Where The Light Gets In: Analyzing Web Censorship Mechanisms in India

Tarun Kumar Yadav[*]
IIIT Delhi, India
tarun14110@iiitd.ac.in

Akshat Sinha[*]
IIIT Delhi, India
akshat14132@iiitd.ac.in

Devashish Gosain[*]
IIIT Delhi, India
devashishg@iiitd.ac.in

Piyush Kumar Sharma
IIIT Delhi, India
piyushs@iiitd.ac.in

Sambuddho Chakravarty
IIIT Delhi, India
sambuddho@iiitd.ac.in

ABSTRACT

In this work we present a detailed study of the Internet censorship mechanism in India. We consolidated a list of potentially blocked websites from various public sources to assess censorship mechanisms used by nine major ISPs. To begin with, we demonstrate that existing censorship detection tools like OONI are grossly inaccurate. We thus developed various techniques and heuristics to correctly assess censorship and study the underlying mechanism used by these ISPs. At every step we corroborated our finding manually to test the efficacy of our approach, an exercise largely ignored by several others. We fortify our findings by adjudging the *coverage* and *consistency* of censorship infrastructure, broadly in terms of average number of network paths and requested domains the infrastructure censors. Our results indicate a clear disparity among the ISPs, on how they install censorship infrastructure. For instance, in Idea network we observed the censorious middleboxes in over 90% of our tested intra-AS paths, whereas for Vodafone, it is as low as 2.5%. We conclude our research by devising our own novel anti-censorship strategies, that does not depend on third party tools (like proxies, Tor and VPNs *etc.*). *We managed to access all blocked websites in all ISPs under test.*

KEYWORDS

Censorship, OONI, India

1 INTRODUCTION

Free and open communication over the Internet, and its censorship, is a widely debated topic. It is not surprising that an overwhelming majority of prior studies on censorship activities and their mechanism, primarily center around overtly censorious nations like China [27, 34, 40, 52] and Iran [23]. Most of these studies involve reporting censorship activities, with some categorically focusing on the in-depth description of the actual censorship techniques and mechanism that are employed by such nations *i.e.* — describing the network location of the censorship infrastructure — what triggers them — and how are clients notified of such filtering.

Through our studies over the past few years, we discovered that even democratic nations like India, have slowly, and rather covertly, evolved an infrastructure for large-scale Internet censorship, involving several privately and federally operated ISPs. India's Internet censorship policies have remained arbitrary (at best ambivalent)[1]. Over time several networks have upped their barriers against users accessing sites, which the administration "believes" to be "unfit for consumption", resulting in enough citizens facing web censorship.

Our previous work [21] emphasized on hypothetical scenarios of potential (future) large scale censorship (or surveillance) by the state. A mere preliminary report was also presented highlighting the inconsistent web censorship policies amongst ASes.

We thus formally approached the authorities, filing a *Right to Information* [15] request (RTI), inquiring about the policies and mechanism the government uses to block content. In response, the authorities shared that while the censorship policies are confidential, the onus of implementing them lied with the individual ASes who could employ any mechanism they chose.

Ambiguous answer from authorities motivated us to conduct our own detailed analysis of the different censorship mechanisms the major network operators of the country employ. We began our research by compiling a corpus of about 1200 potentially blocked sites (PBWs), curated from various Internet sites (*e.g.* Herdict [6], Citizen Lab [10]). Thereafter, we obtained network connections for nine popular ISPs — Airtel, Idea, Vodafone, Reliance Jio, MTNL, BSNL, Siti, Sifi and NKN.

For gauging censorship, we ran the popular censorship assessment tools like OONI [18] on clients hosted in these networks. OONI runs two sets of tests, one at the client and other at their remote control site (assumed to be unfiltered). A mismatch between the results signals potential censorship. However, our initial tests yielded considerably high false positives and negatives when tested for different ISPs. For instance, in Airtel, we obtained a false positive rate of ≈ 80% and a false negative rate of ≈ 11.6%.

We thus decided to devise our own analysis techniques. We began by observing the ensuing network connection traffic between our client and the censored site. In one particular ISP network, we observed that whenever the client connected to the censored site, it received a valid HTTP response bearing a statutory censorship

[*] All the three authors have equal intellectual contribution.

[1] For *e.g.* in August 2015, the government issued orders to block 857 websites, but later backtracked under public outcry [13]

IMC '18, October 31-November 2, 2018, Boston, MA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5619-0/18/10…$15.00
https://doi.org/10.1145/3278532.3278555

notification with appropriate sequence number and bits (*e.g.* FIN, RST) in the TCP headers that enforce the client to disconnect with the server. Eventually, the actual response from the censored site also arrives, but by then the connection is already terminated, and the packet is discarded.

All such protocol exchanges hinted toward the presence of malicious network elements (we collectively call *middleboxes*) that snoop (or intercept) users' traffic and upon observing requests to filtered sites, inject the aforementioned crafted packets to censor traffic.

To identify the network location of such censorship infrastructures, we devised a technique which we collectively call *Iterative Network Tracing* (INT), that works on the principle used by `traceroute`. It is quite similar to those proposed earlier by Xu *et al.* [52] and involves sending web requests to censored sites but with increasing IP header TTL values. These messages encounter middleboxes, that are triggered upon the arrival of request to the censored sites.

Using our approach, INT, and various heuristics which we developed after observing peculiarities of censorship techniques, we conducted an investigative study of various censorship mechanism, employed by major ISPs in the country. Our research engages long-term data collections to answer the following questions:

- What sequence of protocol messages triggers censorship?
- Exactly what techniques are employed by ISP networks to filter users' requests to censored sites?
- Approximately what fraction of network paths are impacted by these censors?
- Is censorship uniform and consistent across the various ISPs? More specifically –
  - Do various ISPs block the same set of sites?
  - Do various censorship devices of an ISP (*aka* the *middleboxes*) block the same set of sites?
- How hard or easy is it to bypass such censorship mechanism?

Unlike several previous efforts, that directly draw conclusions based on the results generated by their respective techniques [30, 31, 44] ours, at every possible step, involves corroborating the results via manually connecting to the sites and inspecting the results.

The key contribution of our research efforts, spanning over 18 months, involves detailed answers to all the aforementioned questions. Our findings show that four of these ISPs *viz.* Airtel, Vodafone, Idea and Reliance Jio (potentially carrying a large fraction of network traffic [21]), employ stateful inspection of HTTP requests alone to censors access. For some ISPs, like Idea Cellular, we detected the presence of censorship infrastructure in over a very large fraction (>90%) of the intra-AS network paths.

Others, *viz.* BSNL and MTNL, prime government operators, poison DNS responses for censored sites. In our measurements, we found about 600 censorious DNS resolvers spread across the two ISPs.

Traffic of non-censorious ISPs transiting the censorious ones often gets inadvertently filtered. We observed such phenomenon for various non-censorious ISPs in India. For example, censorship in Vodafone network causes collateral damage to NKN, an otherwise non-censorious educational network.

Through our detailed explorations, we discovered network middleboxes that either intercept traffic (like trans-proxies) or merely snoop on users traffic and sends back specially crafted messages disconnecting the client–server connection. While a vast majority of previous efforts, like [35, 42, 49] report the latter, we discover the presence of intercepting middleboxes (similar to Syria [25]) in one of the ISPs.

Finally, while overtly censorious nations have evolved mechanisms to counter censorship circumvention (proxies and VPNs) [17, 30], we demonstrate simple, yet effective, techniques that can be used to bypass censorship, that do not rely on such proxies, and may go undetected by such censors. Our approach relies on identifying the packets generated from the middleboxes and filtering them at the client, or sending crafted requests that are not detected by the middleboxes, but are correctly recognized by the server. This is harder for ISPs to identify which work by restricting access to anti-censorship infrastructure. Moreover, efforts to retrofit the solution into existing censorship middleboxes may incur high costs on the part of the middlebox manufacturers and the ISPs, without factoring in downtimes and potential failures.

2 BACKGROUND AND RELATED WORK

According to ONI report, India is among the list of countries that restricts the Internet content and ranks India as “Partly Free” [9]. Internet censorship in India can be traced back to the year 1999, where website of the popular Pakistani daily newspaper ‘Dawn’ was blocked from access within India, immediately after the Kargil War [14]. Since then, there are numerous instances of Internet censorship recorded [9] by the orders of the government to an extent of Internet shutdowns. In the year 2015, there were at least 22 instances of Internet shutdowns in different parts of the country [50]. And later in the same year, ISPs were directed by the government to block 857 websites, on the basis of restricting access to pornographic content [3].

