Middleboxes in the Internet: a HTTP perspective

Shan Huang
Queen Mary University of London
shan.huang@qmul.ac.uk

Félix Cuadrado
Queen Mary University of London
felix.cuadrado@qmul.ac.uk

Steve Uhlig
Queen Mary University of London
steve.uhlig@qmul.ac.uk

***Abstract*—Middleboxes are widely used in today's Internet, especially for security and performance. Middleboxes classify, filter and shape traffic, therefore interfering with application performance and performing new network functions for end hosts. Recent studies have uncovered and studied middleboxes in different types of networks. In this paper, we exploit a large-scale proxy infrastructure, provided by *Luminati*, to detect HTTP-interacting middleboxes across the Internet. Our methodology relies on a client and server side, to be able to observe both directions of the middlebox interaction. Our results provide evidence for middleboxes deployed across more than 1000 ASes. We observe various middlebox interference in both directions of traffic flows, and across a wide range networks, including mobile operators and data center networks.**

I. Introduction

Middleboxes such as firewalls, load balancers and deep packet inspection (DPI) boxes are a major part of today's network infrastructure. A middlebox can be defined as any intermediary network device performing functions other than standard functions of an IP forwarding between two end hosts [1]. Currently, the reasons driving the deployment of middleboxes come in two main categories: (1) security [2], [3], [4], [5] to enhance the visibility of network traffic and enable the enforcement of security policies, and (2) performance enhancements [2], [6], [7] through traffic shaping, caching and transparent proxying.

Compared to forwarding devices such as switches and routers, middleboxes are complex. Indeed, they operate on flows of packets at multiple layers of the network stack, from the network layer to the application layer, and do so at line rate. Middleboxes interfere with end-to-end packet transmission, application functionality, and restricting or preventing end host applications from functioning properly [1]. Middlebox interference can be categorized into three types. First, middleboxes intentionally drop or filter packets according to policies [8], [9]. For example, network administrators filter P2P file sharing traffic to avoid the legal implications of copyrighted content [10]. Second, middleboxes modify the content of packets [11], [8], [5]. Some web proxies modify HTTP headers to control meta information between client and server (e.g., cache preferences). Finally, middleboxes also inject forged packets, e.g., for blocking purposes. A notorious example is the Great Firewall of China (GFC) that blocks specific sites by injecting spoofed DNS responses, with obvious consequences in terms of Internet censorship [12].

Middleboxes are widely used in various types of networks. From a survey of 57 enterprise network administrators, it was concluded that there are probably as many middleboxes as routers inside the network [2]. Also, the survey of edge-network behavior [13] showed evidence of middlebox traffic manipulation in common ISPs. As much as it is widely expected that middleboxes are widely present across today's networks, there is still relatively limited evidence regarding how widely middleboxes are deployed, and how much they interfere with traffic flows.

At the same time, Internet traffic is changing, e.g., HTTPS represents a significant fraction of Internet traffic [14]. Considering the complexity of middleboxes, today's applications and network traffic, we argue that better methodologies must be developed to detect and analyse middlebox interference on traffic flows.

In this work, we develop such a methodology, and exploit the *Luminati* proxy network to launch HTTP requests from vantage points distributed in nearly 10,000 ASes across 196 countries. Our methodology relies on crafted probes and controlled client-server interactions. All traffic traces we produced will be made publicly available.

Our contributions are twofold. First, we introduce our methodology to detect middlebox interference based on a client-server architecture. We also explain how to use the *Luminati* platform to run large-scale measurements. Second, based on our methodology, we find evidence for a significant amount of middlebox interference on both directions of the traffic flows in different networks. We observe a wide variety of injected HTTP headers in HTTP requests, some known and some never reported before. Surprisingly, we even observe new headers that are only added by mobile networks and cloud platforms. Overall, we find that injected headers expose the presence of multiple types of middleboxes across diverse networks. Further, the interference on HTTP responses often reveals the corresponding functions of the middleboxes, such as proxying, caching, URL filtering, and WAN optimization.

The reminder of this paper is structured as follows. We discuss the prior middlebox detection methodologies in related work (Section II). In Section III, we introduce the *Luminati* platform and our own methodology. We examine middlebox interference on HTTP requests in Section IV-A and describe response manipulation in Section IV-B. Finally, Section V summarizes our paper and discusses further work.

II. Related Work

A number of recent studies have explored middleboxes, especially the behavior and impact of middleboxes on traffic

flows.

Back in 2011, Honda et al. [11] developed a tool made of a client and a server, and examined middlebox interference on TCP across diverse networks. Their idea of controlling both end hosts provided the ability to generate, capture and analyse TCP segments freely. However, the considered middlebox interference was focused on TCP SYN/SYNACK segments.

Also in 2011, Wang *et al.* [8] did large-scale measurements in more than 100 cellular ISPs, unveiling NAT and firewall policies of carriers. Their methodology relied on probes running on smartphones and a dedicated server. The results from this work demonstrated the importance of understanding the interference from these policies, affecting the performance of applications and mobile devices. This work attracted the attention of cellular network carriers and mobile application developers, making them reflect on the impact of middleboxes. Despite the achievements of this work, middleboxes are much more complex and diverse, and therefore require considering wider interactions.

Tracebox [5] is a traceroute-like tool to identify packet modifications performed by upstream middleboxes, and help locate the involved middleboxes hop-by-hop. Similar to traceroute, Tracebox sends probes with increasing TTL values and waits for ICMP time-exceeded replies. Comparing the crafted packets with the ICMP time-exceeded replies, Tracebox finds out the modifications in the packet header or the payload, inferring the presence of some middleboxes. However, the absence of a server-side prevents Tracebox from detecting middlebox interference in both directions of the traffic. Tracebox is a seminal work in the area of middlebox interference. However, to better understand middlebox interference, especially at the application layer, a different methodology is necessary.

