detect middleboxes modifications (e.g., `tracebox`) is not limited in IPv6 by routers not quoting more than the first 64 bits of the transport layer within ICMPv6 time-exceeded; 99.9% of all IPv6 routers are quoting the entire packet, while 74% of all IPv4 paths involve at least one of those routers.

A. Intended Functions/Purposes

Fig. 6(a) displays the middlebox policies inferred purposes. TCP Initial Sequence Number re-shuffling middleboxes and unknown TCP options dropping policies are counted as security purposes. Policies rewriting MSS (Maximum Segment Size), WScale (Window Scale), and ECN (Explicit Congestion Notification) are considered as performance-related policies. Cross-protocol interoperability policies contains NATs policies and we counted IP DSCP rewriting as a packet marking purpose policy.

Options	*Modif. (%)*			
Field:	Close to VP	Network Core	Close to Dest.	Total
IP::ECN	3.5 / 0	0.90 / 10	0.70 / 1.2	5.10 / 11.2
TCP::ISN	ϵ / 0	0.03 / 0.04	0.02 / 0.25	0.07 / 0.31
MSS(1460)	3.0 / 0	0.60 / 0.04	0.09 / 0.25	3.29 / 0.94
MSS(1600)	3.5 / 0	0.80 / 0.04	0.11 / 0.29	4.51 / 0.33
WScale(14)	ϵ / ϵ	0.05 / 0.07	0.11 / 0.12	0.16 / 0.19
$\ominus$ MPTCP	0 / 0	0.04 / 0.1	0.09 / 0.22	0.13 / 0.32
NAT	-	0.01 / ϵ	0.01 / ϵ	0.02 / ϵ
IP::DSCP	0.15 / 0.7	4.50 / 3.5	1.10 / 1	5.75 / 5.2
IP::ID[2]	0 / 0	0.02 / 0	0.04 / 0	0.06 / 0
Proxies	-	-	-	0.02 / ϵ

TABLE 2: Observed modification rates, in % of all tested paths. IPv4 rate / IPv6 rate. $\ominus$ means option was removed, else it was modified. An ϵ rate means the frequency of the modification is negligible.

The amount of paths crossing a middlebox with security-related policies is higher over IPv6 than IPv4: 0.9% against 0.5%. According to Table 2, this difference is due to the TCP Initial Sequence Number re-shuffling middleboxes.

We also found a few middleboxes that stripped the `MPCapable` and `MPJoin` options (MultiPath TCP) from their TCP segments. The numbers, as displayed in Table 2, are quite low (0.21% of IPv4 and 0.18% of IPv6 paths) but still significant as it may end up in a client blackhole.

Among the performance-related middlebox policies, we highlight middleboxes that clear ECN bits. As shown in Table 2, we find that 5.1% of IPv4 and 11.2% IPv6 paths systematically cleared the ECN Congestion Experienced (`CE`) bit of the IP layer. A previous study by Kühlewind et al. found, with a completely different methodology, that 8.2% of the tested paths clear the `CE` bit as well [8]. This behavior makes it impossible for the client to report to the server that its incoming traffic experienced congestion, making ECN unusable. The high `CE` clearing rate (11.2%) over IPv6 that we found is to put in perspective with the increasing amount of ECN-Capable IPv6 servers (47.52%) [8]. The first may interfere with the latter, as ECN may be negotiated but unable to report any congestion.

We also find a Europe-based IPv4 router that sets the Congestion Experienced (`CE`) bit of packets regardless of the ECN-Capable Transport (`ECT`) bit, probably a legacy router that extendedly modified the DSCP (`ToS`) field.

B. Actions

Fig. 6(b) displays the middleboxes actions breakdown. A middlebox is classified as a rewriting middlebox if it has at least one policy that has been spotted rewriting a packet. For this figure, we did not take into account routers IP DSCP rewriting as this practice is not breaking TCP/IP end-to-end assumptions, nor RFC recommendations.

We found that routers had rewriting policies enabled in 8.5% of IPv4 paths and 12.3% of IPv6 paths. We also found a rather small amount of paths comprising middleboxes with dropping policies; 0.6% over IPv4 and 0.5% over IPv6. Then, we found a few proxies: 51 in IPv4 and 3 in IPv6.

As shown in Table 1, the drop rates for ECN-setup `SYN` packets are quite low; 0.37% over IPv4 and 0.35% over IPv6.

[2]IP::FlowLabel for IPv6 probes.