Very recently transparency reports published by Facebook [4] and Google [5] confirm that censorship in India is on the rise. It indicates that there were a total of 21 instances of complete Internet shutdowns and 1, 228 instances of content removal by Facebook because a majority of content restricted was alleged to violate local laws relating to defamation of religion and hate speech.

Thus, we conducted a detailed study of web censorship trends pertaining to Indian ISPs, specifically aimed to explore the censorship mechanism and its associated infrastructure deployed in the country. We thus begin by discussing important studies in the area of Internet Censorship, primarily reporting the *type* and *mechanism* of censorship. Zittrain [55] in his seminal analysis of censorship observed IP, keyword and DNS filtering in China. Later many studies focused on censorship specific to particular countries for *eg.*, China [27, 51], Pakistan [42], Italy [19], Greece [48] Iran [23], Egypt and Libia [28] etc. Verkamp *et al.* [47] extended this work by deploying clients in 11 countries to identify their network censorship activities encompassing IP and URL filtering, keyword filtering and DNS based censorship *etc.* Gill *et al.* [33] rather than deploying clients, used data gathered by the OpenNet Initiative to detect censorship in 77 countries.

Dalek *et al.* [29] used data from Shodan [16] to identify URL filtering products deployed across many countries including Qatar, Yemen, Saudi Arabia and India. For large scale detection of censorship across multiple countries, there are several projects which provide tools to determine censorship policy: HerdictWeb [54], CensMon [45], Encore [24], OONI [18] and Augur [43].

However, a significant portion of censorship literature focuses only on the People's Republic of China — Great Firewall of China (GFW) [27, 30, 31, 38, 41, 49, 52, 53, 55]. Winter *et al.* [17] studied how DPI-enabled routers detect Tor bridges based on specific TLS cipher suits. Others such as [39] reported that China is heavily contributing towards collateral damage by DNS filtering. Khattak *et al.* [37] observed that GFW operates similarly to NIDS and found exploitable flaws in state management of GFW. Later Wang *et al.* [49] reported that GFW has evolved over a period of time and previous solutions to bypass it [37] are now ineffective. They proposed a novel tool, INTANG, to bypass GFW using carefully crafted packets, without relying on proxies or VPNs.

In the year 2017, we [21] explored that Indian ISPs have incoherent censorship policies and they implement their own content filters resulting in dramatic differences in the censorship experienced by customers. Also, we studied the hypothetical scenario — assuming in future, the Government of India plans to implement strict censorship what would be the probable 'key points' for them to place the filters, for different censorship mechanisms *viz.,* IP filtering, Prefix Hijack, DNS filtering etc.

In this research, we rather carried out a comprehensive study on the '*present*' Internet censorship implementation in India, a vital study missing in our previous analysis. Thus, in this work we developed our own heuristics, with which we tried determining the type of censorship mechanism involved (and in some cases the approximate location of the censorship infrastructure as well). At every step we validated our results against the ground truth, an effort largely ignored by several others in the recent and distant past.

3 DATA COLLECTION AND APPROACH

For our research, we curated a list of potentially blocked websites (which we refer to as PBW) from different sources which include Citizen Labs [10], Herdict [6] and various past government and court orders of the country [11]. The list includes a total of 1200 websites which we consider to be sensitive (and thus potentially censored). They span across 7 major categories *viz.,* escort services, pornography, music, torrent sites, politics, tools and social networks.

We commenced our research by using the already popular censorship detection tool, OONI [**?**]. A client which intends to detect possible instances of censorship (at different layers of network stack) installs OONI probe. This fully-automated tool, reports the blocked websites and possible underlying censorship mechanisms being used. After running OONI from five different vantage points we observed that it results in high false positives and negatives. Thus, we created our own scripts to detect Internet censorship in India.

Popular ISPs	**Censorship Type**			
	Total	DNS	TCP	HTTP
MTNL	0.57, 0.42	0.44, 0.10	0, 0	0.60, 0.64
Airtel	0.19, 0.11	0, 0	0, 0	0.19, 0.11
Idea	0.57, 0.62	0, 0	0, 0	0.57, 0.62
Vodafone	0.69, 0.82	0, 0	0, 0	0.70, 0.78
Jio	0.34, 0.15	0, 0	0, 0	0.36, 0.14

Table 1: Accuracy of OONI: Precision and recall values, measured in various ISPs.

3.1 The OONI tool

Open Observatory of Network Interference (OONI) [**?**], is an open source tool under the Tor project and is designed to detect censorship.

We executed OONI on five popular ISP networks, using the PBW, and recorded the results. To corroborate our findings we also manually checked the sites that were reported by OONI as being censored. To our surprise, we observed very few true positives. An exceedingly large number of sites which were reported as being censored were however easily accessible.

Table 1 summarizes our findings. The rows represent the ISPs, columns correspond to the type of censorship reported by OONI, and each entry is a 2-tuple (P,R) representing the precision and recall.

To explain our results better we use the example of data gathered for Airtel (a major ISP of the country[2]). The OONI tool reported that about 78 sites (B_O) were being blocked by the Airtel. Upon manual inspection we observed this number to be much higher, *i.e.* 133 (B_M). Only 15 websites ($B_O \cap B_M$) that were actually being censored were also correctly detected by OONI. This provides us with a precision of 0.19 ($|B_O \cap B_M|/|B_O|$) and a recall of 0.11 ($|B_O \cap B_M|/|B_M|$). Similar results were observed for other ISPs as well.

Such low values of precision and recall can be attributed to the fact that OONI tool uses rudimentary approaches to detect potential censoring activities. For instance, while detecting DNS filtering, it compares the IP address of a given host name returned by Google DNS resolver (which they assume to not be tampered) with the IP address mapped to that website by the client's ISP. If the two IP addresses of the same website are different they assume it to be censorship. But, in many cases differences in URL resolution is likely an artifact of network hosting architectures (*e.g.* CDNs).

Also, while detecting HTTP filtering, OONI sends an HTTP request to a given website over the network where the client (running the OONI probe) is hosted. Following that, the same request is sent from the control server (of OONI). HTTP responses obtained from these requests are compared (based on a threshold) and the website is assumed to be filtered if the responses differ. However, while conducting our experiments, we observed that in spite of observing difference in HTTP responses, that OONI uses to report censorship, the websites were actually not blocked. We explain the reasons for incorrect reporting in detail in section 7.

Thus, for our research, we abandoned OONI and created our own *semi-automated* scripts to record the censorship instances by various ISPs across India. For instance, similar to OONI, we tried sending GET requests to PBWs from the client in test ISPs and

[2] In terms of network paths it intercepts [21]

through Tor. If the difference in responses was less than a certain threshold we considered them *non-censored*, otherwise we manually inspect the responses further, unlike OONI, which directly flags them as censored. For instance, in Airtel network, we observed that for 390 PBWs, the difference between the contents of HTTP responses was more than 30%. On manually verifying (*i.e.,* identifying the censorship message in HTTP response) we confirmed that 156 of those (*i.e.,* 40%) were actually blocked. We repeated the same experiments for several other ISPs and found that 30 − 40% of the websites which would have been flagged as censored by OONI were actually false positive. Thus, we selected 0.3 as the threshold for our experiments (explained in detail in subsection 3.4).

We now present our approach for determining the type of censorship (DNS, TCP/IP and HTTP) and mechanisms behind them.

3.2 DNS Blocking

I. Background: For ordinary netizens, DNS resolution is the primary step for accessing any website. URL entered by such netizens is first resolved to its associated correct IP address. Thus, invariably censors exploit this step, and often return an incorrect IP address, resulting in website's unavailability.

DNS based blocking can be achieved by (1) *DNS poisoning* [46]— whereby a corrupt(-ed) resolver replies with the incorrect IP for specific DNS queries. (2) *DNS injection* [22] — where some middlebox between the client and resolver intercepts the DNS query and deliberately responds with a forged response bearing an incorrect IP address.

II. Identifying sites filtered using DNS requests: In order to identify DNS filtering by ISPs, we selected PBWs that could otherwise be successfully resolved via Tor circuits (ending in exit nodes in non-censorious nations). Thereafter, we attempted to resolve these PBWs through ISPs under test.

A URL might resolve to multiple IPs everytime a resolution is attempted. Further, these responses may also differ when resolution is attempted via Tor and from the ISPs under test (due to reasons such as CDN based hosting). Thus, URL resolutions, attempted via Tor and the tested ISPs, resulting in an overlapping set of IPs, were considered to be uncensored and eliminated from further inspection.

In order to ascertain DNS filtering on the remaining URLs (among the PBWs), we first tested the following intuition: an ISP may deliberately resolve multiple blocked websites to some unique IP addresses. We went ahead and performed DNS resolution for all these remaining URL from the ISP under test.