Netalyzr is a network measurement service, which provides different types of network functionality tests, thanks to a large number of volunteers [13]. Using this service, Weaver *et al.* found that 14% of the clients from their collected measurements passed via web proxies [6]. Further, this service has also been used in cellular networks [15], [16]. It was shown that 58% of 6918 sessions from 119 countries were going through HTTP proxies, and 18% of the sessions were using a DNS proxy [16]. Moreover, 13% of 299 mobile operators were observed to manipulate HTTP headers for user privacy, security and network operations. In this paper, we confirm the prevalence of middleboxes across the Internet, and different from Netalyzr-based works, we expose the extent to which they specifically interact with HTTP headers.

Meanwhile, the Open Observatory of Network Interference (OONI) [17] has processed some network measurements which aim to detect internet censorship, traffic manipulation and other signs of surveillance since 2012. The OONI project is under the Tor project, collecting millions of network tests across more than 90 countries. The researchers published the testing methodology to identify HTTP Header Field Manipulation and the collected HTTP headers on the website.

Recently, [18] used peer-to-peer network Hola to explore HTTP headers in-the-wide, revealing that 25% of measured ASes modify HTTP headers. Part of this work has confirmed the presence of some middleboxes. While, the focus of this work was not on detecting middleboxes, it shed insight on the types of headers, expose network and regional trends. Indeed, our dataset covers nearly 10,000 ASes, which is wider than the dataset of [18], illustrating much more middleboxes in various of networks.

These studies attempt to explain the mechanisms of detecting and locating particular middleboxes in networks, investigating the header manipulation in the wild. On the other hand, our work aims at detecting any behaviors or effects of middleboxes on HTTP application traffic flows in diverse networks, and discuss the networks where the middlebox interference occurs.

III. Methodology and Dataset

In this section, we describe our methodology, aimed at detecting the presence of middleboxes through their interaction with HTTP requests and answers. To do this, we adopt a client-server architecture, with control on both sides of the end-to-end flow. Our client-side generates crafted probe packets and matches the sent probes with the responses. The server-side responds to the crafted probes, potentially modified on the way by middleboxes, and compares the received probe with the original one sent. The server also sends crafted responses back to the client-side. All probes sent and received are collected and kept for further analysis. Note that an earlier description of our methodology can be found in [19].

To sample middlebox interference across the Internet, we want the probes to be sent through a physical infrastructure distributed across the Internet. However, the infrastructure used to send the probes should provide significant and as representative as possible vantage points, i.e., beyond a purely academic one such as PlanetLab. Indeed, PlanetLab is not suitable for our middlebox study. We have used our methodology on the PlanetLab infrastructure as well, but hardly found any middlebox deployement this way, only a few non-representative instances of middleboxes. Therefore, in this paper, we use the commercial Peer-to-Peer (P2P)-based HTTP/S proxy service, *Luminati*, based on the Hola network, to launch HTTP requests across the Internet.

A. Hola and Luminati

Hola is a P2P VPN service, which allows users to route traffic over a large number of country peers, from nearly 280 countries. These country peers run on users' machines, therefore based on a variety of devices, e.g., laptops, mobile devices, and distributed across various types of networks. In practice however, Hola forwards traffic via super proxies located in a few countries (e.g., the UK or the USA), instead of going though each country peer.

To get full advantage of the vantage points from the Hola proxy network, one needs to rely on Luminati. Luminati is a paid HTTP/S service that is based on the Hola network. Luminati forwards users' traffic via Hola country peers, not the specific super proxy, therefore providing a much larger

set of vantage points from which to launch probes. Furthermore, Luminati enables users to select country peers in the Hola network. For instance, the Luminati users could launch multiple requests from same country peer by adding the same *-session-number* parameter to their subsequent requests. The session number could be a random number, and Luminati will keep forwarding the subsequent requests via the same country peer, for a period of 60 seconds only. After these 60 seconds, Luminati chooses another country peer, potentially different from the prior one. Also, even though in principle a country peer from the chosen country will be used, there is no guarantee that this is actually going to be the case.

In the received HTTP responses, Luminati super proxies add two headers, for logging and debugging the selected country peer (**X-Hola-Timeline-Debug** and **X-Hola-Unblocker-Debug**), carrying the identifier **zID** for the selected country peer. By recording the **zID**, we can check whether our probes are forwarded via the same country peer, even if the country peer has changed its IP address in the meantime.

B. Measurement Methodology

1) Available Country Peers: Although Luminati provides diverse country peers across the Internet, a user is only able to select country peers at the granularity of an AS or a country, without indication about the network type inside which the country peer is. Luminati does not enable a controlled selection of the ingress nodes, and selects the available country peers randomly. As we aim to probe from as many networks as possible in our measurements, we launch HTTP requests via as many as possible country peers. For each country among the 275 available countries of the measurements, we keep sending probes using random session numbers. We launch sets of 500 requests for a given country at a time. If for a given country, after a set of 500 requests, we observe 5 duplicate country peers, we stop probing this country for a while[1].

2) DNS resolution: In principle, Luminati allows users to set the DNS resolution by adding the **-dns-remote** parameter to the request. This DNS resolution can be done at two different places. The first place where the DNS resolution can be done is at the super proxies, which send the DNS query to Google's public DNS service. The second place is at the country peer, which sends the DNS query to its local DNS resolver. To ensure that our probes are forwarded by country peers, and not the super proxies, we forward the original HTTP GET request directly to the country peer, and ask for the DNS resolution to be also done by the country peer. Figure 1 illustrates the process of launching HTTP requests using the Luminati platform. For all requests, we select the same super proxy from the United Kingdom, using all available country peers.

