H2020 European Union funding
for Research & Innovation

Measurement and Architecture for a Middleboxed Internet

H2020-ICT-688421

Middlebox Classification and Initial Model

Author(s):	ULg	Benoit Donnet (ed.), Korian Edeline
	UNIABDN	Iain R. Learmonth
	SRL	Andra Lutu

Document Number:	D2.1
Internal Reviewer:	Brian Trammell
Due Date of Delivery:	30 June 2017
Actual Date of Delivery:	2 July 2017
Dissemination Level:	Public

Disclaimer

The information, documentation and figures available in this deliverable are written by the Measurement and Archticture for a Middleboxed Internet (MAMI) consortium partners under EC co-financing (project H2020-ICT-688421) and does not necessarily reflect the view of the European Commission.

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The user uses the information at its sole risk and liability.

Contents

European Commission | Horizon 2020 European Union funding for Research & Innovation

Executive Summary

The Measurement and Architecture for a Middleboxed Internet (MAMI) project has as a goal the development and experimental deployment of a Middlebox Cooperation Protocol (MCP). In work package 2 (WP2) a model of types of potential middlebox interference is developed on the background of measurements of Internet path transparency. We use this model as input for the design of the MCP, as well as to evaluate the MCP in a controlled environment.

This deliverable describes the first step to tackle this task by describing a taxonomy and initial model of middlebox behavior. The described taxonomy is based on observed middlebox behavior in the wild, and will serve as a basis for building a simulator for the controlled reproduction of a variety of middlebox behaviors. Further, a more detailed model of observed single path conditions, also derived from measurements, is described that is used to provide a comparable description of measured path impairments as a input for protocol design decisions and for data preservation.

1 Introduction

In MAMI WP2 (“Experimentation”), the project aims (among others) to classify and model middlebox behaviors, in order to develop a simulator for testing new protocols that enable cooperation between endpoints and middleboxes, as developed in WP3.

The design of new protocols must cope with an Internet full of middleboxes, and each mechanism in these protocols must be assessed with respect to how middlebox-proof it is [10, 8]. For protocol designers, a summary of the potential kinds of middlebox interference with network traffic is a valuable asset for this assessment.

The purpose of this deliverable is to report what has been achieved so far in MAMI WP2. In particular, we describe efforts made in proposing a classification and taxonomy of middleboxes. We propose a path-impairment-oriented middlebox policy taxonomy that categorizes the orginial purpose of a certain middlebox policy as well as its potential unexpected complications for traffic on the path. Further, we also describe the observable conditions that these policies and complications lead to, as a common format for ongoing longitudinal measurement studies of middlebox prevalence as well as as input to the protocol design process itself.

We first report results on large-scale measurements done with `tracebox` [6] and NAT `Revelio` (Chap. 2), with the purpose of detecting and identifying middlebox behaviors in the wild. Based on these observations, we next propose a middlebox taxonomy (Chap. 3) that aims at categorizing the initially intended purpose of a middlebox policy as well as its potential unexpected complications. Finally, we describe conditions as observed on the path by active, passive, and/or hybrid measurement. These conditions are the high-level output of the `PATHspider` measurement tool, and form core of the data model of the Path Transparency Observatory (PTO) (Chap. 4), to be described in detail in a future deliverable.

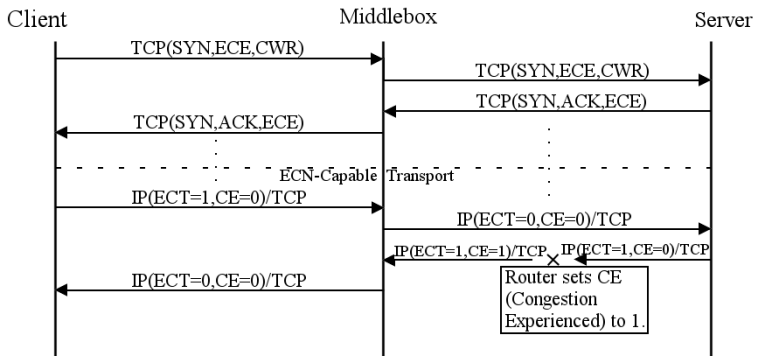


Figure 1: Clearing IP ECN bits.

2 Measurement and Assessment of Middlebox Behavior

2.1 Example Malfunctions

Middleboxes cause various malfunctions, especially with end-to-end protocols, by dropping packets that contain certain options or by stripping those options from, e.g., TCP `SYN` segments to prevent them from being negotiated [11]. However, middlebox-related problems are not limited to stripping and dropping and thus can be much more subtle. In this section, we describe three different problems as examples to what problems middleboxes may cause when end-to-end header information is modified on the path. A notion of these problems have been derived from measurements performed by MAMI partners in preparation of the MAMI and continued in D1.1 as well as others even before that.

2.1.1 Explicit Congestion Notification

In this setup, the middlebox allows 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 clears the ECN bits in the IP header, rendering other on-path devices unable to report any congestion.

As shown in Fig. 1, the client request an ECN-Capable connection with a remote server by sending a TCP `SYN` segment: both TCP ECN header flags ECN Echo (ECE) and Congestion Window Reduced (CWR) are set. To confirm the use of ECN the server sends back a TCP `SYN+ACK` segment with the `ECE` flag set. Both packets are forwarded unmodified by the middlebox. After successful negoatiation, the following packets of the connection are marked as ECN-Capable Transport (`ECT`) in the IP header by both ends, but the middlebox systematically clears both IP bits carrying either the `ECT` codepoint or the `CE` (Congestion Experienced) codepoint if congestion ocurred on the path between an endpoint and the middlebox. That means if an intermediate router sets the `CE` (Congestion Encountered) codepoint of a packet to signal

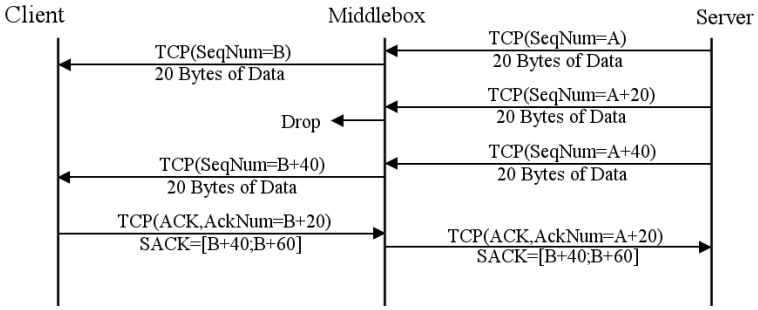


Figure 2: TCP Initial Sequence Number re-shuffling middlebox and Selective ACKnowledgement.

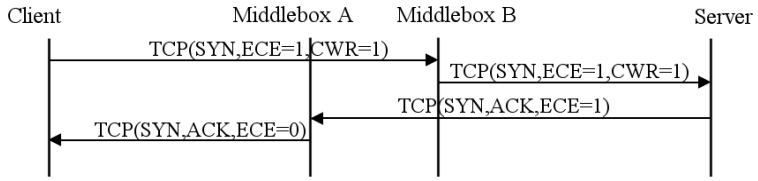


Figure 3: State announcement and asymmetric paths/load balancing.

congestion, this information will be lost as it will be cleared by the middlebox afterward [12].

2.1.2 TCP Sequence Number

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

The server sends 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. On 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 notifies the sending side by acknowledging the first and the third data segment using the `ACK` number and the `SACK` option. The middlebox modifies 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 [10].

2.1.3 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.