Invariably, we found several blocked websites resolving to the same IP address belonging to an ISP under test. In general, several sites may resolve to the same IP address, an artifact of modern commercial hosting. Thus, before conducting our measurements we eliminated sites that were actually hosted with the same IP.

Additionally, in several cases the blocked URLs also resolved to bogon IP [1] addresses.

Hence, we applied the following heuristic to decide if the DNS responses were seemingly manipulated by the censor.

1. *Resolved IPs belong to the same AS that hosts the client:* None of the PBWs were hosted in the clients' AS. Thus if any of the resolved IPs belong to the clients' AS (the one under test), the AS is considered censorious and the corresponding URL is marked censored.
2. *Resolved IPs are Bogons*: If any of the resolved IPs is a bogon [1], we consider the AS to be censorious and the URL as being censored.

We applied these heuristics and identified the manipulated IP address responses and removed the corresponding URLs from further analysis. In order to confirm that *only* the aforementioned strategies, and none others, are employed by the tested ISPs, we sent HTTP request to the remaining IPs (obtained initially when resolution was attempted from the tested ISPs), via Tor and manually inspected the corresponding response to confirm that they were as expected.

III. Identifying DNS filtering mechanism: After identifying the set of websites which are censored due to DNS filtering, we intended to identify the mechanism behind the blocking. To that end, we began by identifying all open DNS resolvers of the ISP under consideration. To do this we sent DNS queries requesting resolution for otherwise uncensored sites (*e.g.* our own institution's website) whose correct IP address is known beforehand, to the entire IPv4 address space of the said ISP. DNS resolvers, even if censorious but otherwise configured correctly, are expected to respond to such queries with legitimate IP addresses.

In order to identify only the *censorious resolvers*, we sent 1200 DNS queries, corresponding to the individual PBWs, to each of the DNS resolvers (in each of the individual censorious ISPs). Resolvers which responded with manipulated IP address (for any of the DNS queries), are considered to be censorious.

To determine **how and where censorship happens**, *i.e.* if censorious DNS responses were due to middleboxes or by the poisoned resolvers, we used a variant of INT (as shown in figure 1). We began by identifying the router-level path between the client and the censorious resolvers using `traceroute`. Thereafter, the client sends DNS requests (corresponding to PBWs) to *only* censorious DNS resolver by iteratively increasing IP TTL values. Identifying censorship mechanism involved checking if the responses (between the client and a PBW) arrive from any network hop other than the last one. Such responses are likely due to middleboxes, else they are due to poisoned resolvers.

In all our tests we received manipulated IP addresses from the last hop only, indicating the presence of **DNS poisoning**.

3.3 TCP/IP Packet Filtering

Network protocol header based filtering is a rather ill defined, albeit very commonly assumed, network censorship technique. It is frequently believed by netizens that ISPs filter traffic based on IP addresses and port numbers. Not surprisingly several past research efforts focused on detecting censorship where authors claimed that ISPs filter traffic based on IP addresses. To that end, they primarily relied on packet drops corresponding to TCP connection attempts [43] and claimed them to be due to IP-level censorship.

In general, such techniques may often mis-classify various kinds of systemic failures such as network congestion, outages and delays in route re-computations, as IP address based censorship. Also,

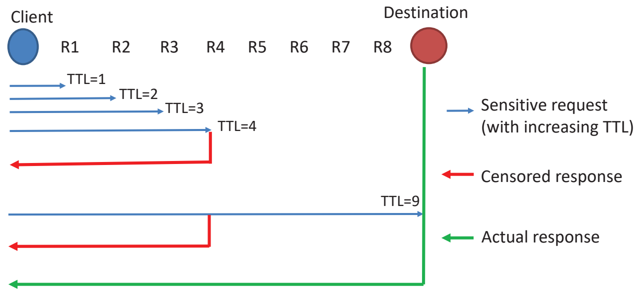

Figure 1: Iterative Network Tracer: Client sends a crafted query (DNS query/HTTP `GET` request) containing a blocked domain with increasing TTL. Censorship response is observed from the censor's network element.

unlike HTTP censorship, which often involves users receiving censorship notification packets, IP address filtering may reveal no information to the client, making it hard to distinguish from the other reasons mentioned. Such scenarios are very difficult to validate (an important, and often ignored aspect of prior research [43]).

Nevertheless, we used a straightforward approach to detect filtering based on network and transport protocol headers. We attempted TCP 3 − *way* handshakes with the PBWs. These were tunnelled through Tor circuits terminating in non-censorious countries. For those websites where the connection succeeded (via Tor), we again attempted five subsequent TCP 3 − *way* handshakes (from the ISP under test) with a delay of approximately two seconds between each of them. If it failed in *all* the attempts, it implied TCP/IP filtering. *However, in none of the tested ISPs, we ever obtained this form of censorship.*

3.4 HTTP Filtering

I. Background: HTTP filtering aims at hampering the communication between client and server by observing content of HTTP packets. The censor can achieve this type of filtering by deploying middleboxes in the network (placed between client and blocked domain).

II. Detecting HTTP filtering: In our experiments, we tested all our chosen ISPs for potential censorship using our curated list of 1200 PBWs. We began by creating Tor circuits terminating in non-censorious countries. Through these, we tried accessing all the PBWs. The retrieved contents were compared against the contents obtained when directly accessing the respective PBWs, from our clients hosted in the individual ISPs.

One may expect that in case of overt censorship the difference between the aforementioned responses to be very high (*e.g.*, >80%). However, there might be false negatives in case the differences are lower. We thus chose a relatively lower difference threshold of about 30%. Manual inspection of site content, when the differences were lower than this threshold, revealed no censorship. Whereas, when the difference was greater than 30%, we found several instances of censorship notifications. In general we observe that whatever threshold we take for measuring the difference between HTTP responses, for the cases where difference is more than the threshold, we need manual inspection. Depending upon the censorship response and ISP under test, the threshold may be adjusted accordingly.

III. Which HTTP messages trigger censorship: We began our study of determining what triggers censorship by observing the protocol messages between client and PBWs. For *e.g.*, for a client hosted in Airtel, we observed that after sending an HTTP `GET` request to a PBW (following a regular TCP 3-way handshake), a HTTP 200 OK response packet arrived, whose source IP address is that of the PBW. It had the TCP FIN bit set and payload carried the censorship notification. TCP FIN bit forced the client's browser to initiate TCP 4-way connection termination with the PBW. Eventually, the packet from the PBW *also* arrived. We were thus unsure as to what triggered censorship – requests from the client to the PBW or their responses?

In the past researchers have reported middleboxes in China that inspect both *request* from the client and *response* from the server for censoring content [26, 27]. Thus, we also required heuristics to determine if censorship was triggered by requests or by responses.

In order to distinguish between the two possibilities, we adopted the following approach. Initially, the client runs `traceroute` to obtain the number of hops (n) to the actual website. Thereafter, the client establishes a TCP connection with the website and sends two consecutive HTTP `GET` requests for the blocked website. The IP header of the first request has a TTL value of $n - 1$ and is not expected to reach the site, and thus no responses from the site are expected. Whereas, the second is sent to the site bearing a TTL value of n (and is expected to be handled like a regular request).

Depending upon what the middleboxes en route inspect, there could be three possibilities:

- Possibility 1 (Middlebox get triggered *only* by the request): Both the above requests would traverse the middlebox that would respond back with a censorship notification-cum-disconnection message.
- Possibility 2 (Middlebox get triggered *only* by the response): The middlebox would be triggered when it inspects the response messages, which happens only when the request actually reaches the site and elicits it (*i.e.* only corresponding to the second request).
- Possibility 3 (Middlebox triggered by request *and/or* response): The middlebox sends censorship notification for both the requests.

In our measurements, we observed censorship notification-cum-disconnection packet for both the requests (*i.e.* for TTL=$n - 1$ and TTL=n). This directly rules out the possibility 2, *i.e* when middleboxes are triggered *only* through responses.

For both the remaining possibilities the middleboxes could inspect both requests and responses. In order to distinguish these possibilities, we crafted our own HTTP `GET` request such that PBW interprets it correctly but *not* the middlebox. For *e.g.*, in Airtel network, merely manipulating the case of the HTTP header field `Host` and changing it to `HOST` was sufficient for the request to go undetected by the middlebox (while correctly interpreted by the PBW). We show in section 6 that for all ISPs, we managed to bypass the censorship by only modifying the HTTP header fields of the

GET request. *This confirms that middleboxes are only inspecting the requests (possibility 1) and not the responses, as otherwise, we would still be receiving censorship notifications when the responses, carrying censored content, would encounter the said middleboxes.*

IV. How* GET *request triggers the middlebox: Since middleboxes inspect the GET request for potential censorship, we intend to confirm exactly how the middleboxes get triggered. By default a regular GET request bares only the domain name along with the requested page. We first ran traceroute to obtain the number of hops to the server. Then, we crafted a GET request whose IP TTL was set to the value of penultimate hop, such that it passes the middlebox but never reaches the server. Thus, we ensured that response (if) received is from the middlebox and not the actual server.