3) Crafted Probes and Answers: With our methodology, we control both the client and server side, sending crafted probes (HTTP requests) and responses (HTTP responses).

[1]But we guarantee that we launch a total of 1000 requests in any given country in our measurements, even if a single country peer is available.

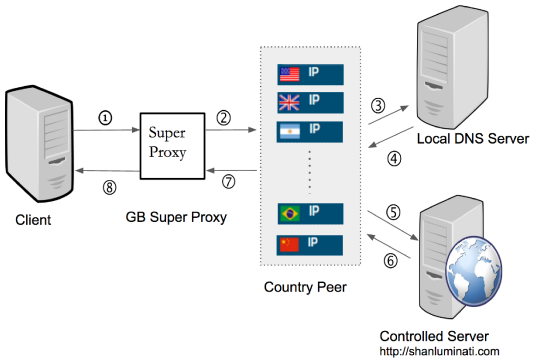

Fig. 1: Luminati-based Probing Methodology.

Unfortunately, Luminati does not provide direct access to the country peers for probes to be sent, but only allows requests to be forwarded through them. Figure 2 shows the traffic exchange process between country peers and our controlled server. We treat country peers as our original clients, and detect middlebox interference on the path between the country peer and the server. As shown in Figure 2, the actual origin of the probes is one of the available Luminati country peers acting as client, and the target of the probes is a server under our control, located in Ireland Amazon cloud data center, running the server-side of our methodology.

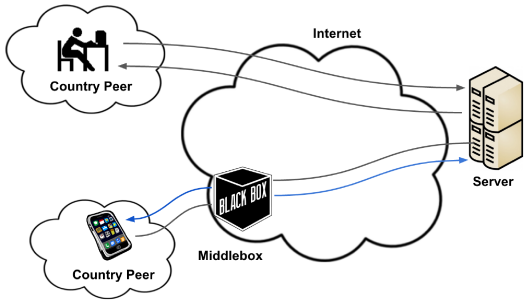

Fig. 2: Traffic Flow between Country Peer and Target Server.

TABLE I: Default HTTP Headers.

	Name	Default Value
Request Header	Host	shanluminati.com
	Accept-Charset	utf-8
	Accept	*/*
	User-Agent	curl/7.35.0
	Accept-Encoding	gzip
	Connection	keep-alive
	Keep-Alive	max=20, timeout=10
Response Header	Accept-Ranges	bytes
	Connection	keep-alive
	Keep-Alive	max=20,timeout=10
	Content-Type	text/html; charset=UTF-8
	Date	Mon, 15 Aug 2016 16:59:35 GMT
	Etag	90-52ecccfbb0285
	Content-Length	144
	Last-Modified	Thu, 24 Mar 2016 15:07:56 GMT
	Server	Apache/2.4.6 (Red Hat Enterprise Linux)

We set up measurement clients, operating in Amazon cloud instances in Ireland. We launch HTTP requests to our target

server, via the Luminati country peers[2], with 7 default request headers: Host, User-Agent, Accept, Accept-Charset, Accept-Encoding, Keep-Alive and Connection. The default values of these headers are shown in Table I.

Our target server is operating in an Amazon cloud instance, located in Ireland as well[3]. The server acts as a standard Apache HTTP server, waits for the TCP connection and GET request, and returns the HTTP response with a crafted payload. As we are forwarding requests through other users' machines, to avoid potential ethical problems for these users, we only respond to requests with a simple "Hello World" HTML page with 9 default response headers. Although not every country peer would return us with a response, we compared all received HTTP messages with the sent probes. As explained in Section III-A, the super proxy of Luminati adds two headers (**X-Hola-Timeline-Debug** and **X-Hola-Unblocker-Debug**) to HTTP response, illustrating the information of selected country peer. These two headers do not affect or overwrite other original headers. When comparing HTTP responses, we do not take these two headers into consideration, but investigate other headers.

C. Dataset

We now give a brief introduction of our collected dataset. We kept launching requests through different country peers for a period of 5 days, from September 29 until October 3 2016. Table II shows the number of the IP addresses seen through the requests and responses, their AS and country. In Table III, we investigate the AS coverage based on the classification provided by Dimitropoulos et al. [20]. Despite the fine-grained nature of this classification, nearly 67% of the ASes we observe are unclassified (last row of Table III). Also, the classifier abstains from classifying some ASes, when the information about them is not sufficient (labeled as "Missing Sufficient Information" in Table III).

TABLE II: Overview of Dataset

	Requests	**Responses**
# of IPs	401,746	372,603
# of ASes	10,060	9,634
# of countries	196	196

For requests, we rely on the source IP address of the received request at the server. This IP address will either be the one from the Luminati country peer or a TCP-terminating middlebox located between the country peer and our server. Luminati adds the IP address of the selected country peer to the response header, and therefore the IP address we use for the response in our dataset is one of the selected country peers (not of a middlebox). All the corresponding ASes and countries are inferred from these IP addresses. We provide the distribution of selected country peers in Figure 3 and Figure 4. Note that the dataset will be made publicly available upon publication.

[2]As we rely on the Luminaty proxy peers, the location of the clients is technically irrelevant for the middlebox detection.

[3]Only in the case of requests launched from country peers in Ireland, we rely on an Amazon cloud instance in the US as target server to ensure our probes travel through the Internet, not inside the Amazon infrastructure.