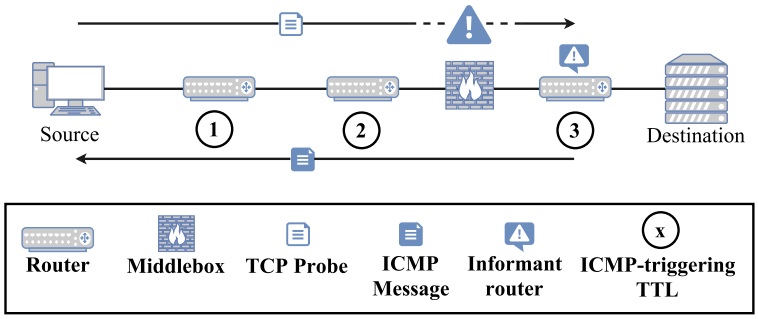


Figure 4: Middlebox detection with `tracebox`.

Both ends are trying to share state related data (i.e., `ECE` bit). The path between them is asymmetrical and ingress and egress traffic cross different middleboxes, A and B respectively. Middlebox A clears the `ECE` bit and middlebox B does 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 [17], but not widely deployed [19].

2.2 Middlebox Detection and Identification Methodology

In this section, we describe the measurements methodology that we use to not only detect a certain impairment (as done in D1.1.) but also identify the specific middlebox on a network path. Two measurement campaigns for middlebox identification have been performed: (i) a `tracebox`-based campaign (Sec. 2.2.1) that looks at middleboxes in general and (ii) a NAT `Revelio`-based campaign (Sec. 2.2.2) for specifically revealing Carrier-Grade NATs which even goes beyond the scope of the initial measurements presented in D1.1. This section describes the measurement tools and setup, while the observed results and inferred information about middlebox behavior are discussed in the next section.

2.2.1 `tracebox`-based Campaign

To reveal the presence of middleboxes along a path, we use `tracebox` [6], an extension to the widely used `traceroute` [20] that allows middlebox detection and IP-level localization. More details on `tracebox` can be found in D1.1 [7].

We deployed `tracebox` [6] on wired IPv4 networks via PlanetLab. We selected the maximal number of nodes available for each campaign (between 108 and 129). Target destinations have been selected using the top 1M Alexa list of websites that we resolved once to 594,241 unique addresses beforehand. We conducted 14 campaigns over nine different ports (80, 8080, 8000, 8800, 443, 53, 12345, 1228, 34567) with TCP SYN probes including the most commonly used TCP options (`MSS` and `SACKP`) for a period of two months between March, 3rd and May, 8th 2016, each campaign lasting between three and seven days. The total amount of data collected corresponds to 1.3TB. This dataset is freely available[1].

[1] `https://observatory.mami-project.eu/`

Campaign		Raw Data			Middleboxes	
ID	Port	#IPs	#ASes	#Probes	#Paths w/MB	#ASes w/MB
1	80	886,065	2,953	40,392,061	2,175,335	337
2	80	886,263	2,955	42,774,781	2,382,938	347
3	8000	816,656	2,891	24,860,295	1,773,409	252
4	80	887,171	2,950	41,767,341	2,346,832	334
5	8080	816,192	2,897	24,252,300	1,667,765	226
6	8800	820,653	2,889	38,750,758	1,673,998	241
7	443	856,918	2,938	41,590,151	3,415,643	364
8	12345	813,152	2,880	39,234,092	2,286,052	311
9	8080	813,955	2,895	22,466,692	1,609,383	220
10	80	884,808	2,955	42,866,154	2,369,670	342
11	1228	812,213	2,885	39,489,438	2,282,698	329
12	443	882,658	2,955	41,593,420	3,454,363	361
13	34567	820,305	2,893	39,225,840	1,806,605	269
14	53	820,698	2,887	39,202,907	2,784,658	260
HTTP		930,842	2,969	250,983,908	5,696,282	510
non-HTTP		888,596	2,939	267,482,322	3,011,418	368
Total		948,457	2,977	518,466,230	5,832,789	661

Table 1: General statistics about data collected with `tracebox`, after filtering, before preprocessing.

Once the raw data collected, we selected the nodes that remained available during all campaigns (89 PlanetLab nodes). We also filtered out PlanetLab-related errors and kept only probes that reached the destination or that expired or triggered an ICMP message after at least 10 hops. Doing so, we obtained an exploitable dataset made of 518 millions probes. 34% of those probes reached the destination, 64% expired, and 2% triggered an ICMP message before reaching the destination. The high amount of timeout is due to probes with non-HTTP ports that are dropped by a firewall before reaching the destination. From this filtered dataset, we extracted 38 millions observations of middlebox behavior (i.e.: a single modification, addition or deletion of single field of a probe, on a single path, in the course of one campaign). We note that we did not witness any significant difference in middlebox behavior towards HTTP and non-HTTP probes, apart from the fact that HTTP probes are less likely to be blocked, and thus are able to highlight more middleboxes.

During the entire measurement campaign, we observed 948,457 different responsive hops (excluding vantage points and targets addresses), scattered over 2,977 different ASes. The most represented ASes are Cogent (35.7% of all addresses – Tier 1 network), CenturyLink (10.6% – Tier 1 network), Telia Carrier (6.3% – Tier 1 network), NTT (3.4% – Tier 1 network), Rackspace (1.8% – cloud services), Level3 (1.6% – Tier 1 network), and Chinanet (1.5% – Chinese ISP). The corresponding addresses are geographically distributed in North America (40.4%), Europe (37.5%), Asia (18.7%), Latin America and Caribbean (2.7%), and Africa (0.7%) according to the regional Internet registries. The same addresses were registered under 189 different country codes. Table 1 provides additional general statistics about data collected. In particular, the last two columns show the precedence of middleboxes in the dataset and thus on the Internet.

This dataset was obtained through a 3-step preprocessing. The objective of the preprocessing

is to merge multiple `tracebox` observations of a given middlebox into a single identifier. In the first step so-called *offender*, i.e., the router preceding the middlebox on a given path, are identified. Then in the second step, offenders are grouped together to obtain a common profiles of observed middlebox behavior. Finally, offenders are aggregated into unique middleboxes[2].

2.2.2 NAT `Revelio`-based Campaign

In this section we describe NAT `Revelio`, the test suite we designed to actively detect one specific set of middleboxes, namely Network Address Translation (NAT) in the Internet Service Provider's (ISP) access network. In Fig. 5 we depict the residential setup we consider for NAT `Revelio` in the context of a DSL access network. The home network may have an arbitrary topology consisting of multiple hosts, routers and switches including multiple levels of NATs. The home network connects to the Internet through the Customer Premises Equipment (CPE) also known as "home router" or "home gateway". The access link connects the CPE with the ISP access network. In the case of DSL technology, the access network includes the digital subscriber line access multiplexer (DSLAM), the broadband remote access server (BRAS) and the Core Router (CR). The ISP network connects with the rest of the Internet.

In terms of IP addressing, the home network generally uses private IP address space. The home gateway usually performs the NAT function (home-NAT) from the private addresses within the home network to the addresses used in the ISP access network which may be public, private or shared. In some cases, end-users can configure several different realms of private addresses within their home network in the context of cascaded home NATs. Independently of the home network topology, when a host within the home network communicates with a host in the rest of the Internet, the private address used by the host in the home network translates to a public address that we call the Globally Routable Address (GRA). For the majority of the residential Internet market, the ISP configures the GRA on the Internet-facing interface of the CPE and the NAT function in the CPE translates from the private addresses in the home network to the GRA. An alternative, incipient, setup is one including an additional NAT function that operates in the ISP network (in addition to the NAT function in the CPE) and performs the final translation to the GRA. These configurations are usually called Carrier Grade NAT (CGN), Large Scale NAT (LSN) or NAT444. In this case, packets flowing between the home network and the Internet go through two upstream NAT-capable devices: the CPE (customer grade NAT) and the ISP NAT (Carrier-Grade NAT).