In the payload, we fudged the domain name and its offset within the request, to determine exactly what triggered censorship. For *e.g.*, we set the HTTP Host field to that of an uncensored site, while the domain name of the censored site was positioned at a random offset within the HTTP header (say beyond the requested page indicated in the GET field). In all our tests we observed only when the Host field is set to the domain name of the censored site, we observed the censorship response. Further details of more related experiments are presented in section 5.2.

As described ahead in section 5, in three of the four ISP where we observed HTTP censorship, the middlebox responds back to the client with a variant of the aforementioned censorship notification-cum-disconnection messages. These were mostly HTTP 200 OK responses carrying the statutory censorship notification along with appropriate TCP bits enabled that force the client to terminate the connection with the PBW. They bore appropriate sequence and acknowledgement numbers (along with other protocol header information) to make them indistinguishable from legitimate packets which the client's underlying protocol stack expects, *w.r.t.* the initial TCP connection to the PBW.

In section 6, we show how we exploit this knowledge of protocol header idiosyncrasies, along with deliberate fudging of the requested domain name in the Host field of the GET requests to sidestep censorship.

V. Identifying location of HTTP middleboxes: After characterizing the blocking behavior, we intended to identify the *network location* of middleboxes *viz.,* IP address. As earlier 3.2, we first ran traceroute to determine the number of hops between the client and a PBW (to be tested). We then use INT (shown in figure 1) whereby following a regular TCP handshake to the PBW, we sent series of crafted HTTP GET request to it, with increasing TTL values until it encounters the middlebox. The middlebox, upon observing the GET request, bearing the domain name of the PBW, responds with a censorship-notification-cum-disconnection message (with TCP FIN/RST bit set). Correlating the TTL value of the request (observed by the middlebox) against the IP address hops reported by traceroute helped us identify the middleboxes.

4 EXPERIMENTAL SETUP AND ETHICAL CONSIDERATIONS

All our experiments were carefully designed to avoid any unintentional or unethical network disruptions or system downtimes and failures, of third-parties, including individuals, ISPs, institutions and governments. To perform our large-scale censorship studies, we used our own client machines hosted in various networks. For this we bought connections of about nine popular Indian ISPs. To augment our results we used hosts deployed outside India – about 50 planetlab hosts (which by default provide "sudo"-able administrative access), about ten virtual machines in various cloud hosting services and around ten more hosts placed in institutions where we had collaborators, who were kind enough to lend us their infrastructure (granting administrative access and all essential privileges to conduct our experiments).

In our initial studies, we sent traffic to a small sample of PBWs (which were curated from various sources, as already explained in section 3), from each of the clients under our control. Using pcap we passively inspected the protocol responses to determine the actual mechanism through which censorship was enforced (as explained ahead).

Further, we also conducted large-scale studies to quantify the impact of censorship. Our efforts involved sending HTTP GET requests (≈ 440 Bytes) to Alexa top-1000 sites, from the clients we controlled, at every 8–10 second intervals. This was slow enough to not impact network performance of other users.

5 EXPERIMENTAL RESULTS

In order to determine the censorship mechanism we inspected for DNS, TCP/IP and HTTP blocking for the list of PBWs in nine major ISPs of the country (as explained in section 3). For all ISPs, we found instances of DNS and HTTP filtering only.

5.1 DNS filtering

We began our study by identifying the open DNS resolvers in a chosen ISP. Thereafter applying our heuristics presented in subsection 3.2 we determined which of the 1200 curated PBWs were being censored along with their corresponding DNS resolvers.

We observed poisoned DNS resolvers in only two of the nine ISPs, *viz.,* MTNL and BSNL.

Before presenting the results, we propose two metrics to analyze the extent of DNS filtering within the ISPs:

1.***Coverage***: Ideally all DNS resolvers of an ISP must be poisoned. We define *coverage* as the fraction of all the resolvers of the ISP which are poisoned.

2.***Consistency***: Ideally the same set of sites must be blocked by all the *poisoned* resolvers of an ISP. We determined the set of filtered URLs as well, as all the resolvers that blocked them. For every filtered URL we determine the fraction of poisoned resolver blocking it. *Consistency* is the average of these fractions.

In MTNL, we found a total of 448 resolvers, out of which 383 were poisoned, *i.e.* coverage was around 77%. Whereas, in BSNL we found only 17 poisoned resolvers out of a total of 182 (a smaller coverage of around 9.3%).

The consistency of each ISP can be inferred from figure 2. Websites which are blocked in any of the two ISPs are represented on the X-axis. The percentage of resolvers blocking the website are represented on Y-axis. For the sake of preserving anonymity, we represent the sites with unique numbers, rather than actual names. It can be clearly seen from the figure that in general a single website (for *eg.,* website ID 450) is blocked by more number of resolvers in MTNL (44%) than in BSNL (6.6%). The consistency metric in MTNL (42.4%) is also higher than that of BSNL (7.5%).

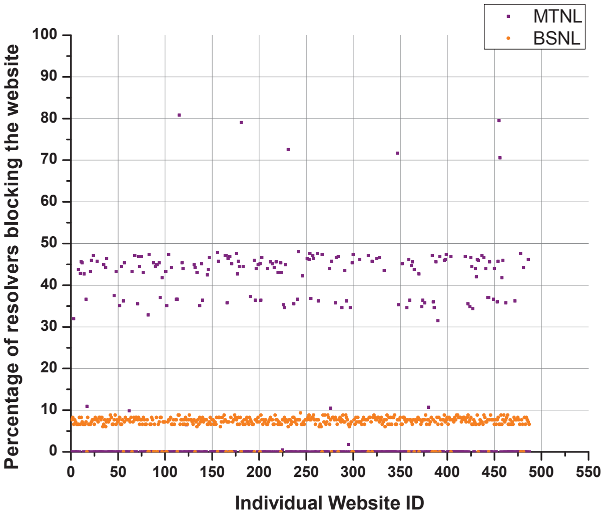

Figure 2: Consistency of DNS resolvers.

5.2 HTTP filtering

We found HTTP filtering in four out of nine ISPs. As already discussed in section 3.4, ISPs have deployed middleboxes which inspect the packets between the client and blocked websites with an intent to censor traffic.

We began by identifying all those websites, among the 1200 PBWs, that were censored by the ISP. For instance in Airtel, we observed a total of 234 websites to be censored. The corresponding number for the remaining three ISPs is presented in the last column of table 2. Using the approach described in subsection 3.4 we determined that censorship was triggered *solely* due to request and not the response.

Finally, we attempted to find the actual network location of the middleboxes with a variant of INT, involving crafted HTTP GET requests. However, we were unable to pinpoint the exact IP address of the middleboxes in most of our measurements because of anonymization by the ISP. We discuss this in detail in section 7.

We now present in the behavior of the different types of middleboxes, we identified in the wild and describe their censorship mechanisms in detail.

5.2.1 Types of middleboxes. In our experiments, we identified two kinds of middleboxes–*viz. Interceptive Middlebox (IM)* and *Wiretapping Middlebox (WM)*. IMs are akin to transparent proxies which intercept connections between the client and server and establish a new connection to the server. In our studies, we found IMs that intercept client–PBW request and respond with censorship notification messages, while dropping the actual requests.

The other, *i.e.* WMs, involve a host that is connected to an active network element via a wiretap. It receives a copy of all the packets exchanged and inspects for requests that need to be censored. Thereafter, it crafts responses and sends it back to the censor, with appropriate TCP header bits to terminate existing connections.

However, the WMs cannot outpace the client–PBW traffic flow, as they work with a copy of the packets. Thus, they are not as effective in filtering every single request with real-time efficiency, compared to IMs. For WMs, roughly in 3 out of 10 attempts to access blocked website, the middleboxes were ineffective in censoring the content. Whereas, for IMs, all such attempts were unsuccessful.

Interceptive middleboxes. We used our variant of INT (explained in subsection 3.4) to first obtain the location of middlebox in the network path intervening the client and a filtered site.

