On The Implementation Of Internet Censorship In India

By

Kartikey Singh

Under the supervision of

Dr. Sambuddho Chakravarty

Indraprastha Institute of Information Technology Delhi

July, 2022

On The Implementation Of Internet Censorship In India

By

Kartikey Singh

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

To

Indraprastha Institute of Information Technology Delhi

July, 2022

Certificate

This is to certify that the thesis titled *On The Implementation Of Internet Censorship In India* being submitted by *Kartikey Singh* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Dr. Sambuddho Chakravarty

July, 2022

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

Acknowledgements

I would like to sincerely thank my advisor, Dr. Sambuddho Chakravarty, whose constant guidance, availability, and support have enabled me to navigate every obstacle encountered in the course of this project. I have learned and grown immensely through their constructive feedback on my work.

I am also thankful to former PhD students Dr. Devashish Gosain and Dr. Piyush Sharma and current PhD student Jithin S. for their patient guidance on this work.

I would also like to acknowledge my family and friends for being a consistent source of support through exceptionally trying times in my life.

Lastly, I am grateful for my time at IIIT Delhi, which has proved to be a turning point in my life and has provided me with the opportunity to learn from exceptionally knowledgeable professors to better myself at my craft. Every lecture has been a privilege.

Abstract

In this work we highlight the evolution of Internet censorship within the largest democracy *i.e.,* India. We first report a new form of Internet censorship *i.e.,* mobile app blocking and describe in detail the mechanics involved. We studied 220 mobile apps that were recently blocked in India. Through our research efforts, we observed a novel "three-tiered" app censorship scheme in India, with each tier increasing the sophistication of censorship. After thoroughly analyzing the app censorship mechanisms, we present effective circumvention techniques to bypass the tiered app censorship. We were able to access all the blocked apps with the said techniques.

Secondly, we also present a detailed study of the evolution of web censorship within India. We performed tests in the four major ISPs (that together serve $> 95\%$ of Indian netizens). We discovered that they have upgraded their censorship techniques and infrastructures. For example, three of these ISPs are now engaged in wide-scale HTTPS censorship, *in addition* to the previously reported HTTP censorship. Furthermore, we also discovered that the fourth ISP actively employs DNS injection (along with DNS poisoning), a fact hitherto unreported for India. We managed to bypass all forms of web censorship by intelligently crafting packets, without needing to use conventional tools (*e.g.,* VPNs).

Overall our work highlights the evolving censorship ecosystem (for app as well as web) in India.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Free and open communication over the Internet is a fundamental human right [1]. However, many nation-states have attempted to curb the free flow of information over the Internet by citing different reasons—national security, protection of confidential information, safeguarding the privacy of citizens *etc.* This has resulted in wide scale censorship in different parts of the globe [2, 3].

While there exist a plethora of studies that systematically analyzed the Internet censorship in different countries [4–6], a significant fraction of them focused on "Great Firewall of China" and reported its multi-faceted censorship [7–18]. In this work, we take a different direction and focus on a less studied country India [19–21]. Our research highlights how India, a democratic country, has gradually upgraded to achieve a stricter control over the Internet.

Previous censorship studies on India reported different categories of blocked websites [19], the techniques used [21] and detailed analysis of mechanics employed by Indian ISPs, to achieve censorship [20]. However, in this study we further explore the evolved censorship landscape in India. We first study a new form of censorship *i.e.,* blocking of mobile apps, followed by a measurement study that highlight the surreptitious evolution of web censorship within India.

Due to the rapid growth of mobile Internet applications, app-censorship will become an essential component of any nationwide censorship system. Already, there are anecdotal evidences of banning of mobile apps by different countries [22–25]. But to best of our knowledge, no prior

research has systematically analyzed app censorship and the involved mechanics. Thus, we fill in this research gap by conducting a first comprehensive study to analyze the *mechanics* of this less studied form of censorship (by considering India as a case study). Banning of mobile apps is a recent phenomenon in India. In June 2020, Indian government first issued orders to block 59 popular Chinese apps [26, 27]. Since then, India continued its app blocking spree and presently a total of 220 Chinese apps are blocked.