In order to discern where the translation to the GRA occurs, NAT `Revelio` performs active tests from a device connected to the home network. The *probe* running NAT `Revelio` connects to the home network and may or may not be directly connected to the CPE, i.e., there may be multiple hops, including ones performing NAT function(s), between the *probe* and the CPE. The target of the active tests the *probe* performs are servers located in the Internet (Fig. 5). NAT `Revelio` does not require any cooperation from the ISP beyond forwarding Internet packets to and from the customer.

The goal of NAT `Revelio` is to detect whether the *device performing the translation to the GRA* (hereinafter, the GRA-NAT) *resides in the home network or in the ISP network*. In order to do this, NAT `Revelio` attempts to pinpoint the location of the GRA-NAT with respect to the access link. If the GRA-NAT lies between the *probe* and the CPE, we conclude that the user is not

[2] A detailed description of the data preprocessing is out of the scope of this deliverable. We let a more detailed description for Deliverable D1.3 due to June 2018.

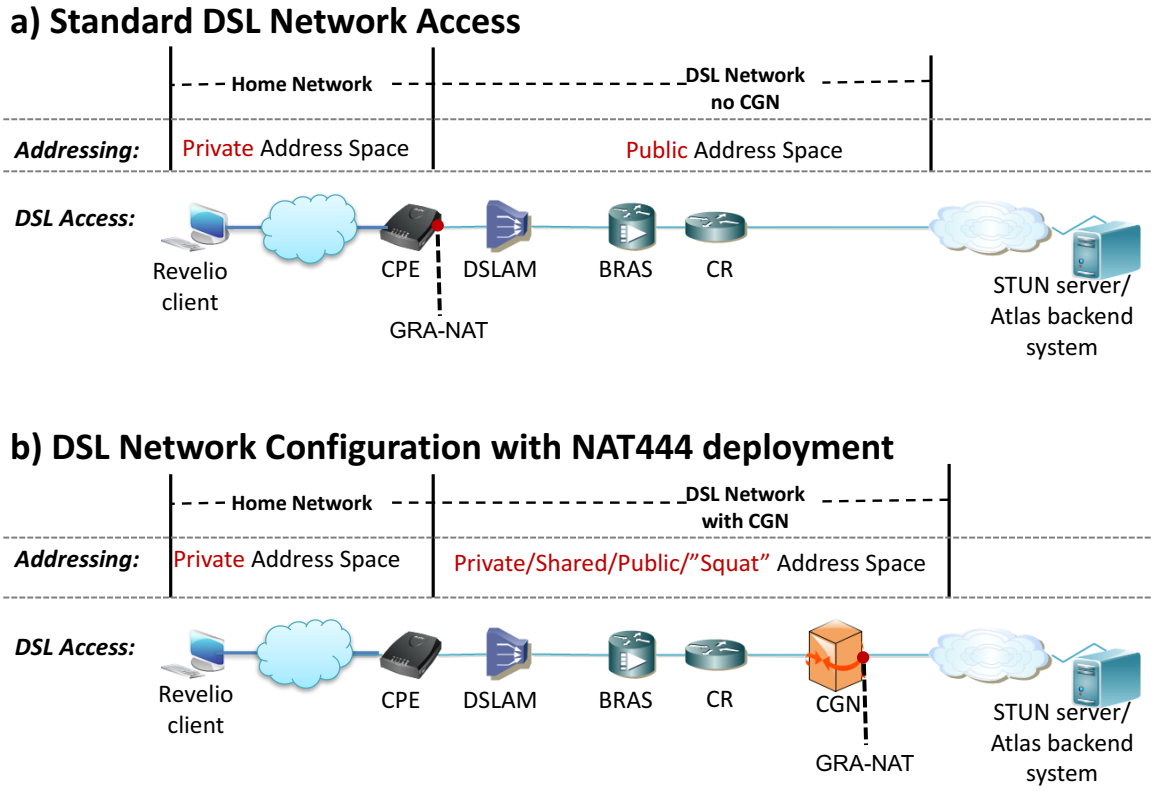


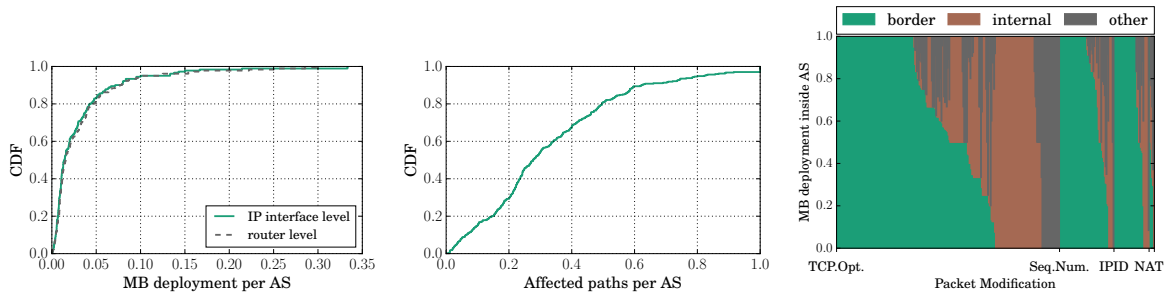
Figure 5: NAT `Revelio` experimental setup for a DSL access network (the DSLAM, BRAS and CR are standard elements in the DSL architecture). The NAT `Revelio` client runs on a device connected to the home network, whose exact topology we do not know. In the case of (a) standard DSL network access, the CPE performs the translation to the GRA, thus it is the GRA-NAT. In the case of (b) standard DSL network configuration with NAT444 deployment, the CGN is the one performing the translation to the GRA, thus it is the GRA-NAT.

behind a CGN. If the GRA-NAT lies after the CPE, we conclude that the ISP deploys CGN. To achieve this, NAT `Revelio` needs to determine the location of the GRA-NAT and the location of the access link with respect to the *probe* and compare them.[3]

2.3 Middlebox Behavior in the Wild

In this section, we describe measurement results that drives the middlebox taxonomy as described in the next section. We first look at middlebox *prevalence* (e.g., if a firewall is set up, does all traffic go through that firewall? – see Sec. 2.3.1) and next at *persistence* over time (e.g., is a middlebox up and running all the time, or do we observe any dynamics as for IP networks? [15, 1, 5] – see Sec. 2.3.2). Third, we focus on the prevalence of a particular kind of middlebox: Carrier-Grade NATs (Sec. 2.3.3). Finally, we look at middlebox policies (Sec. 2.3.4).

[3] A detailed description of NAT `Revelio` techniques for revealing GRA-NAT is out of the scope of this deliverable. We let a more detailed description for Deliverable D1.3 due to June 2018.



(a) Infrastructural deployment of middleboxes

(b) Middlebox popularity

(c) AS topological location of middleboxes

Figure 6: Middlebox prevalence evaluation.