Thereafter, we sent crafted HTTP `GET` requests, bearing the censored domain in the `Host` field, with TTL values large enough to get past the network hop, corresponding to the middlebox. Regardless of further increments to this IP TTL value, we never observed the expected ICMP TTL Expired responses, but rather received the censorship notification messages, *indicating that the middleboxes might be intercepting and dropping these requests.*

In order to verify that IMs are triggered only for the blocked domains, we sent a crafted GET request where `Host` field bore a non-censored domain, while also iteratively increasing IP TTL values. Interestingly, we always received ICMP TTL Expired messages, even when TTL was large enough for the packets to transit the middlebox. This confirms that IMs only inspect `Host` field of HTTP `Get` request.

We went a step ahead and selected an array of hosts we controlled in different networks[3] outside Indian ISPs. On these machines, we hosted an ordinary webserver. From our client, hosted in the ISP under test, we created TCP connections to these remote machines. The remote host simultaneously monitored its own traffic. The client sent crafted `GET` requests with `Host` field requesting a censored domain. The destination IP address, however, was that of the remote host. Upon traversing a censorious middlebox positioned on the network path in-between, the client receives a censorship notification-cum-disconnection packet, with TCP FIN bit set. The subsequent 4-way disconnection always timed out (very likely dropped by the middlebox). Finally, the client attempted terminating the connection by sendingR a RST packet.

The remote host however receives *none* of the packets, other than the initial handshake messages and a RST packet. But, the TCP sequence number of this RST packet differed from the terminal RST packet sent by the client, thereby confirming that it was sent by a middlebox. This confirmed the presence of IMs.

We repeated the same exercise, by replacing the `Host` field with that of an uncensored domain. Interestingly enough, the request reaches the remote host unfiltered. The functioning of the IMs can be schematically shown in figure 3.

[3] Planetlab, cloud services and hosts in different universities

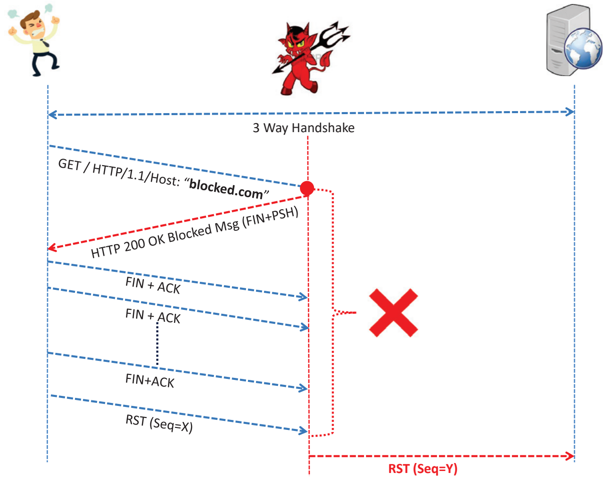
3 Way Handshake
GET / HTTP/1.1/Host: “blocked.com”
HTTP 200 OK Blocked Msg (FIN+PSH)
FIN + ACK
FIN + ACK
FIN+ACK
RST (Seq=X)
RST (Seq=Y)

Figure 3: Censorship mechanism of a Interceptive Middlebox.

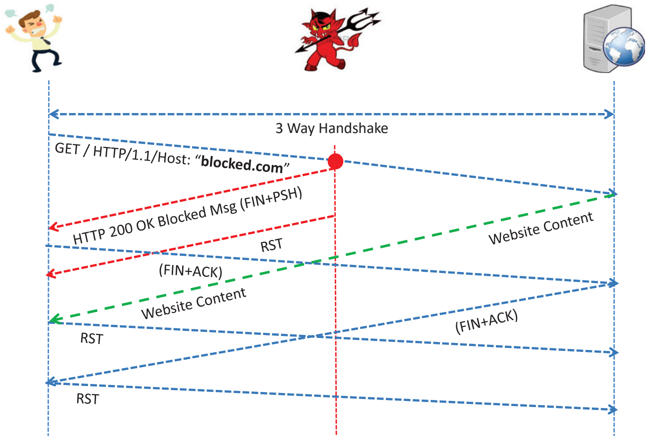

Figure 4: Censorship mechanism of a Wiretapping Middlebox.

Wiretapping middleboxes. Similar to IMs, we used our own variant of INT, to first obtain the location of the middlebox in the network path intervening the client and a filtered site.

Thereafter, we sent HTTP GET request to a blocked domain. We then inspected the network traffic (at the client) for the said message exchanges through pcap, and observed that the client receives the censorship notification-cum-disconnection packet, with the forged IP address (of the server) and TCP FIN+PSH bits enabled, which thereby enforces connection termination. Further even before the termination process resolves, the client receives a fresh TCP RST packet from the middlebox, bearing the forged IP address of the server that forces the client to terminate the connection immediately, regardless of whether the termination process, which is underway, completes or not.

Surprisingly, actual response from the filtered site eventually also arrived at the client, but the connection to the server was already terminated. The client responded with a TCP RST packet (as expected). *Such behavior indicate the presence of WMs.*

In order to confirm the censorship mechanism of WMs, we adopted an approach similar to the one described for IMs, involving remote servers under our control. We sent crafted HTTP GET requests bearing a filtered domain, to the remote servers under our control. These packets elicit the censorship notification-cum-disconnection messages, bearing the (forged) IP address of the remote hosts. The remote hosts, however, upon receiving the GET requests, ignore them as they do not host the requested domains. The behavior of WMs is shown in figure 4.

Caveat: Are middleboxes stateful or do they inspect all packets? Our initial traffic inspections using pcap hint towards stateful middleboxes that commence traffic inspection only after complete TCP 3-way handshake is resolved.

To confirm our hunches we began with the client using traceroute command to record the number of network hops between itself and the filtered site. Thereafter the client sends a TCP SYN packet with TTL just large enough to get the packet to the penultimate hop (and *not* the destination), thus avoiding a full-fledged TCP 3-way handshake.

Following this, the client sent a crafted GET request whose Host field pointed to a filtered domain bearing the TTL value, such that it expires upon reaching the penultimate hop.

If the middleboxes commence traffic inspection upon observing every fresh TCP SYN packet, they must also then inspect the subsequent crafted GET request and respond back to the client with the censorship notification-cum-disconnection message. However, we never observed censorship in such cases. All other similar heuristics, such as starting by sending a SYN+ACK or not sending the final ACK of a regular 3-way handshake, but then sending the subsequent crafted GET request never elicited censorship messages.

Finally, a crafted HTTP GET request, bearing censorious domain requests in the Host field, but with no preceding TCP handshake, also does not seem to trigger censorship.

This confirms that the middleboxes are statefull and commence traffic inspection only when they observe a complete TCP handshake. These seem different from what were observed by Wang *et al.* [49] who looked into the architecture of Chinese censorship infrastructure.

5.2.2 *Analyzing the Extent of HTTP Filtering.*

In order to analyze the extent of HTTP filtering in an ISP we proposed variants of the two previous metrics, *viz. coverage* and *consistency*.

1. ***Coverage***: A censorious ISP which is willing to use HTTP filtering must typically deploy middleboxes in a manner such that they intercept all the router-level paths inside an ISP. *Coverage* is the fraction of all such router-level paths that are intercepted by middleboxes (we call them as *poisoned paths*).

2. ***Consistency***: This metric attempts to answer the question — “how uniformly does an ISP block content” Ideally same number of websites must be blocked on all the poisoned paths of the ISP. In such a case we say the ISP is 100% consistent. For every filtered URL we determine the fraction of *poisoned* paths blocking it — (paths

that block a particular website)/(paths that block any website). *Consistency* is the average of these fractions.

In order to find consistency and coverage we started our experiments with single vantage point (VP) in the ISPs. As already discussed earlier, HTTP censorship middleboxes are agnostic to the destination IP addresses of the HTTP GET requests (as long as they appear to be a part of an existing TCP connection). We harness this behavior of middlebox to find their coverage and placement statistics.

VP within ISPs: For each of the nine ISPs under considerations, we establish TCP connections with Alexa top 1000 websites from the client machine and sent GET requests with Host fields pointing to all 1200 PBWs. Even if for single GET request we observed censorship, we considered that path to be *poisoned* by the middlebox. For Reliance Jio ISP, we only observed 64 out of 1000 paths to be tainted with middlebox. This gave us the hint that maybe middleboxes are not placed optimally to intercept a large fraction of ISP paths.

VPs outside the ISPs: To further test our observations with more VPs, we used various hosts outside India, but under our control (PlanetLab nodes, cloud infrastructure, and few other hosts in various universities). Our aim was to find the maximum number of middleboxes and the fraction of paths they intercept, inside an ISP.

For doing so, we began by scanning all live IP prefixes[4] for a particular ISP, and searched for hosts with open TCP port 80. Then we randomly sample two such IPs per prefix. We recorded the router-level path leading and the number of hops to each of these prefixes, from each vantage point, using traceroute.