TABLE III: AS Classification

AS Classification	**Requests (# of ASes)**	**Responses (# of ASes)**
Customer	927	887
University	324	319
Internet Exchange	3	3
Network Information Center	43	43
Tier 1	39	38
Tier 2	1631	1611
Missing Sufficient Information	197	198
Unclassified	6896	6536

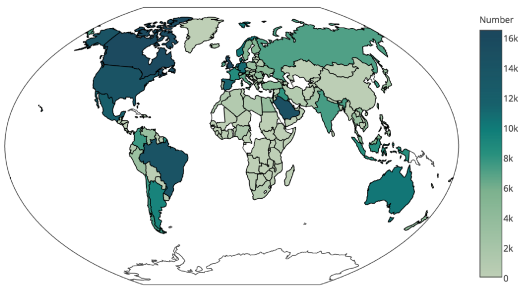

Fig. 3: Number of IP Addresses per Country (max 16433).

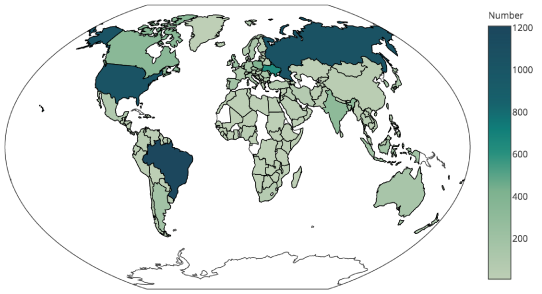

Fig. 4: Number of ASes per Country (max 1200).

IV. Middlebox HTTP interference

By using our methodology described in Section III on the *Luminati* proxy network, we collected data regarding HTTP header manipulation by middleboxes. In this section, we analyse this data, by breaking down the modifications into two main classes, that affect either HTTP requests (Section IV-A) or HTTP responses (Section IV-B). For each type of modification, we will study not only the variety of headers affected, but also try to infer the type of network in which these take place.

A. Request Header Injection

In the upstream direction, i.e., on the path the HTTP request takes towards the server, we see a variety of new headers injected by middleboxes. These headers mostly relate to common network functions, such as proxying, caching, filtering and load balancing. When comparing the injected

TABLE IV: Injected Request Headers Related to Proxy or Cache Functions.

	Injected header	# of ASes	# of countries	Note
Proxy-Related	Via	695	117	Via: 1.1 rcdn9-cd1-dmz-wsa-1.cisco.com:80 (Cisco-WSA/9.0.1-162)
	X-Forwarded-For	535	106	X-Forwarded-For: 192.168.2.157
	X-Proxy-ID	178	58	X-Proxy-ID: 2004304525
	X-IMForwards	30	20	X-IMForwards: 20
	Max-Forwards	5	4	Max-Forwards: 10
	Client-IP	5	7	Client-IP: 10.224.164.34
	Client-ip	3	2	Client-ip: 192.168.23.5
	X-BlueCoat-Via	49	9	X-BlueCoat-Via: fb09b83d12ade53b
	CUDA_CLIIP	19	11	CUDA_CLIIP: 172.16.20.138
	X-IWS-Via	7	6	X-IWS-Via: 1.1 51066FAS (IWSS)
	X-IWSaaS-Via	1	1	X-IWSaaS-Via: 1.1 scannerdy-an-20-3012-a-pro-18293387:8080 (IWSaaS)
	X-RBT-Optimized-By	2	2	X-RBT-Optimized-By: LGEPS-PC-ACC-3070M-A (RiOS 8.6.2c) SC
	RVBD-CSH	1	1	RVBD-CSH: ::ffff:172.25.80.199
	RVBD-SSH	1	1	RVBD-SSH: ::ffff:172.17.12.199
	Surrogate-Capability	8	5	Surrogate-Capability:srv015.guape.zigdigital.com.br="Surrogate/1.0 ESI/1.0"
	X-Tinyproxy	1	1	X-Tinyproxy: 10.192.9.79
	X-If-Via	1	1	X-If-Via: 1.1 i-FILTER84982
Cache-Related	Cache-Control	750	106	Cache-Control: max-stale=0
	Pragma	4	3	Pragma: no-cache
	X-Loop-Control	35	2	X-Loop-Control: 151.233.132.133 151D44BFA8F6E036603564C1B622E01C
	If-Modified-Since	24	13	If-Modified-Since: Thu, 24 Mar 2016 15:07:56 GMT
	If-None-Match	21	11	If-None-Match: "90-52ecccfbb0285"

headers from the requests with those from responses, we see a wider diversity of different headers being manipulated in responses. We also observe that most of the manipulated response headers relate to proxies or caches that inject new headers into requests. Though the sheer numbers do not constitute conclusive evidence, this may indicate that middleboxes affecting the upstream direction (requests) are actually a subset of those affecting the downstream direction (responses). Given that middleboxes are stateful devices that see both directions of the traffic flows, it is natural to expect a significant overlap between manipulations done in both directions of the traffic.

1) Proxies/Caches request header injection: In Table IV, we list all instances of injected headers corresponding to proxies and caches. For each header instance, we also provide the number of ASes and countries of the possible location of the injection. As previously mentioned in our methodology, we infer the AS and country of the source IP address of the received requests. This IP address will either be the one from the Luminati country peer or from a middlebox located between the country peer and our server. Therefore, even though it is not the definitive location of the middlebox, it will be typically at the edge of the Internet given the vantage points provided by Luminati. From how often these headers are observed in different networks, we get a measure of the popularity of these two important network functions overall. Meanwhile, we check the values of the injected headers (see examples provided in the last column of Table IV). We find that the values of the headers are consistent with the names of the headers, reflecting the related network functions played by the corresponding middlebox.