2.3.1 Middlebox Prevalence

The *prevalence* of middleboxes for an AS, as we define it, can be assessed at three levels. First, we examine the infrastructural deployment of the middleboxes, i.e., the proportion of middleboxes deployed by the AS compared to typical Layer-3 devices (IP interfaces or routers). This allows us to know whether ASes deploy as many middleboxes as Layer-3 devices, as previously stated for enterprise networks [18]. Second, we consider the popularity of middleboxes, i.e., the fraction of paths, inside an AS, that is harmed by at least one middlebox. A value of 0, for the popularity, would mean that all observed paths traversing the AS do not encounter a middlebox (and, as such, the AS does not deploy any middlebox). On the contrary, a value of 1 means that all observed paths are harmed by a middlebox. In that extreme case, one can say that middleboxes are prevalent as they impact every packet traversing the AS. Finally, we evaluate where in its topology an AS is likely to deploy a middlebox. Two locations are envisioned: (i) at the border of the network (meaning that it is very likely the middlebox will process every packet entering/leaving the AS network) or (ii) in the core of the AS network (meaning that middleboxes are deployed for very dedicated services and traffic).

To assess the first point, Fig. 2.6(a) shows the proportion of middleboxes deployed by ASes as a fraction of IP interfaces discovered in our measurement campaign (plain line) and as a fraction of routers (dashed line) – the alias resolution has been performed using CAIDA's ITDK dataset [4]. At the AS level, the deployment of middleboxes is rather marginal compared to Layer-3 devices[4]. In general, less than 5% of the deployed infrastructure is dedicated to middleboxes. For instance, Cogent (one of the most seen AS in our measurement campaign) deploys between 1% and 1.5% of middleboxes compared to Layer-3 devices.

Fig. 2.6(b) shows the middlebox popularity (X-Axis) as a cumulative mass. We observe that in 20% of the cases, more than 50% of the AS paths are affected by a middlebox. This suggests thus that, in some cases, middleboxes are prevalent, even if, in general, an AS does not deploy that many middleboxes in its network. However, it is worth noticing that this metric has its inherent limits. For instance, Cogent is one of the most visible ASs, in our dataset ($>$ 44

[4]Obviously, the amount of observed middleboxes in our dataset is strongly related to the way we performed the measurement campaign. And, in particular, results are biased due to the target being the Top 1M Alexa web sites. We believe, however, that results given in this report are a lower bound, and given so a first insight on how ASes deploy and use middleboxes.

millions of paths) and we detect a middlebox presence on more than 2 million paths traversing Cogent. This gives a popularity of "only" 5%, while we believe middleboxes are quite prevalent across this AS.

Further, we found that that a large part of the middleboxes (52.48%) are located at the border and a smaller part is internally in the AS's network (36.55%). In some cases we were unable to derive the offender position (for instance, because all addresses were non publicly-routable), or the offender appears as being at the border of the AS for some paths, and AS internal for others. Those two situations appeared in 9.07% of the cases. Finally, a few offenders appeared to have been moved from the border to the core, or vice versa, from one campaign to another but this is a rare case (1.9%). This is aligned with results presented in Fig. 2.6(b). Indeed, if the majority of paths, within an AS, are affected by (at least) one middlebox, it is expected to see this middlebox at the ingress (or egress) of the AS, creating so a kind of bottleneck in which the majority of the traffic must go through.

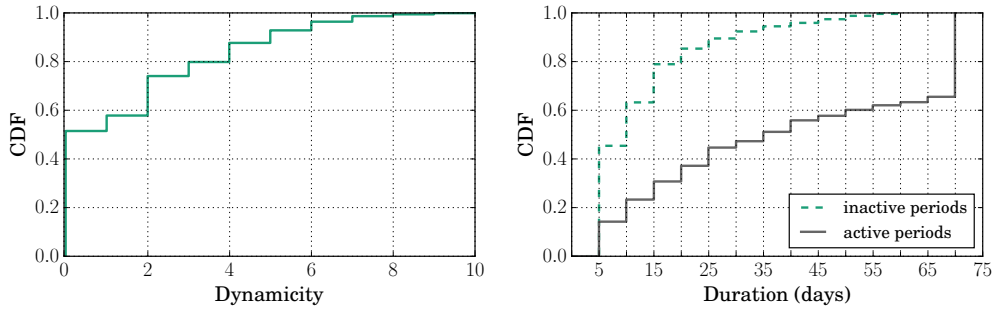
Fig. 2.6(c) shows the distribution of middlebox location, split in four categories, per ASes on which they are deployed. The categories are the following: (i) TCP options: This regroups traffic engineering middleboxes that modify, strip or add TCP options (`MSS` and `SACKP`), that we highlighted based on the value of the actual TCP options, TCP offset and IP Length, (ii) TCP sequence number modification, security-related middleboxes that set the TCP initial sequence number to randomly chosen value, (iii) IP-ID modification, middleboxes that set the IP-ID field to unique non-null value for each transmitted packet, and, (iv) NAT, middleboxes that remap the source port of our probes, often combined with other previously described behavior. We observed 254 ASes that deploy middleboxes that modify TCP options, among which 88 deploy all of them at their border. 149 ASes deploy at least half of them at the border, 44 ASes deploy all TCP-options-modifying middleboxes in their core. 20 ASes also deploy such middleboxes, but we cannot make a conclusion as to their position. 62 ASes deploy middleboxes that shuffle the TCP Sequence Number. 31 ASes deploy all those middleboxes at their border, while 48 ASes deploy at least half of their shuffling middleboxes also at the border. Finally, only 4 ASes deploy those middleboxes in their network core. 40 ASes deployed middleboxes that check for IP-ID uniqueness, 25 of them put all such middleboxes at their border, 5 others in their core. 6 ASes were observed making use of NATs, but without privileged position.

Overall, we found that ASes tend to deploy most of their middleboxes at their border, at the exception of 65 ASes (19% of the ASes with labelled middleboxes) that deploys at least half of their middleboxes in their core.

2.3.2 Middlebox Persistence

In the section, we analyze the persistence of middleboxes over time. We consider that a middlebox is active during a campaign if at least one of its offender IP address was responsive during this campaign, and that it was used for labeling. We consider a middlebox as inactive if at least one of its offender IP address was responsive and none of them was used for labeling. We take care not to count the cases were the middlebox is not observable because of the absence of any RFC1812-compliant informant router. Finally, a middlebox is considered offline if none of its offender IPs were responsive during a whole campaign.

From the set of labelled middlebox, we selected those that were responsive to probes with HTTP and non-HTTP ports, excluding middleboxes located on path segments invisible to certain non-HTTP probes because of port-based blocking, to be able to analyze their persistence



(a) Global overview (per middlebox).

(b) Duration (per period, 8,788 active periods, 5,121 inactive).

Figure 7: Middlebox Dynamics.

through all the campaigns. From the 8,005 labelled middleboxes, we selected 5,888 in this manner.

We compute the dynamics by comparing the set of active campaigns to the set of inactive campaigns. The *middlebox dynamic* is thus the number of time the middlebox switched from active state to inactive and vice versa. We chose to ignore the offline states not to draw conclusions from the absence of observations, to make sure that the address was not simply not observable (e.g., the probe took a different path). A value of zero means that a middlebox is constant (i.e., it is always up and running), while a value larger than one provides the number of times a middlebox switches from an active (respectively inactive) state to an inactive (respectively active) state.

Fig. 7 displays the distribution of middleboxes dynamic. In particular, Fig. 2.7(a) provides a general overview of how middleboxes are affected by dynamics. We see that 51% of the middleboxes are stable over time (i.e., a value of 0); the corollary being that the other half of the middleboxes exhibits a dynamic behavior. The second most frequent value is 2, meaning that a middlebox will switch its state twice over the measurement campaign. The maximum value of ten is quite rare but suggests that some middleboxes are very unstable over time.