We tailored our INT, targeting traces to each of these IPs (for all ISPs), where for each targeted host, we send 1200 HTTP GET requests, corresponding to each of the PBWs. Upon obtaining the censorship notification-cum-disconnection response for even a single site, we considered the corresponding network path to be *poisoned*.

We summarize our results in table 2. Column two and three represents coverage for an ISP from a single VP within ISP and multiple VPs outside of the ISP. Column four describes which type of middlebox (interceptive or wiretap) is deployed in the ISP and last column describes the total number of websites blocked out of 1200 PBW. It can be observed that Idea has highest coverage (90%) whereas Vodafone has very low coverage value (2.5%).

For Reliance Jio, we observed a very different behavior. While we saw a relatively low coverage of about 6.4% when searching for middleboxes from a vantage point positioned inside the network, we found no middleboxes when probed from the remote VPs to IPs belonging to the ISP (with open TCP port 80). There are two possible explanations for this. Firstly, the middleboxes may be sub-optimally positioned and thus the requests from the remote VPs are not intercepted. Alternatively, the middleboxes maybe filtering request not only on domain names but also for source IPs belonging to Jio network itself. Since we were unable to pinpoint the IP addresses of the middleboxes, we lacked the necessary information to further quantify our findings.

After finding the coverage of different ISPs, we now present the results obtained from computing consistency for each of them. In figure 5, X-axis represents websites which are blocked in any of the three ISPs (Vodafone, Airtel and Idea). The percentage of ISP paths that block a particular website are represented on Y-axis. It is evident from the figure, that on an average Idea network has highest consistency (76.8%) followed by Airtel (12.3%) and then Vodafone (11.6%). One may conclude that in Idea network a single website is blocked on 76.8% of the poisoned paths, as opposed to Airtel and Vodafone in which it is blocked by only ≈ 11 − 12% of the poisoned paths.

ISP	Coverage (%) (VP: within ISP)	Coverage (%) (VPs: outside ISP)	Middle-Box Type	No. of websites blocked
Airtel	75.2	54.2	WM	234
Idea	92	90	IM	338
Vodafone	11	2.5	IM	483
Jio	6.4	0	WM	200

Table 2: HTTP filtering in different ISPs.

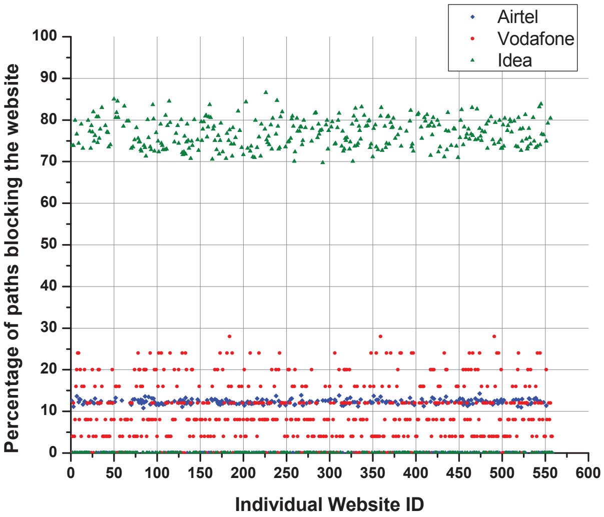

Figure 5: Consistency of middleboxes.

So far we discussed all details regarding HTTP filtering, but ignore HTTPS. We observed fewer than five instances of HTTPS filtering which were actually due to manipulated DNS responses by poisoned resolvers, and not because of SNI field in TLS client hello.

5.3 Filtering by upstream Indian ISPs

An ISP shares contractual commercial agreements with its neighbors for routing Internet traffic among themselves [32]. In our study for ISPs like NKN, Sify[5] and Siti, we never observed any filtering caused by their own policies. Rather, all the censorship instances were solely due to the policies of its neighboring ISPs. Whereas, for MTNL and BSNL it is the cumulative effect of its own and neighbors' policies.

To precisely identify the locations of middleboxes, *i.e.* which ISP they belong to, we used our tool INT. For cases, where we did not observe the IP address of the middleboxes, we used idiosyncrasies

[4] Live IP prefixes were obtained from CIDR report [2].

[5] We could not independently study TATA communications because its customer business had closed during the course of study.

of different middleboxes to identify the ISPs they belongs to (as explained in subsection 7.3).

ISPs where censorship was observed due to upstream providers, might indicate "collateral damage" (unintentional censorship) [20, 39] within the same country. This occurs when traffic of a non-censorious ISP is filtered due to a neighboring censorious ISP. In our studies we observed such unintentional censorship in several non-censorious ISPs. Table 3 summarizes our findings.

ISPs (cesnored)	**Neighboring ISPs** (causing censorship)
NKN	Vodafone (69), TATA (8)
Sify	TATA (142), Airtel (2)
Siti	Airtel (110)
MTNL	Airtel (25), TATA (134)
BSNL	Airtel (1), TATA (156)

Table 3: Filtering by upstream providers: Non-censorious ISPs observe censorship due to their censorious neighbors. For *e.g.,* in NKN, we observed 69 **websites were blocked by Vodafone and** 8 **were blocked by TATA communication.**

6 ANTI-CENSORSHIP APPROACHES

We observed two types of censorship techniques in popular ISPs of India *viz.,* HTTP filtering and DNS poisoning. In order to bypass them, we opted for techniques relevant to the middleboxes involved. Our solutions are simple and extremely effective.

Evading *DNS poisoning:* In order to circumvent poisoned DNS resolver, we tested using OpenDNS, Google's public DNS (8.8.8.8) and many other non-poisoned resolvers which belong to non-censorious countries like Ireland, Canada, and Sweden. With each of them, we were able to bypass the DNS based censorship.

Evading *HTTP filtering:* As already explained in section 3.4 middlebox gets triggered upon merely identifying a blocked domain in the `HOST` field of `GET` request. *Our goal is to craft a* `GET` *request, which is goes undetected by middlebox, but correctly interpreted by the actual website.* We tried various techniques involving string fudging [36], such as manipulating the `Host` field values, prepending `www` to the website name, changing cases of the keywords like `HTTP, GET` and `HOST`, adding spaces before and after the domain name *etc.*. Additionally we also tried approaches, like sending fragmented `GET` requests and using HTTP 2.0 as the underlying web protocol (instead of HTTP 1.1). Different approaches worked for subverting different middleboxes.

I. Wiretapping middleboxes: There are two approaches with which we bypassed these middleboxes.

- Changing the case of `Host` keyword in the `GET` request: Most popular browsers, like Mozilla Firefox and Google Chrome, use the title case for the `Host` keyword. Merely changing the case (*e.g.* changing it to `HOst`, `HoST`, `HoSt` or `HOST`*etc.*) was sufficient for request to go undetected by the middleboxes (of Airtel and Jio), but resulting in response from the actual blocked webserver. This suggests that the webservers, corresponding to the PBWs, adhere to RFC 2616 [8] and accept the keyword `Host` agnostic of the case, while the middleboxes look for exact keyword matches.
- Dropping the packets with RST or FIN bit set: As mentioned earlier, the censorship notification-cum-disconnection packet bears the TCP FIN bit set. Subsequently the middlebox also sends a TCP RST packet to enforce the client to disconnect. Using `iptables` utility, all the packets (of blocked website's IP) which have FIN or RST bit set were dropped by the kernel. For Airtel, we observed that responses from middleboxes of Airtel always bear a fixed IP-Identifier value of 242. Thus, we added a general rule that FIN or RST packets with IP-Identifier field 242 must be dropped. This effectively filters the responses from the middleboxes.
  Since the actual GET requests are not dropped by the middlebox, they reached the blocked website and elicit regular responses. These response containing the actual content of the website and are accepted by the client browser.

II. Interceptive middleboxes: We further found two types of interceptive middleboxes *i.e.,* one which sends only censorship notification-cum-disconnection message to the client (*overt*) and other which sends only a RST packet to the client without any censorship notification (*covert*).