The most frequently injected request header in our dataset is **Cache-Control**. This header sets specific directives for cached copies, and is seen in about 7.5% of all ASes we sample in our measurements. The next most popular injected header is **Via**, injected by proxies to inform end points of its presence, sometimes also adding information about the name and version of the middlebox. We observed the **Via** header across 695 ASes in 117 countries. Middleboxes do more than tell their function. They also add private information about the end-point originating the HTTP request, as from the **X-Forwarded-For** header that carries the IP address of the original client. Doing this is surprising, if the intended usage of proxy is to provide anonymity for end users, since adding the IP address of the original client defeats the very purpose of proxying, by revealing to the server the originator of the query. The next most popular injected header is **X-Proxy-ID**, seen in 178 ASes across 58 countries, which carries the identifiers of the proxies.

Injected HTTP headers also reveal a significant number of vendor-specific middleboxes. For example, **X-IWS-Via** and **X-IWSaaS-Via** are headers added by Trend Micro middleboxes, running the InterScan Web Security service. InterScan Web Security (IWS) is a software appliance that dynamically protects traffic flows on Internet gateway [21]. Another expected header is **X-IWSaaS-Via**, from the Amazon cloud instance inside a Japanese data center. Beside the typical functions of proxying and caching, we find headers related to services such as private IP mapping (**X-Tinyproxy**), traffic flows filtering (**X-If-Via**) and WAN optimization (**RVBD-CSH** and **RVBD-SSH**). Although not very common, these instances provide evidence of the diversity of roles played by middleboxes in today's Internet, way beyond the usual functions such as caching and proxying.

As shown in Figure 5, most of involved ASes are Tier-2 or customer networks, supporting our expectation that the middleboxes are generally located at the edge of networks.

2) Mobile devices/networks request header injection: So far, we have studied injected headers for which the function of the middlebox is straightforward, because the header is well known or the name strongly suggests its function. This is not always the case unfortunately. When the header does not tell us its purpose, we try to guess its function based on its name

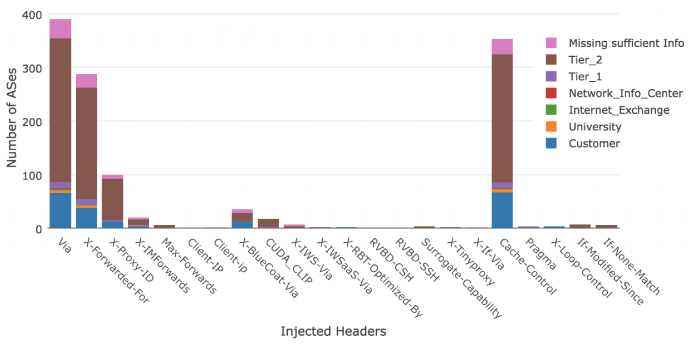

Fig. 5: AS Classification of Injected Request Header

or its network. In Table V, we list a set of headers for which we could guess the purpose. We also provide the name of the networks from which the requests come, and the countries of these networks.

TABLE V: Request Headers Exposing Mobile Network and Device Information.

Injected Header	**# of ASes**	**# of countries**
x-gateway	1 (O2-ONLINE)	1 (GB)
o2gw-id	1	1
X-Nokia-Gateway-Id	1 (MIRS)	1 (IL)
X-Nokia-MaxDownlikBitrate	1	1
X-Nokia-MaxUplinkBitrate	1	1
X-Nokia-BEARER	1	1
X-Roaming	1 (MTNSS)	1 (SS)
X-RAT-TYPE	1	1
MSISDN	2 (ZAIN, Jordan Tele)	2 (IQ,JO)
msisdn	2 (AXIATA,globacom)	2 (BD,NG)
mpesaMsisdn	1 (VODAFONE)	1 (RO)

We observe that some headers are used to expose information about cellular networks and the mobile devices that use these networks. For example, **x-gateway** and **o2gw-id** are specially used in the O2 (GB) mobile network, to carry the ID and location of gateways [15]. **X-Roaming** and **X-RAT-TYPE** are both injected to describe the radio technology used in a particular mobile network.

From either the name of the header, or the network where the middlebox is located, it is obvious that the headers shown in Table V are used for mobile devices and mobile networks. Unexpectedly, looking at the content of the headers reveals information about the mobile users, which is visible to the server-side. Unfortunately, and despite the huge number of vantage points provided by Luminati, we only observed this in 8 ASes, 4 of them could be classified. Two of them are customer networks, and the other two are Tier-2 networks. As much as Luminati provides vantage points at the edge of the Internet, few of these are actually located inside mobile networks, at least as seen from modified HTTP headers. Our results suggest that relying on vantage points located inside mobile networks is necessary to better sample middleboxes deployed across the Internet.

3) Injected Request Headers: The remaining injected headers are either non-standard, or have names or values that make it challenging, though sometimes possible, to guess the purpose of the HTTP header or the function of the corresponding middlebox. We list these on Table VI. We provide the name, country of the network (when found) and example values.

Some injected headers nicely exemplify specific types of middlebox interference, such as URL filtering, real-time content filtering, or the specific networks where the middleboxes are deployed, e.g., mobile networks or cloud providers. For example, the **x-up-vfza-id** header is added by the VODACOM network, revealing the ID of the upstream gateway inside the mobile network. The headers with the prefix **X-TMCE** are added by Trend Micro middleboxes, and from looking at the AS number of the Luminati proxy, these headers are injected for the IWSaaS service inside the Amazon cloud platform. Similarly, the headers containing **FFI** and **HCF** are likely to be added by specific vendor devices. The **X-FCCKV2** header is added by middleboxes running the Fortinet software for traffic filtering [6]. Finally, despite its rather generic name, the **Server-Slot** header is added by a middlebox inside the French cloud computing company OVH. OVH offers VPS, dedicated server and other web services, suggesting that the original request went through some web host running inside a cloud instance before reaching our target server.