Fig. 2.7(b) gives the states duration in terms of consecutive days for both active (plain line) and inactive (dashed line) periods of middleboxes. To compute this figure, we normalized campaigns durations to 5 days. The maximum duration for an active period is 70 days (in 38% of the cases), i.e., the whole measurement campaign duration. This corresponds to middleboxes with a dynamic of 0. Moreover, we observe that 50% of the active periods are longer than 35 days (half of the campaigns). Inversely, we notice that 44% if inactive periods are short-lived, lasting only 5 days, while 20% of inactive periods were longer than 20 days.

Globally, we showed on the one hand that the largest part of middleboxes (more than 75%) tends to be constantly active, or presenting few short periods of inactivity. On the other hand, a minority of middleboxes are more dynamic, alternating between active and inactive states 3 times or more.

As a next step we plan to further Investigate the cause of middlebox dynamics.

2.3.3 Prevalence of CGNs in Europe and the U.S.

To further investigate the presence of CGNs, as one specific kind of and probably most dominant kind of middlebox, we deployed NAT `Revelio` on a total of 5,121 different customer lines (or probes) by leveraging access to United States Federal Communication Commission's (FCC) initiative "Measuring Broadband America" (FCC-MBA) and also RIPE Atlas, the measurement platform deployed and maintained by the RIPE NCC. In function of the upstream NAT configuration, NAT `Revelio` classifies each *probe* into one of the following cases:

- *inconclusive* (cases NAT `Revelio` was unable to draw any conclusion due to incomplete or inconsistent results).
- *no home NAT* (i.e., the *probe* where NAT `Revelio` runs is directly connected to the public Internet).
- *simple home NAT* (the CPE performs the GRA-NAT).
- *Carrier Grade NAT* (the GRA-NAT is outside the home network, in the ISP's network).

Results are aggregated by the inferred upstream NAT configuration (Table 2).

Inconclusive. For 1,276 *probes* (307 *SK probes* and 969 *Atlas probes*), NAT `Revelio` gave inconclusive results either because none of the tests could run on the *probe* or because we did not obtain enough information to properly interpret the results we were able to collect. Our approach is conservative and tags as inconclusive the case of mixed responses from different tests. For example, traceroute limitations and ICMP traffic being filtered along the path to the external target server hamper our capacity to identify the access link. Without knowing the location of the access link, when the end-user deploys several levels of NAT in the home, we cannot draw conclusions regarding the presence of NAT in the ISP. These *probes* account for approximately 24% of the total, (12% of the *SK probes* and 36% of the *Atlas probes*). We discard these cases from further analysis.

No home NAT. NAT `Revelio` found that in 299 different cases (85 in *SK probes* and 214 *Atlas probes*), the NAT `Revelio` client was running on a *probe* configured with a public IP address that was also the GRA. These *probes* were operating in the public Internet, which implies that the lines were not connected behind a NAT solution. In all these cases, the *traceroute to the GRA* test also confirmed the lack of a NAT solution in the corresponding ISPs.

Simple home NAT. Out of the rest, for 3,454 *probes* (2,009 *SK probes* and 1,445 *Atlas probes*) NAT `Revelio` established the presence of simple home NAT and excluded the possibility of further NAT in the ISP. NAT `Revelio` reports the simple home NAT configuration (and, thus, the lack of NAT in the ISP for the respective line) when at least one of the *traceroute to GRA* and *invoking UPnP actions* tests establish that the home gateway is performing the GRA-NAT. In the case of the UPnP test, for 1,300 *SK probes* the address retrieved through UPnP from the CPE matched the GRA, concluding that the CPE was the GRA-NAT. For 815 *SK probes*, the NAT `Revelio` client was unable to communicate with the CPE through UPnP, either because the CPE did not supported UPnP or because the *SK probe* was not directly connected to the CPE. In the case of the `traceroute` to the GRA test, for 2,965 *probes* (1,520 *SK probes* and 1,445 *Atlas probes*) NAT `Revelio` located the GRA-NAT before the access link, concluding that the CPE was also the GRA-NAT. As a interesting data point, using *pathchar to the GRA* test NAT `Revelio` purged 165 of cases where the CPE replied as being two different hops, creating

ISP ID	CC	Tech.	# of probes	Inconclusive	Simple Home NAT	Carrier Grade NAT	Confirmed
1 (Undisclosed ISP)	US	Satellite	76	0	0	76	Yes
2 (Kabel Deutschland)	DE	Cable	49	27	14	8	Partially
3 (Fastweb)	IT	Fiber	26	14	8	4	Yes
4 (OTE)	GR	DSL	21	5	14	2	No Reply
5 (Liberty Global)	NL	Cable	280	133	146	1	Yes
6 (Zen)	UK	DSL	32	11	20	1	No Reply

Table 2: List of ISPs with at least one probe with ***positive*** NAT `Revelio` result (i.e., operates behind a CGN). We report the Country Code (CC), the access technology (Tech.), the total number of probes we tested for that ISP (# of probes), the number of probes for which NAT `Revelio` gave inconclusive results (Inconclusive), the number of probes NAT `Revelio` tested negative (Simple Home NAT), the number of probes NAT `Revelio` tested as positive (Carrier Grade NAT) and the current status of the confirmation from representatives of the ISP with positive NAT `Revelio` results (Confirmed). For the latter, we mark this field with *Yes* if the ISP confirmed the NAT `Revelio` results at the IP level, *Partially* if the ISP confirmed they use CGN but did not confirm the specific IP lines tested, *No Reply* if we did not get any feedback from the ISP.

false positives. In particular, NAT `Revelio` detected this behavior in one single ISP for 78 out of 228 *probes*.

Carrier Grade NAT. For 92 *probes* in 6 ISPs (76 *SK probes* in 1 ISP and 16 *Atlas probes* in 5 ISPs) NAT `Revelio` detected the presence of CGN technology in the ISP's network. Table 2 details the number of *probes* that tested positive for CGN per ISP.[5] We identified one satellite provider in the U.S. where all *probes* tested positive for CGN. For the rest of the ISPs, we detected a mix of some *probes* that tested positive for CGN and others that did not. Overall, about 2% of the *probes* tested positive for CGN. About 10% of the ISPs we tested hosted at least one *probe* that tested positive for CGN. Of these latter ones, only one ISP had a widespread deployment of CGN, while the other ISPs presented a few scattered *probes* that tested positive, hinting a localized deployment, e.g., possibly for trials or suggesting a specific service.

2.3.3.1 Validation of NAT `Revelio` Results

NAT `Revelio` tested 5,121 Internet lines in 64 different ISPs worldwide. In total, it reported 92 end users with an upstream CGN, which connected to 6 different ISPs. We validated both the positive (upstream CGN) and negative (no upstream CGN) results at the IP level through different means, including direct contacts with the involved ISPs or, in one case, using the WHOIS database information.

Positive NAT `Revelio` Results. We obtained confirmations at the IP level from 4 ISPs (89

[5] We only disclose the names of the ISPs we tested using the RIPE Atlas platform. We are currently awaiting the approval of the FCC for disclosing the names of the ISPs we tested with the FCC-MBA testbed.

probes) for the presence of CGN in their network for the lines we tested and received no replies from the other 2 ISPs (3 *probes*). In Table 2 we report on the status on communication with the ISPs for which NAT `Revelio` identified the presence of CGN. In particular, for ISP#1 from Table 2 – the satellite provider in the US for which all *probes* tested positive – the operator confirmed that its normal configuration includes performing the NAT function in the ISP network and that all the 76 lines that tested positive were indeed behind a CGN. ISP#3 (Fastweb) confirmed both the positive and the negative NAT `Revelio` results. For ISP#5 (Liberty Global) from Table 2, the GRA associated with the *probe* is actually tagged in the WHOIS database (in the *Organization* field) as CGNAT (the other 279 *probes* in the same ISP did not have a GRA in the subnet marked as CGN). ISP#2 (Kabel Deutchland) from Table 2 confirmed that it is using CGN in its network. However, we did not obtain explicit confirmation from their representatives that the exact lines we detected as positive are actually behind a CGN, which is why we marked it as a *partial* confirmation.