- *Overt Censorship*: To bypass such middleboxes which overtly censors the content, we fudged the `Host` field of the `GET` request. The standard domain request looks like "`Host:` blocked.com", *i.e.* only one space between ':' and 'blocked.com'. But, instead, if we place additional spaces (or tab) in-between, *i.e.* "`Host:`   blocked.com", then the requests go undetected by the middleboxes, but servers interpret them correctly. Also, adding extra spaces (or tables) after the domain name works, *e.g.* "`Host:`blocked.com ".
- *Covert Censorship*: For bypassing such middlebox we intentionally inserted multiple `Host` fields (with different domain names) embedded in the same `GET` request to check which of those is inspected by the middlebox. In all the cases, we observed that censorship is triggered upon inspecting *only* the last `Host` keyword. Thus, by appending an uncensored domain request to the array of such `Host` keywords, we were able to bypass the middleboxes, but the server also neglects it as the request is not a standard one. Thus we crafted an unusual `GET` request, which looked something like "GET / HTTP/1.1 `Host:` blocked.com...`\r\n\r\n` `Host:` allowed.com". This request is neglected by the middlebox but on the other hand, accepted by the actual blocked website. Since middlebox is only looking at last `Host` keyword, it interprets that packet as non-suspicious and allows it to pass through. The server, on the other hand, treats the '`\r\n\r\n`' as the end of the GET request and the subsequent "`Host:` allowed.com" as a separate request. Thus, the client receives two responses from the website — the actual content due to the first `Host` field and the BAD REQUEST message for the subsequent one.

The Ant-Censorship tool — ESCAPER

We packaged all the aforementioned anti-censorship approaches into a tool called "ESCAPER". The tool runs a local HTTP proxy which needs to be configured in the browser. The client also needs to add a file mentioning the filtered websites it would like to access. The tool is written in Python and has been integrated with Mozilla Firefox browser. It runs on Windows and Linux platforms. Currently, it is maintained by us, and may be provided on-demand.

7 DISCUSSION

7.1 Count of middleboxes in the ISP

In the previous study on China [52] authors reported that they found 495 router interfaces that have filtering device attached to them. However, in India, we could not follow the same approach. Throughout our research, we used `traceroutes` and INT, with an intent of finding the location of censorship infrastructure. In all our tested ISPs, generally middlebox (or routers to which they are attached) show up as unresponsive routers (asterisked) when probed using `traceroute`. It is natural to ask if IP address of the middlebox is not known then can it be confirmed that the observed the censorship is because of the tested ISP or one of its upstream provider? We applied few heuristics:

(1) On the paths where we observed IPs of middleboxes, first we confirmed that they belong to same tested ISP. Thereafter, we recorded the corresponding censorship notifications, and used them to classify other anonymized middleboxes.

(2) In path segments where asterisked router appeared between visible ones, we checked if the latter belonged to the same ISP under test. If so then we assumed that anonymized IPs belong to the same ISP.

(3) The censorship notification messages have unique characteristics *for eg.,* in Airtel, the censorship notification packet has an embedded `iframe` which redirects to *"airtel.com/dot"* and in Reliance JIO censored response redirects to its own unique IP address. Using such unique characteristics we can easily identified the ISP of anonymized middlebox.

7.2 Issues with OONI

As already explained in section 3.1, OONI performs two sets of experiments for a given list of PBWs (1) accessing sites from client machine and (2) and accessing the same from a control sever (of OONI). If discrepancies in IP address resolutions (DNS censorship) or retrieved site contents (HTTP censorship) are observed, OONI flags the PBW as censored.

However, we found that results of OONI were misleading. They suffer from both false positives and false negatives. We now outline few possible reasons for false positives (incorrect flagging of sites as being censored):

- An unavailable website, previously hosted on hosting services (like GoDaddy) if removed, may result in different HTTP responses when accessed from different locations — an artifact of distributed hosting. Though not a case of censorship, OONI flags them as filtered.
- Many websites have dynamic content such as live news feeds and advertisement embedded in the HTTP 200 OK messages that are often location dependent. These are also mis-classified by OONI as being censored.

Also, OONI tool inspects differences in HTTP headers and body lengths of the response. If differences are greater than a threshold, it considers the site to be filtered. We observe that for a website hosted on CDN, the response at different geographic locations may come through different servers having obvious differences in the response metadata. In reality, such sites may not be blocked.

Thus when we created our scripts for detecting censorship, we only calculated the difference in the content of the response, and not the headers. If the difference is greater than the threshold (as already explained in subsection 3.4), rather than directly reporting them as blocked, we manually verified them for blocking.

We now discuss why OONI often fails to detect a censored site (false negatives). In order to identify censorship, OONI calculates the differences between (1) lengths of HTTP responses (obtained via the control server and directly through the ISP under test) (2) the HTTP header field names (3) the HTML title.

Even if one of the aforementioned condition does not hold true [7, 12], OONI considers the website to be non-censored. The following are few possible cases where OONI reports false negatives:

- We observed that for some websites, the response does not bear any content, rather a redirection link sent by the actual server. Similarly, in the censorship notification-cum-disconnection packets, there is an embedded `iframe` (which redirects to blocked page). For both the cases, the difference in the body length (of the responses) may be very less[6]. Thus, violating the first condition.
- OONI flags a website as non-censored if the header fields (and not their values) of both the HTTP response matches exactly[7]. In our measurements, we observed that most of the middleboxes use the same HTTP header as that of regular websites. Thus, the headers of censorship notification-cum-disconnection packets (generated by middleboxes) very often match the headers of the responses from the actual websites. So, OONI mistakenly classifies a censored website to be non-censored, thus violating the second condition.

Unlike the regular responses from actual websites, the censorship notification packets bore no HTML tags. OONI compares the title tags *only if* atleast one word, in both the tags, is atleast five characters long. Thus, in the absence of the title tags, OONI ignores the inspection of the censorship notifications, thereby reporting incorrect results.

7.3 Idiosyncrasy of middleboxes

- Idea middleboxes inspect traffic agnostic of their port number, while all the rest inspect only requests destined to TCP port 80.
- WM specific to Airtel have a unique characteristic — all packets generated from these middleboxes have a fixed IP-ID value (242) in the IP header; for all others', this is variable.

[6] Other variants of such scenarios are also possible *for eg.,* a small sign up/login page upon accessing the website.

[7] As verified from source code.

- Some otherwise unavailable websites[8] were still blocked by the ISPs (both through HTTP and DNS filtering). This implies that ISPs are not updating their blacklists.
- Middleboxes (IM and WM) maintain a state for all transiting TCP connections. They inspect all the connections for a duration of 2 – 3 minutes, waiting for sensitive content to arrive. If they do not receive any packet in that duration, they time out and purge the corresponding TCP state data. However, if fresh packets (corresponding to individual flows, regardless of whether they carry `GET` requests or not) arrive with the 2 – 3 minute window, the middleboxes reinstate the inspection timeout.

8 CONCLUDING REMARKS

In this work, we report a comprehensive analysis of censorship mechanism and infrastructure in nine popular ISPs of India. We commenced our research using popular censorship detection tool, OONI. However, since we observed high false positives and negatives, we discontinued using it. We developed our own automated approach (Interative Network Tracing), along with various heuristics, which we used to determine the type of censorship mechanism involved (and in some cases the approximate location of the censorship infrastructure as well). At every step we confirm our findings against the ground truth, *an effort largely ignored by several others in the recent and distant past.*

We found DNS and HTTP filtering as the *only* techniques of censorship employed by these ISPs. Further, we evolved metrics, *viz. coverage* and *consistency* that respectively describe how well the censorship infrastructure covers the ISP and how consistent they are in censoring filtered domains. In passing, we also observed interesting cases of collateral damage within the ISPs of the same country. Finally, we developed novel anti-censorship techniques, involving local firewalling and manipulating the HTTP `GET` requests, through which we were able to bypass all forms of censorship without relying on conventional methods like proxies and VPNs.

9 ACKNOWLEDGEMENTS

We would like thank our reviewers and our shepherd Johanna Amann for their valuable inputs which fortified our paper. Also, we humbly thank our colleague, Aishwarya Jaiswal for her important comments. Further, not only do we thank Dr. H.B Acharya for his inputs, but also for suggesting a title for this paper. Finally, we would like to thank Persistent Systems Ltd., India for funding the conference registration and travel.