4) Summary: Overall, our results from HTTP request header manipulation demonstrate the variety of different middleboxes deployed inside today's networks, and their many purposes and behaviors. From injected HTTP headers inside requests, we found that proxy and caches are the prevalent type of middlebox, and these are deployed world-wide, more than 100 countries. Despite relying on the Luminati probing infrastructure that does not particularly sample well mobile networks or cloud providers, we still found evidence of middleboxes in these networks. Finally, we observed multiple instances of vendor-specific middleboxes, which define their own HTTP headers, or of service-specific behaviors outside proxying and caching.

B. Response Manipulation

In this section, we explore HTTP response header manipulation, as well as web page blocking specifically. Before going into details of the response header manipulation, let us mention that most of the middlebox interference observed on the downstream part (by the client receiving the response) overlaps with the middlebox functions observed in the upstream direction. As previously mentioned, this makes sense as middleboxes typically are stateful devices, that see both directions of the traffic. However, this does not imply that HTTP header manipulation will take place in both directions for a given flow, or that the same headers will be affected. Therefore, one cannot expect that HTTP header manipulation in the two directions of the traffic will be similar, but at best consistent.

1) Response Header Injection: Expectedly, similar to the case of HTTP request headers, most instances of injected

TABLE VI: Remaining Injected Request Headers.

Injected Header	AS Num/ISP	Country	Note
x-up-vfza-id	1 (VODACOM-AS)	1 (ZA)	x-up-vfza-id: 65501
x-subscriber-info			x-subscriber-info: 10.139.195.196
cli	1 (QA-ISP)	1 (QA)	cli: 97433872509
imsi			imsi: 427012926009698
X-TMCE-GUID			X-TMCE-GUID: 48c1b6b0-4a2f-11e6-9c7d-0a44fff0175
X-TMCE-Token	1 (Amazon.com, Inc)	1 (JP)	X-TMCE-Token:48c1b6b0-4a2f-11e6-9c7d-0a4fff0175fc0faa422e1e04e.....
X-TMCE-User			X-TMCE-User: %40ce-ac7caab8-74df-4bc4-ae2d-e71a515dc0d
FFIClient			FFIClient: True
FFI-Authenticate	1 (NL-SOLCON)	1 (NL)	FFI-Authenticate: e78d964b-99db-4c70-88c5-3c927bb888a3
FFI-AuthenticateUser			FFI-AuthenticateUser: enno
FFI-UrlToFilter			FFI-UrlToFilter: http://shanluminati.com/?TYPE1_nl_21408
HCFVer	1 (TTNET)	1 (TR)	HCFVer: 3.7.18
HCFType			HCFType: server
X-FCCKV2	2 (ENERGOTEL,TTSLMEIS)	2 (SK,IN)	X-FCCKV2: GAJ3kZcRPNFiiMihhS2K+3EH0ofDY3+IbjlTCQ=
X-Bloxx-Result	1 (DATAWEB B.V)	1 (NL)	X-Bloxx-Result: [201, 203, 250, 251, 254, 255, 260, 261, 266, 267, 401, 425, 3009]
Server-Slot	1 (OVH)	1 (FR)	Server-Slot:ovh01FR.openvpn.wifiprotector.com_0
Referer	2 (NHN-AS,CHINA-UNICOM)	2 (KR,CN)	Referer: http://www.baidu.com/s?wd=www
X-delete-header	1 (CHINANET-BACKBONE)	1 (CN)	X-delete-header: gzip
Accept-Xncoding	1 (Bezeqint Internet Backbone)	1 (IL)	Accept-Xncoding: gzip
NCLIENT50	2 (VIA-NUMERICA,Hanyang University)	2 (FR,KR)	NCLIENT50: NCLIENT50
serialnumber	1 (INFOCLIP-AS)	1 (FR)	serialnumber: V2401625

response headers relate to proxying and caching, such as the cache hit record, the age for the cached copies and proxy connection status.

As shown in Table VII, **X-Cache** is the most frequently added response header, observed from 519 ASes across 105 countries[4]. The next most popular, **X-Cache-Lookup** is observed in 401 ASes, nearly 4% of all ASes we observe. Both of them are used to handle cache implementation details.

Surprisingly, we find the header **Set-Cookie** injected in some of our responses, while the server should be adding it, not a middlebox. Although we could not identify the host that actually sets these cookies, the injection implies the existence of a third-party server (or a middlebox) responsible for such an injection. Though we do not see the third-party actually tracking the browsing behavior of the client, the existence of such a third-party constitutes a privacy risk for end-users who are unlikely to be aware of its presence.

Compared to the injected request headers, we see less information about the unique user or gateway is injected in the response headers by the middleboxes. We observe 12 injected request headers that carry the information about the original user (private IP address) and the name or identification of proxies. Only two injected response headers record the cache hit results, carrying information about caches on the path. Although upstream and downstream traffic flows are likely to cross the same middleboxes, the middlebox interference we observe in both directions of the traffic is different. More private information about subnets or clients is added to requests compared to responses.

[4]Different from the case of requests, for responses we rely on the IP address of the country peer to infer the AS number and country of this header modification.