Based on the ground truth we collected, we conclude that NAT `Revelio` did not generate any false positives. Thus, provided that NAT `Revelio` can successfully run, its precision[6] is 100% reported to the set of probes which the ISPs validated.

Negative NAT `Revelio` Results. Out of the 5,121 lines NAT `Revelio` tested, its results pointed to a simple NAT configuration (no CGN) for 3,454 *probes* in 63 different ISPs. For the negative results, we obtained validation from 4 ISPs for which all *probes* tested negative for upstream CGN in the ISP. The 4 ISPs account for 508 *probes*. We mention that (confirmed) negative results from NAT `Revelio` testing do not preclude the existence of CGN technology in the corresponding networks.

Based on the ground truth we collected, we conclude that NAT `Revelio` did not generate any false negatives. Thus, provided that NAT `Revelio` can successfully run, its recall[7] is 100% reported to the set of probes which the ISPs validated. However, the NAT `Revelio` methodology reported inconclusive results in 24% of the cases (this numbers drops to 12% if the measurement platform supports both UPnP and traceroute based tests).

2.3.4 Middlebox Policies

By looking in depth into middleboxes, we have observed common behavior. In particular, middleboxes exhibit some *capabilities*, i.e., what a middlebox expects to achieve, its purpose. Those capabilities are typically:

1. translation (e.g., network address translation)
2. normalization (i.e., limit of a protocol features to a restricted subset to prevent the use of unwanted features)
3. correction (e.g., sequence number randomness or `IP-ID` uniqueness)
4. packet marking (i.e., via the legacy `IP ToS` field, encroaching on the `Explicit Congestion Notification (ECN)` bytes)

[6] The precision represents the ratio between the number of true positives and the sum of the true positives and the false positives.

[7] The recall represents the ratio between the number of true positives and the sum of the true positives and the false negatives.

Capability	Percentage
translation	1.3%
normalization	41.0%
correction	54.8%
packet marking	2.4%
authorization	NA
unknown	0.5%

Table 3: Capabilities distribution in the `tracebox` dataset.

5. authorization (e.g., source-based filtering or TCP window checking

In this document, we establish the observation count for the `tracebox` dataset and match each observations to middlebox categories. The categories that we use are explained in the following section. We identified six middlebox policies: TCP option stripping, TCP sequence number shuffling, NATs, blind IP ECN modifications, modification of the TCP reserved field and modification of the TCP UrgentPtr field.

Table 3 shows the distribution of those capabilities in the `tracebox` dataset. We did not observe authorization policies because they are mostly out of scope for our measurement methodology. Our probes inferred few translation capabilities because NATs modification are required to be reverted in the ICMP payload, making them invisible to regular `tracebox` [9]. However, we recently developed a technique to be able to detect NATs regardless of this behavior [21]. We plan to use it in the future to highlight NAT deployment. We also detected few packet marking capabilities because we only kept such policies if they are prone to create transport-level impairment, which is rare. Finally, we observed large proportions of normalization and correction middleboxes, with a large proportion in the the second category. This can be explain by the fact that middleboxes implementing these policies tends to be located closer to access networks, and such positioning affects more paths. A second explanation is that normalization policies have more incentives to be invisible than correction policies because they reveal pieces of information about AS traffic engineering.

We have also observed that middleboxes along a path can lead to complications for the functioning of the traffic on the path. The causes and consequences of those complications are evaluated in Table 4. We find that the most common impairments are traffic disruption and disabled features, and that they are caused by, respectively, incomplete packet modifications and over-normalization.

Complications		Percentage
Causes	malconfiguration	2.9%
	incomplete modifications	56.1%
	over-normalization	41.0%
Consequences	traffic disruption	57.2%
	blocked traffic	1.3%
	disabled features	41%
	unknown	0.5%

Table 4: Causes and consequences of middleboxes along a path.

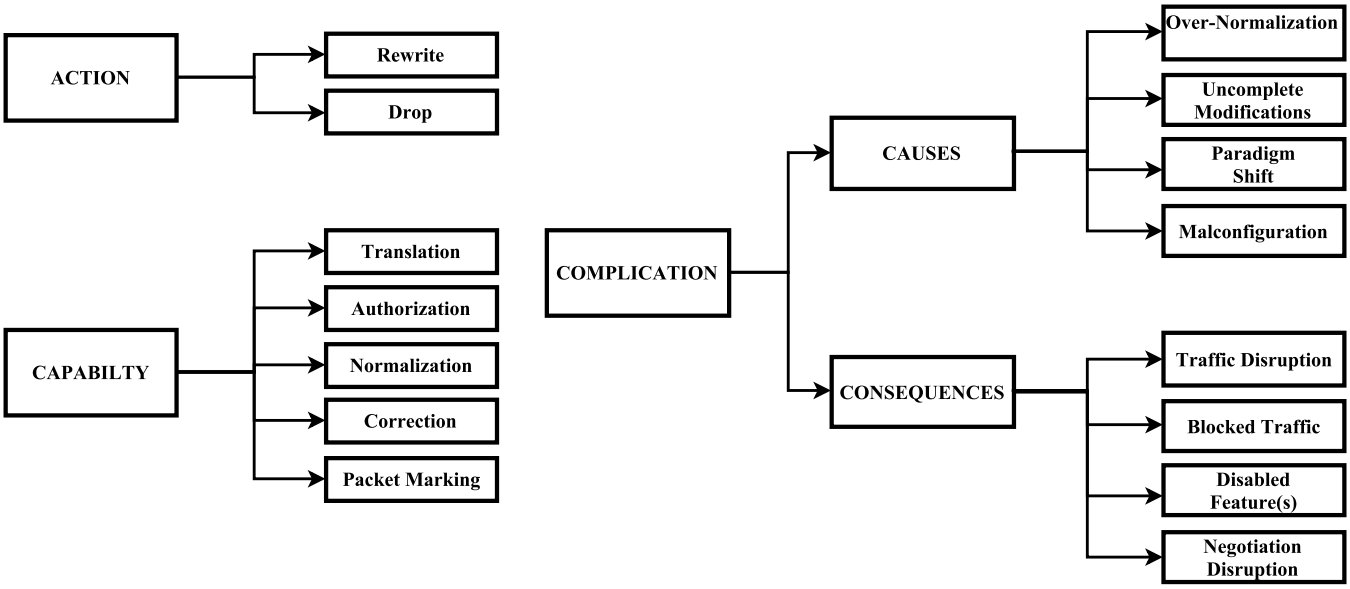


Figure 8: A path-impairment oriented middlebox policy taxonomy.

3 Middlebox Taxonomy

In this chapter, we present a categorization of inferred middlebox behavior, derived from measurements as described in D1.1 and in the previous section. We divide middlebox behavioral characteristics into those that describe the inferred capabilities and those that describes operation characteristics of the network device on which those capabilities are implemented.

3.1 Middlebox Policies and Capabilities

In this section, we propose a path-impairment oriented middlebox policy taxonomy, that aims at categorizing the initially intended purpose of a middlebox policy as well as its potential unexpected complications for traffic passing through it.

We chose to classify middlebox *policies* rather than middleboxes themselves because the latter often combine multiple policies. Our taxonomy focuses on packet-mangling middlebox and the network interferences that may result. The middlebox capabilities have been detected by using `tracebox` [6], reachability testing [2] or PATHSpider [13], and we focus on these capabilities that are prone to cause transport-level complications.