REFERENCES

[1] Bogon ip addresses. https://ipinfo.io/bogon.
[2] Cidr report. https://www.cidr-report.org/as2.0/.
[3] Dna india. http://www.dnaindia.com/india/report-government-orders-blocking-of\-857-pornographic-websites-2110545.
[4] Facebook transparency report 2017. https://transparency.facebook.com/country/India/2017-H1/.
[5] Google transparency report 2017. https://transparencyreport.google.com/government-removals/by-country/IN.
[6] Herdict:Help Spot Web Blockages. http://herdict.org/.
[7] How ooni detects http filtering? https://ooni.torproject.org/nettest/web-connectivity/.
[8] Http 1.1 rfc 2616. https://tools.ietf.org/html/rfc2616.
[9] Instances of internet censorship in india. https://opennet.net/research/profiles/india.
[10] List of potentially blocked websites in india — citizen labs. https://github.com/citizenlab/test-lists/blob/master/lists/in.csv.
[11] Ministry of it orders isp to ban sexual abuse material. http://www.meity.gov.in/content/order-issued-meity-isps-adopt-and-implement\-iwf-resources-prevent-distribution-and.
[12] Ooni source code. https://github.com/TheTorProject/ooni-probe/blob/master/ooni/nettests/blocking/web_connectivity.py.
[13] Porn websites blocked in india: Government plans ombudsman for online content. http://gadgets.ndtv.com/internet/news/porn-websites-blocked-in-india-government\-plans-ombudsman-for-online-content-723485.
[14] Rediff. http://www.rediff.com/computer/1999/jul05dawn.htm.
[15] Right to information, a citizen gateway. http://rti.gov.in/.
[16] Shodan-search engine for internet-connected devices. https://www.shodan.io/.
[17] How the great firewall of china is blocking tor. In *Presented as part of the 2nd USENIX Workshop on Free and Open Communications on the Internet* (Berkeley, CA, 2012), USENIX.
[18] Ooni: Open observatory of network interference. In *Presented as part of the 2nd USENIX Workshop on Free and Open Communications on the Internet* (Berkeley, CA, 2012), USENIX.
[19] Aceto, G., Montieri, A., and Pescapé, A. Internet censorship in italy: An analysis of 3g/4g networks. In *Communications (ICC), 2017 IEEE International Conference on* (2017), IEEE, pp. 1–6.
[20] Acharya, H., Chakravarty, S., and Gosain, D. Few throats to choke: On the current structure of the internet. In *Local Computer Networks (LCN), 2017 IEEE 42nd Conference on* (2017), IEEE, pp. 339–346.
[21] Acharya, H., Chakravarty, S., and Gosain, D. Mending wall: On the implementation of censorship in india. In *SecureComm 2018 - 13th EAI International Conference on Security and Privacy in Communication Networks* (2017), Springer.
[22] Anonymous. The collateral damage of internet censorship by dns injection. *SIGCOMM Comput. Commun. Rev. 42*, 3 (June 2012), 21–27.
[23] Aryan, S., Aryan, H., and Halderman, J. A. Internet censorship in iran: A first look. In *FOCI* (2013).
[24] Burnett, S., and Feamster, N. Encore: Lightweight measurement of web censorship with cross-origin requests. *ACM SIGCOMM Computer Communication Review 45*, 4 (2015), 653–667.
[25] Chaabane, A., Chen, T., Cunche, M., De Cristofaro, E., Friedman, A., and Kaafar, M. A. Censorship in the wild: Analyzing internet filtering in syria. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (2014), ACM, pp. 285–298.
[26] Clayton, R., Murdoch, S. J., and Watson, R. N. Ignoring the great firewall of china. In *International Workshop on Privacy Enhancing Technologies* (2006), Springer, pp. 20–35.
[27] Crandall, J. R., Zinn, D., Byrd, M., Barr, E. T., and East, R. Conceptdoppler: a weather tracker for internet censorship. In *ACM Conference on Computer and Communications Security* (2007), pp. 352–365.
[28] Dainotti, A., Squarcella, C., Aben, E., Claffy, K. C., Chiesa, M., Russo, M., and Pescapé, A. Analysis of country-wide internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (2011), ACM, pp. 1–18.
[29] Dalek, J., Haselton, B., Noman, H., Senft, A., Crete-Nishihata, M., Gill, P., and Deibert, R. J. A method for identifying and confirming the use of url filtering products for censorship. In *Proceedings of the 2013 conference on Internet measurement conference* (2013), ACM, pp. 23–30.
[30] Ensafi, R., Fifield, D., Winter, P., Feamster, N., Weaver, N., and Paxson, V. Examining how the great firewall discovers hidden circumvention servers. In *Proceedings of the 2015 Internet Measurement Conference* (2015), ACM, pp. 445–458.
[31] Ensafi, R., Winter, P., Mueen, A., and Crandall, J. R. Analyzing the great firewall of china over space and time. *Proceedings on privacy enhancing technologies 2015*, 1 (2015), 61–76.
[32] Gao, L., and Wang, F. The extent of as path inflation by routing policies. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE* (2002), vol. 3, IEEE, pp. 2180–2184.
[33] Gill, P., Crete-Nishihata, M., Dalek, J., Goldberg, S., Senft, A., and Wiseman, G. Characterizing web censorship worldwide: Another look at the opennet initiative data. *ACM Transactions on the Web (TWEB) 9*, 1 (2015), 4.
[34] Guo, S., and Feng, G. Understanding support for internet censorship in china: An elaboration of the theory of reasoned action. *Journal of Chinese Political Science 17*, 1 (2012), 33–52.
[35] Internet censorship in iran.
[36] Jermyn, J., and Weaver, N. Autosonda: Discovering rules and triggers of censorship devices. In *7th USENIX Workshop on Free and Open Communications on the Internet (FOCI 17). USENIX Association, Vancouver, BC. https://www.usenix.org/conference/foci17/workshop-program/presentation/jermyn* (2017).
[37] Khattak, S., Javed, M., Anderson, P. D., and Paxson, V. Towards illuminating a censorship monitor's model to facilitate evasion. In *FOCI* (2013).

[8] Tested via Tor circuits ending in non-censorious country.

[38] Knockel, J., Ruan, L., and Crete-Nishihata, M. Measuring decentralization of chinese keyword censorship via mobile games. In *7th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 17)* (2017), USENIX Association.

[39] Levis, P. The collateral damage of internet censorship by dns injection. *ACM SIGCOMM CCR 42*, 3 (2012).

[40] MacKinnon, R. Flatter world and thicker walls? blogs, censorship and civic discourse in china. *Public Choice 134*, 1-2 (2008), 31–46.

[41] Marczak, B., Weaver, N., Dalek, J., Ensafi, R., Fifield, D., McKune, S., Rey, A., Scott-Railton, J., Deibert, R., and Paxson, V. ChinaâĂŹs great cannon. *Citizen Lab 10* (2015).

[42] Nabi, Z. The anatomy of web censorship in pakistan. In *FOCI* (2013).

[43] Pearce, P., Ensafi, R., Li, F., Feamster, N., and Paxson, V. Augur: Internet-wide detection of connectivity disruptions. In *Security and Privacy (SP), 2017 IEEE Symposium on* (2017), IEEE, pp. 427–443.

[44] Pearce, P., Jones, B., Li, F., Ensafi, R., Feamster, N., Weaver, N., and Paxson, V. Global measurement of dns manipulation. In *Proceedings of the 26th USENIX Security Symposium (SecurityâĂŹ17)* (2017).

[45] Sfakianakis, A., Athanasopoulos, E., and Ioannidis, S. Censmon: A web censorship monitor. In *USENIX Workshop on Free and Open Communication on the Internet (FOCI)* (2011).

[46] Son, S., and Shmatikov, V. The hitchhikerâĂŹs guide to dns cache poisoning. In *International Conference on Security and Privacy in Communication Systems* (2010), Springer, pp. 466–483.

[47] Verkamp, J.-P., and Gupta, M. Inferring mechanics of web censorship around the world. In *FOCI* (2012).

[48] Ververis, V., Kargiotakis, G., Filasto, A., Fabian, B., and Alexandros, A. Understanding internet censorship policy: The case of greece. In *5th USENIX Workshop on Free and Open Communications on the Internet (FOCI)* (2015).

[49] Wang, Z., Cao, Y., Qian, Z., Song, C., and Krishnamurthy, S. V. Your state is not mine: a closer look at evading stateful internet censorship. In *Proceedings of the 2017 Internet Measurement Conference* (2017), ACM, pp. 114–127.

[50] West, D. M. Internet shutdowns cost countries $2.4 billion last year. *Center for Technological Innovation at Brookings, Washington, DC* (2016).

[51] Wright, J. Regional variation in chinese internet filtering. *Information, Communication & Society 17*, 1 (2014), 121–141.

[52] Xu, X., Mao, Z. M., and Halderman, J. A. Internet censorship in china: Where does the filtering occur? In *International Conference on Passive and Active Network Measurement* (2011), Springer, pp. 133–142.

[53] Yang, Q., and Liu, Y. WhatâĂŹs on the other side of the great firewall? chinese web usersâĂŹ motivations for bypassing the internet censorship. *Computers in human behavior 37* (2014), 249–257.

[54] Zittrain, J., Budish, R., and Faris, R. Herdict: Help spot web blockages, 2014.

[55] Zittrain, J., and Edelman, B. Internet filtering in china. *IEEE Internet Computing 7*, 2 (2003), 70–77.