If the client that wants to establish an ECN-capable connection is not configured to retry the connection without the TCP ECN flags, it will have no connectivity. The number we found are in-line with Langley that stated that less than 1% of sites appeared to have connectivity problems due to the use of ECN by clients [16]. We found a few middleboxes that discarded `MPCapable` and `MPJoin` (MultiPath TCP [13]) TCP options.

We observed several servers located behind a unique IPv6-to-IPv6 Network Prefix Translation middlebox (NPTv6).

VI. Related Work

Medina et al. [17] report one of the first detailed analysis of the interactions between transport protocols and middleboxes. They rely on active probing with `tbit` and contact various web servers to detect whether Explicit Congestion Notification (ECN), IP options, and TCP options can be safely used. The `TCPExposure` software developed by Honda et al. [4] is closest to `tracebox`. It also uses specially crafted packets to test for middlebox interference. Wang et al. [2] analyzed the impact of middleboxes in hundreds of cellular networks. This study revealed various types of packet modifications. More recently, Craven et al. [18] proposed TCP HICCUPS to reveal packet header manipulation to both endpoints of a TCP connection. HICCUPS works by hashing a packet header and by spreading the resulting hash into three fields (in case one is changed).

These tools provide great results, but they are limited to specific paths as both ends of the path must be under control or must implement particular techniques in the TCP/IP stack. Further, none of them measures the IPv6 network or provides a path-impairment oriented middlebox policy taxonomy.

VII. Conclusion

Middleboxes are becoming more and more popular in mostly every type of networks. Those middleboxes are supposed to be transparent to users but it has been shown the contrary. In particular, they impact the TCP protocol and its extensions. In addition to this, there is no formal classification of middleboxes according to their effects on packets, on traffic, or on the network quality experienced by users.

This is what we have addressed in this paper. We proposed a path-impairment oriented middlebox policy taxonomy, that aims at categorizing the initial purpose of a middlebox policy as well as its potential unexpected complications. We also confronted our taxonomy to the reality on the ground through an IPv4 and IPv6 measurement campaign based on `tracebox`. We established a dual-stack survey of middleboxes that implement rewrite, drop, and proxy policies that may harm the performance of regular traffic and protocol deployability. Then, we discussed our results in light of our taxonomy.

In the near future, we plan to gather data about home-NAT devices and obtain client-side middlebox policies breakage, to expand our view from the network core and server-side middlebox policies that we analyzed in this paper. We also want to include some security aspects to our taxonomy. Some of them were briefly discussed in this paper, but we aim at extending our middlebox policy taxonomy to include security concerns and middlebox-related complications by performing exhaustive custom made measurements.

Acknowledgments

This work is funded by the European Commission funded mPlane ICT-318627 project. Authors lso thank kc claffy and her team at CAIDA (in particular, Young Hyun) for letting them using the Archipelago infrastructure.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] B. Hesmans, F. Duchene, C. Paasch, G. Detal, and O. Bonaventure, "Are TCP extensions middlebox-proof?" in *Proc. Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox)*, December 2013.

[6] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," Internet Engineering Task Force, RFC 6824, January 2014.

[7] 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.

[8] M. Kühlwind, S. Neuner, and B. Trammell, "On the state of ECN and TCP options on the Internet," in *Proc. Passive and Activement Measurement Conference (PAM)*, March 2013.

[9] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," Internet Engineering Task Force, RFC 3168, September 2001.

[10] B. Trammell, M. Kühlewind, d. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger, "Enabling Internet-wide deployment of explicit congestion notification," in *Proc. Passive and Active Measurement Conference (PAM)*, March 2015.

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

[12] Z. Qian and Z. M. Mao, "Off-path TCP sequence number inference attack - how firewall middleboxes reduce security," in *Proc. IEEE Symposium on Security and Privacy (SP)*, May 2012.

[13] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extension sfor multipath operation with multiple addresses," Internet Engineering Task Force, RFC 6824, January 2013.

[14] M. A. Lemley and L. Lessig, "The end of end-to-end: Preserving the architecture of the Internet in the broadband era," University of California at Los Angeles, Technical Report 2000-19, October 2000.

[15] k. claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: from art to science," in *Proc. IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, March 2009.

[16] A. Langley, "Probing the viability of TCP extensions," 2008, see https://www.imperialviolet.org/binary/ecntest.pdf.

[17] A. Medina, M. Allman, and S. Floyd, "Measuring interactions between transport protocols and middleboxes," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2004.

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