2) Unidentified Response Header: Similar to the request header situation, Table VIII shows the non-standard injected response headers. Again, in such cases we need to guess the purpose of the header. From our inference, it appears that most of these injected headers carry information related to content filtering and identification of middleboxes in different networks. However, we did not find any specific network function that would generally apply in these cases. For instance, **X-IS-ELAPSED** and **X-IS-FILTER** are injected in the same request, but from the values of these two headers we could not infer their function. From their name, we guess they are likely to be injected for filtering. Headers such as those with the **X-Nokia** prefix, or **X-Android**, are injected by the Android operating system, and therefore related to middleboxes located in wireless or mobile networks. The **Client-Date**, **Client-Peer** and **Client-Response-Num** headers are injected by SmarTone, the mobile network operator in Hong Kong. This shows that consistently with the upstream case, we see evidence of middleboxes in mobile networks from the downstream direction of the traffic.

3) Response Header Modification and Removal: For response headers, we also observe header removals (Table X) and value modifications (Table IX). Though we do not have explicit evidence about the type of middlebox in these cases, a large portion of the ASes for these headers overlap with those involved in the **Via**, **Cache-Control** and **X-Forwarded-For** headers in the requests. For example, as shown in Table IX, 77% of ASes for which **Accept-Range** modifications occur overlap with the ASes involved in request header injection. This suggests that these modifications and removals are actually done by the same middleboxes in both directions.

Overlapping ASes also give us the opportunity to look at

TABLE VII: Response Header Injection.

	Injected Header	# of ASes	# of countries	Note
Cache-Related	X-Cache	519	105	X-Cache: MISS from localhost
	X-Cache-Lookup	401	99	X-Cache-Lookup: MISS from localhost:3128
	Age	216	53	Age: 0
	Cache-Control	206	76	Cache-Control: max-age=0,must-revalidate,no-cache,no-store
	X-CFLO-Cache-Result	48	5	X-CFLO-Cache-Result: TCP_MISS
	X-Loop-Control	22	2	X-Loop-Control: 5.202.228.198 179F973C1B7F69B3B4D758538F3616B8
	X-Cache-Full	11	1	X-Cache-Full: MISS from myauth.pirai.rj.gov.br
	Vary	7	6	Vary: *
	X-Cache-Debug	1	1	X-Cache-Debug: TCP_MISS/NODNS-IIP/-
	SPINE-CACHE	1	1	SPINE-CACHE: MISS
	ANIS-CACHE	1	1	ANIS-CACHE: MISS
Proxy-Related	Proxy-Connection:	128	52	Proxy-Connection: Keep-Alive
	X-Cnection	23	7	X-Cnection: close
	X-OSSProxy	19	16	X-OSSProxy: OSSProxy 1.3.337.376 (Build 337.376 Win32 en-us)(Apr 22 2016 15:45:25)
	X-Squid-Error	9	6	X-Squid-Error: ERR-READ-ERROR 104
Third Party Server	Set-Cookie	2	2	Set-Cookie: xodbpb=; Path=/; HttpOnly

TABLE VIII: Injected Response Headers Requiring Inference.

Injected Header	# of ASes	# of countries
X-IS-ELAPSED	2	1
X-IS-FILTER	2	1
X-Android-Selected-Protocol	1	1
X-Android-Response-Source	1	1
Client-Date	1	1
Client-Peer	1	1
Client-Response-Num	1	1
X-Bst-Request-Id	3	4
X-Bst-Info	3	4
X-WS-PAC	3	4
Warning	14	11
Mime-Version	11	8
Location	10	9
Content-Language	6	5
X-Vitruvian	6	5
X-TurboPage	4	5
Refresh	2	1

cases where the ASes from the requests and responses differ. Indeed, when the IP address seen in the request received by the server differs from the IP address seen in the response (identifying the country peer), this means that the former IP address belongs to a TCP-terminating middlebox. We therefore count such IP addresses (1025), ASes (168), and countries (55) where these are located. Unfortunately, these statistics provide us only with a very poor lower bound on the number of middleboxes and networks observed, compared to the evidence from the HTTP header manipulation. Indeed, from the sheer HTTP manipulation we observed, we found evidence of middleboxes in 1011 ASes from the requests, and in 1023 ASes for responses.

Some response header modifications and removals may affect the end-to-end performance. For example, some proxies modify or remove the value of the **Accept-Ranges** header, to disable byte serving. As byte serving allows the server to partially deliver the content, modifying the value of **Accept-Ranges** can very well affect the content transfer. Also, the removal of the **Last-Modified** and **Etag** headers may affect the updating of cached copies.

TABLE IX: Modified Response Headers (with AS overlap).

Modified Header	# of ASes	# of overlap ASes	# of countries
Content-Length	191	108 (57%)	75
Accept-Ranges	61	47 (77%)	38
Content-Type	37	20 (54%)	24
Server	26	15 (58%)	17

TABLE X: Removed Response Headers (with AS overlap).

Removed Header	# of ASes	# of overlap ASes	# of countries
Last-Modified	143	84 (59%)	65
Accept-Ranges	107	64 (71%)	50
Content-Length	85	52 (51%)	36
Etag	73	42 (58%)	39
Server	33	21 (64%)	20

4) Summary: All in all, the manipulations of HTTP responses confirms the diversity of network functions played by today's middleboxes. Similar to the case of request headers, most of the injected response headers are added by proxies and caches. As shown in Figure 6, the classification of ASes which inject new headers inside responses is quite similar to those that do so on the requests. Although the types of injected headers are different between requests and responses, the consistency in the trends in both directions of the traffic possibly indicate that the same middleboxes indeed affect both directions of the traffic. From the downstream part of the traffic, we observed header manipulations that may potentially negatively impact end-to-end performance. Finally,

we also observed instances of page blocking, exposing the URL filtering function of middleboxes.