As we aim at characterizing middlebox-related network interferences rather than establishing an exhaustive middlebox taxonomy [3], we focus on single-hop modifications and ignore other aspects. further 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 many and various reported security flaws created by middlebox policy implementations (see, for instance, Qian and Mao [16]).

We describe each policy implemented in a single (i.e., not multi-hop) middlebox by three as-

pects (i.e., meta-categories), each one including several taxa (i.e., categories); (i) **Capabilities**, *what* the policy expects to achieve, its purpose; (ii) **Action**, *how* the policy tries to achieve its goals, in the fate of a packet crossing a middlebox that implements this policy; (iii) **Complication**, the possible resulting path connectivity *deterioration*. Fig. 8 illustrates our path-impairment oriented middlebox policy taxonomy. Each middlebox policy has to fall in *at least* one taxon for each of the three points of view in order to be consistent with the taxonomy.

Fig. 8 displays the taxonomy and shows its three facets: *Actions*, *Capabilities*, and *Complications*. These meta-categories are described in the subsequent sections. This taxonomy is distinct from the middlebox taxonomy presented in RFC3234 [3] 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.

3.1.1 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 two basic kinds of middlebox policy actions: *Drop* and *Rewrite*. This aspect is decisive because middlebox policies that apply 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 intended purposes is further in detail described in the next section, Sec. 3.1.2. As they break the TCP/IP end-to-end principles, they may cause various problems to protocol end-to-end functions, as further described in Sec. 3.1.3.

3.1.2 Capabilities

A middlebox *capability* is a basic feature that can be configured to enforce a policy. The eponymous meta-category describes the purpose of a middlebox capability (e.g., what it intends to achieve). As we already noticed earlier in this section, a capability can be classified differently if we consider the capability 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 Capabilities: *Translation*, *Authorization*, *Normalization*, *Correction* and *Packet Marking*.

Translation capabilities perform dynamical mapping of certain fields of a flow packets between two networks in order to be understood by each one of them (e.g.: NATs).

Authorization capabilities are implemented by a middlebox that discards a flow if it meets certain criterions. For example, firewalls performs source-address-based filtering or TCP window checking.

Normalization capabilities are all middlebox policies that transform a flow by modifying fields in the transport header, stripping or adding options, to comply to a network policy. For example, middleboxes that limit a protocol features to a restricted subset to prevent the use of unwanted

features.

Correction capabilities are middlebox policies that aims at fixing endpoint implementations by transforming flows. For example, sequence number randomness or `IP ID` uniqueness.

Packet Marking capabilities (`IP Differentiated Services Code Point (DSCP)`) are normally not considered middlebox behavior but a wanted part of the forwarding plane. We chose to keep those in as we can also detect these policies by our measurements methodology. Indeed, we identified middleboxes that still perform packet marking via the legacy `IP ToS` field, encroaching on the `Explicit Congestion Notification (ECN)` bytes. This faulty of ECN bytes leads to the inability to correctly use ECN, and even worse, depending on the value being written, may conceal congestion reports by other nodes on the path or falsely permanently signals congestion leading to unnecessary reductions of TCP's congestion window. We observed routers with defective *ECN* implementations, with consequences similar to defective `DSCP` marking.

3.1.3 Complications

We describe here the potential *Complications* caused by middlebox policies by examining them from two points of view: (i), their technical causes, which are directly related to their initial purposes and, (ii), the associated actions (respectively Sec. 3.1.2 and Sec. 3.1.1), and their unfortunate consequences, i.e., causes and consequences.

3.1.3.1 Causes

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

Over-normalization refers 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 [11]. 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 as 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 SACK option; the modification being therefore incomplete.

A *Paradigm shift* (2-way to `n`-way) happens when both ends running a protocol assume 2-way peering relationships. Middleboxes, by applying modifications *in the middle of the path*, break the end-to-end principle underlying the Internet architecture, and therefore cause both endpoints 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 try to share state related data or to negotiate capabilities, this phenomenon may, in certain

	ACTION			CAPABILTY			COMPLICATION								
	Rewrite	Drop	Translation	Authorization	Normalization	Correction	Packet Marking	Over-Normalization	Uncomplete Modifications	Paradigm Shift	Malconfiguration	Traffic Disruption	Blocked Traffic	Disabled Feature(s)	Negotiation Disruption
Strip unknown TCP option	X				X			X			X			X	
TCP ISN shuffling	X					X			X			X			
Drops MPTCP		X			X			X					X		
SNAT	X		X						X			X	X		
8-bits DSCP (ToS)	X						X				X	X		X	

Figure 9: Examples of middlebox policies

scenarios, put both ends in conflicting states or, combined with an unfortunate load balancing arrangement, may distort protocol negotiations [10]. 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.

Abother cause of complication that we identified are *Malconfigurations*. This refers to vendor implementation or design errors in capabilities, leading to faulty middlebox policies. This is not to be confused with misconfigurations, that are human errors in middlebox configuration. We choose to leave aside those types of errors because they don't involve any technical failures.

3.1.3.2 Consequences

The *Consequences* are the final outcome of network complications, i.e. what both ends actually experience. 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 achieved symmetrically, the consequences are limited to the inability to use the restricted features; it is a *Feature-disabling* policy. If the modifications are asymmetric and the negotiation is not resilient enough, the policy may fail to disable the feature and lead to inconsistent protocol states [10]. Policies resulting in the latter consequences are categorized as *Negotiation disruption* policies.

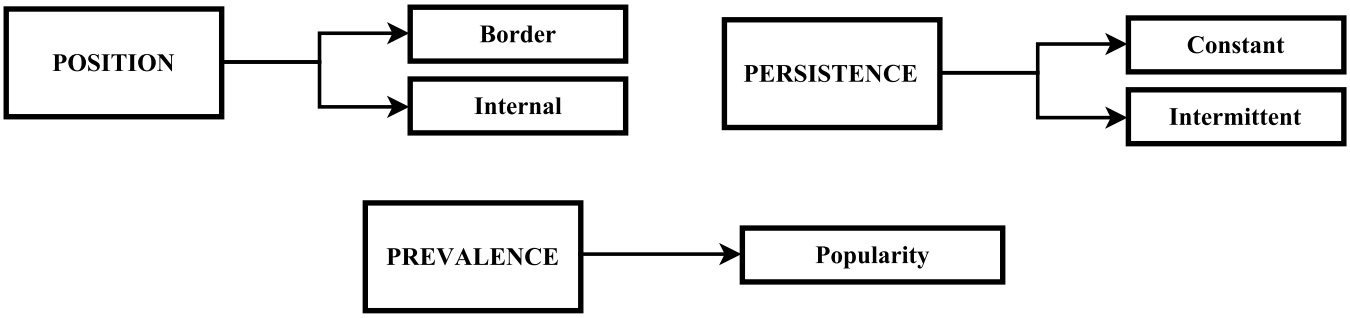


Figure 10: A middlebox operational characteristics.

3.2 Operational Characteristics

Based on the observed middlebox dynamics in our measurements we also assign operational characteristics to each middlebox implementing one or multiple policies in this middlebox taxanomy, as depict in Fig. 10.

The prevalence is defined by the popularity of a middlebox which is determined by its impact on the number of effected paths. The popularity can be assess by several metrics such as the fraction of paths inside the Autonomous System that is affected by a middlebox or the absolute amount of flows affected by the middlebox.

The position of the middlebox is the location where an Autonomous System deploys middleboxes. Based on in our measurement results, we consider two locations: (i) at the *border* of the network (meaning that it is very likely the middlebox will process every packet entering/leaving the AS network) or (ii) *internal*, in the core of the AS network (meaning that middleboxes are deployed for very dedicated services and traffic).