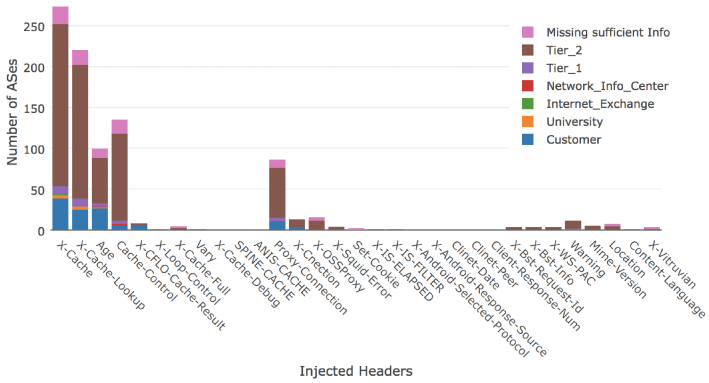

Fig. 6: AS Classification of Injected Response Header

V. Conclusion and Future Work

Middleboxes have become more and more important across a variety of networks. Unfortunately, middleboxes break the end-to-end model, in order to provide network functions to improve user security and performance. However, they are complex and interfere with traffic flows at different layers, sometimes negatively affecting end-to-end performance.

In this paper, we studied the presence of multiple types of HTTP-interfering middleboxes in today's networks. We leveraged the large number of vantage points provided by *Luminati*, launching crafted HTTP requests and responses, using a client-server methodology. We detected middlebox interference in more than 1000 ASes. We observed middlebox interference happening in both directions of the traffic, sometimes happening in mobile/cellular and cloud providers.

We observed a large number and variety of injected headers, demonstrating the prevalence of HTTP-interfering middleboxes across the Internet. The main middlebox functions identified from these interactions are proxying, caching, filtering and NAT. Despite the limited number of instances, we observed HTTP header manipulation done by mobile networks and cloud providers.

Our work provides evidence for the widespread deployment of middleboxes in the Internet. Thanks to *Luminati*, we have observed (a lower bound on) the significant scale at which these interactions happen. However, as *Luminati* controls the country peers, we are limited in our sampling of these country peers. We argue that further work is needed to better estimate the sheer number of middleboxes deployed in the Internet, and their actual impact. In our future work, we plan to make use of the diverse country peers of *Luminati* and our client-server methodology, to count and locate middleboxes. This is where we believe that the seminal work from Tracebox will deliver its true potential. Further, HTTPS plays an increasingly large role in today's Internet. Therefore, we will extend our methodology understand middlebox interference on HTTPS.

Acknowledgment

This work was funded by the China Scholarship Council.

References

[1] S. W. Brim and B. E. Carpenter, "Middleboxes: Taxonomy and Issues." RFC 3234, Mar. 2013.

[2] 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 *ACM SIGCOMM 2012 Conference, SIGCOMM '12, Helsinki, Finland - August 13 - 17, 2012*.

[3] "Guide to intrusion detection and prevention systems(idps)." http://ecinetworks.com/wp-content/uploads/bsk-files-manager/86_SP800-94.pdf.

[4] N. Freed, "Behavior of and Requirements for Internet Firewalls." RFC 2979, Nov. 2015.

[5] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013*.

[6] N. Weaver, C. Kreibich, M. Dam, and V. Paxson, "Here be web proxies," in *Passive and Active Measurement - 15th International Conference, PAM 2014, Los Angeles, CA, USA, March 10-11, 2014, Proceedings*.

[7] "Network caching technologies." http://docwiki.cisco.com/wiki/Network_Caching_Technologies#Proxy_Servers.

[8] Z. Wang, Z. Qian, Q. Xu, Z. M. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, Canada, August 15-19, 2011*.

[9] G. Aceto and A. Pescapè, "Internet censorship detection: A survey," *Computer Networks*, vol. 83, pp. 381–421, 2015.

[10] M. Dischinger, A. Mislove, A. Haeberlen, and P. K. Gummadi, "Detecting bittorrent blocking," in *Proceedings of the 8th ACM SIGCOMM Internet Measurement Conference, IMC 2008, Vouliagmeni, Greece, October 20-22, 2008*.

[11] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend tcp?," in *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2-, 2011*.

[12] "The collateral damage of internet censorship by dns injection," *SIGCOMM Comput. Commun. Rev.*

[13] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: illuminating the edge network," in *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia*.

[14] D. Naylor, K. Schomp, M. Varvello, I. Leontiadis, J. Blackburn, D. R. López, K. Papagiannaki, P. R. Rodríguez, and P. Steenkiste, "Multi-context TLS (mctls): Enabling secure in-network functionality in TLS," in *Proc. ACM SIGCOMM*, 2015.

[15] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, and V. Paxson, "Header enrichment or isp enrichment?: Emerging privacy threats in mobile networks," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes*, HotMiddlebox '15.

[16] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, N. Weaver, and V. Paxson, "Beyond the radio: Illuminating the higher layers of mobile networks," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*.

[17] "Open observatory of network interference." https://ooni.torproject.org/nettest/.

[18] G. Tyson, S. Huang, F. Cuadrado, I. Castro, V. C. Perta, A. Sathiaseelan, and S. Uhlig, "Exploring HTTP header manipulation in-the-wild," in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*.

[19] S. Huang, G. Aceto, F. Cuadrado, S. Uhlig, and A. Pescap, "Detecting middleboxes interference on applications," in *Processdings of ACM CoNEXT, Heidelberg, Dec 1-4 2015*.

[20] X. Dimitropoulos, D. Krioukov, G. Riley, and k. claffy, "Revealing the Autonomous System Taxonomy: The Machine Learning Approach," in *Passive and Active Network Measurement Workshop (PAM) Adelaide, 2006*.

[21] "Interscan web security." http://www.trendmicro.co.uk/products/interscan-web-security/index.html?