Finally, we consider the persistence of middleboxes over time. A middlebox can be active, if it implements a certain capability, or inactive, if it was observed not exhibiting the same middlebox behavior anymore at a certain point of time. Based on these possible states the persistence of middlebox is defined over a certain period of time as either: (i) *constant*, if the middlebox was observed to be active over this period, or (ii) *intermittent*, if the middlebox was observed to be active and inactive at different times during this period.

4 Modeling Observed Path Conditions

In addition to the high-level classification of middlebox behavior in the previous section that is intended to provide input for a middlebox behavioral model to be used for simulative protocol evaluation, in this section we further define a vocabulary to described single observed conditions that have been detected in our path measurement studies described in D1.1 and as also exemplarily described in Sec. 2.1. The condition described in this deliverable reflect an analysis of our measurement data observed so far into a generalization that can be used as a representation in the Path Transparency Observatory (PTO) in a comparable manner, and thereby provide input for the design process of new network protocols. The PTO will be further detailed in the forthcoming deliverable D1.2. Further these conditions are also used as the native output from the `PATHspider` measurement tool that have been developed with a focus on middlebox path impairment measurement, enabling an easy integration of the `PATHspider` output into the PTO. `PATHspider` was introduced in D1.1.

4.1 Hierachical Structure

Path conditions are organised in a hierachical structure with terms concatenated with periods, e.g.: $ecn.connectivity.works$. The first term specifies the protocol feature or extension for which path transparency is tested on the path. In some cases more than one term may be used for this as is the case for $dscp$, where the second term indicates the actual value of the DSCP field used for the measurement.

The term following the protocol feature is the property tested when using the protocol feature. There are reserved properties that apply over a number of protocol features and these share the same generalised definition. Others are specific to the protocol feature.

The following sections describe conditions that have been derived from from our current measurements as an example set of possible conditions.

4.2 General Properties

4.2.1 $*.connectivity.\{works, broken, offline, transient\}$

The $connectivity$ property represents whether or not a connection fails when attempting to use a protocol feature or extension. It does not include whether or not the feature or extension was successful and provided any benefit to the connection, only that there was not a connection failure.

Connectivity is considered broken if an attempt to connect without the protocol feature or extension succeeds but fails when using it. A transient failure is the reverse of this and in most cases will be representative of noise in the collected data. If the connection fails in both cases then the target should be considered offline.

4.2.2 $*.negotiation\_attempt.\{succeeded, failed\}$

If the protocol feature or extension supports a negotiation mechanism, as ECN does for example, the $negotiation\_attempt$ property represents whether or not the negotiation attempt was successful. If the connection failed this property should not be present.

4.3 Protocol Specific Properties

Futher properties are specific to protocol features and so are not generalised in their definitions. These will only appear, with rare exceptions, when connectivity using the protocol feature has been successful.

4.3.1 Explicit Congestion Notification

4.3.1.1 $ecn.ipmark.\{not_ecn, ect\_zero, ect\_one, ce\}.seen$

This property represents whether a particular mark was seen in the Explicit Congestion Notification (ECN) codepoint bits in the IP header for any packet in a connection. There is no not_seen version of this property, the absence of this property only means that a packet with the codepoint was not received, not that it was not sent.

4.3.2 TCP Fast Open

4.3.2.1 $tfo.cookie.\{received, not\_received\}$

This property represents whether or not a TCP Fast Open (TFO) cookie was received in response to a request for a cookie. It is not a failure for there not to be a cookie received, but if a cookie is received then when used the data on the SYN packet should be acknowledged.

4.3.2.2 $tfo.syndata.\{acked, not\_acked, failed\}$

This property represents the disposition of data sent on the SYN packet after a TFO cookie was received. If no TFO cookie was received, no data will have been sent on the SYN and so this property will not appear. $acked$ indicates that the SYN ACK acknowledges the SYN as well as its data, not_acked indicates that the SYN ACK acknowledges the SYN only, and $failed$ indicates that data on SYN leads to connection failure.

4.3.3 Differentiated Services Codepoints

4.3.3.1 $dscp.[0-63].replymark.[0-63]$

This property represents the value of the Differentiated Services Codepoint (DSCP) observed for the reply where a connection was successfully made to test DSCP connectivity. The second term represents the value used for the measurement as the observed value may be dependent on the value sent.

5 Conclusion

This document describes the current progress in MAMI WP2 on classifying and initial modeling of middlebox behavior. This work is based on data of observed impairments derived from measurements described in D1.1 and an extensive dataset collected with `tracebox` and NAT `Revelio` to also detect and identify specific middleboxes and their location, prevalence, and persistence over time.

Derived from those detailed measurements, we describe a path-impairment oriented middlebox policy taxonomy that categorizes the intended purpose of a middlebox policy, as well as its potential unexpected complications for traffic passing through those boxes. We further describe an example set of single conditions observed from these measurements campaigns that are used as a common vocabulary for the output of the `PATHspider` measurement tool, that has been explicitly developed by the MAMI project for path impairment measurements, and the input for the MAMI Path Transparency Observatory.

The taxonomy described in the deliverable is a basis for a detailed middlebox behaviorial model that will be used for implementing a simulator and will be further detailed in the next deliverable in WP2. This simulator enables the evaluation of middlebox interference in protocol development such as for the Middlebox Cooperation Protocol (MCP) as developed by the MAMI project and to be further elaborated in deliverable D3.2. In addition, the described condition format enables access and long-term preservation of comparable data on middlebox impairments as provided by the PTO. This data is also used as input to the protocol design process of the MCP itself.

References

[1] D. G. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan. Topology inference from BGP routing dynamics. In *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, Nov. 2002.

[2] C. Basile, D. Canavese, Pitscheider C., A. Lioy, and F. Valenza. Assessing network authorization policies via reachability analysis. *Computers & Electrical Engineering*, February 2017. to appear.

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

[4] Center for Applied Data Analysis. The CAIDA UCSD internet topology data kit, March 2016. See `http://www.caida.org/data/internet-topology-data-kit`.

[5] I. Cunha, R. Teixeira, and C. Diot. Measuring and characterizing end-to-end route in the presence of load balancing. In *Proc. Passive and Active Measurement Conference (PAM)*, March 2011.

[6] 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.

[7] B. Donnet, K. Edeline, R. Zullo, B. Trammell, M. Kühlewind, O. Gonzlez de Dios, I. R. Learmonth, and A. Lutu. Initial report on measurement development and deployment. Deliverable 1.1, Measurement and Architecture for a Middleboxed Internet (MAMI), December 2016.

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

[9] S. Guha, B. Ford, S. Senthil, and S. Pyda. NAT behavioral requirements for ICMP. RFC 5508, Internet Engineering Task Force, April 2009.

[10] 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.

[11] 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.

[12] M. Kühlewind, 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.

[13] I. R. Learmonth, B. Trammell, M. Kühlewind, and G. Fairhurst. PATHspider: A tool for active measurement of path transparency. In *Proc. ACM/IRTF/ISOC Applied Networking Reserach Workshop (ANRW)*, July 2016.

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

[15] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.

[16] 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.

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

[18] 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.

[19] 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.

[20] V. Jacobson et al. traceroute. man page, UNIX, 1989.

[21] R. Zullo, A. Pescapè, K. Edeline, and B. Donnet. Hic sunt NATs: Uncovering address translation with a smart traceroute. In *Proc. IEEE/IFIP Workshop on Mobile Network Measurement (NMM)*, June 2017.