We commenced our research by analyzing a popular app *TikTok* (which has over 2.6 billion downloads worldwide [28]). As expected, this app was not available on any Indian app stores (for both android and iOS). Thus, we obtained its installation package offline from our overseas contacts. We installed the app on a mobile phone (in India) and attempted using it. We were not able to access the app's content and observed a censorship notification message displayed within the app itself. Thereafter, we analyzed the network traffic (through `pcaps`) to identify the underlying censorship mechanism(s). Since the mobile apps also use the web protocol, we expected to see traditional censorship techniques being used by the Indian ISPs *viz.* DNS filtering, TCP/IP blocking, and keyword filtering [18, 29].

However, to our surprise, we observed no interference from the Indian ISPs. We observed the successful TLS connection between the app and the actual TikTok server confirmed via TikTok's legitimate certificate. Moreover, we also noticed that the censorship notification banner was a part of the same TLS session, proving that it was sent by the TikTok server. *This confirmed that TikTok app censorship is not carried out by the Indian ISPs, rather it was by the TikTok server itself.*

Our observations were similar for all other censored apps as well *i.e.,* Indian ISPs were not at all involved in the censorship. Rather, the app servers were themselves selectively filtering the Indian users. To identify the technique(s) for censorship, we analyzed traffic footprints (through `pcaps`) and even reverse-engineered a few apps. Our observations were surprising—some apps were even *probing the SIM card* for country information (ref. §3 for more details).

However in January 2021, we observed changes in the censorship mechanics used by a few apps. Indian government imposed a "permanent" ban on all the previously banned Chinese

apps. At the same time, we identified that some of the blocked apps started adopting multiple censorship techniques. For example, TikTok servers started identifying Indian users by both probing the SIM cards *and* source IP addresses of the clients. As of now, these apps can be broadly divided into three tiers.

1. *Tier three* apps are those that are unavailable in the Indian app stores. We found 160 such apps (Out of 220 blocked apps, 60 apps were defunct. We couldn't find them in official play stores of foreign countries as well.). 136 out of these can be accessed directly after installation.

2. *Tier two* apps are those, that after installation, would still suffer censorship. These app publishers selectively filter Indian clients using geo-blocking [30]. 23 (out of previous 160) apps fall under this category.

3. *Tier one* apps are those that employ the most sophisticated technique. As already explained with the example of TikTok, not only these apps censor users by identifying locale information from the SIM cards installed, but also use geo-blocking. In total 7 (out of previous 23) apps employ such censorship techniques. However, an additional app, *MICO Chat* is an exception that has only adopted SIM based blocking.

It could further be questioned that since most of these apps would be relying on content distribution networks (CDNs) to deliver content, can app publishers restrict the content on Indian CDN edge-servers (responsible for delivery of content in India)? We present our approach (to answer this question) in §3.2. Our research reveals that presently only one app, *viz.* ChessRush, employs such a scheme to censor Indian users.

In general our findings on mobile apps shed light on some grave concerns. Following this model, in future, many app publishers may adopt similar censorship techniques, if coerced by authoritarian regimes. Alarmingly, *all* apps could censor users, even without communicating to app servers. The logic to extract country information (*e.g.,* from SIM cards) can be embedded within the apps' binary (With a regular update in the app, the publishers can easily introduce such changes).

These recent advancements in (app) censorship, motivated us to further analyze the current state of web censorship in India. We focused on four popular Indian ISPs that collectively serve around 95% of the Indian Internet subscribers [31].

Prior censorship studies [19, 20] showed that India exercises a federated control to achieve censorship *i.e.,* different ISPs employ different censorship schemes (HTTP, DNS *etc.*) and they also maintain separate blacklists. Recently, Singh *et al.* [21] extended the previous work and reported HTTPS censorship (using the SNI field of TLS ClientHello) in a single Indian ISP.

However, in this work we conducted a detailed measurement study to understand the evolution of censorship and answer some more pertinent questions like—What censorship techniques are currently employed? Have other major ISPs also started to employ HTTPS censorship? Do different ISPs still maintain separate blacklists? What are the circumvention schemes to evade such censorship *etc.*?

In comparison to the previous work, we observed major upgradations in the censorship techniques employed. In 2018, none of the ISPs employed HTTPS filtering (using SNI) [20]; in 2020, only Reliance Jio was using this technique. However, amongst the four major ISPs we studied, except ACT, the remaining three ISPs now filter HTTPS traffic as well. Moreover, similar to [21], we also observed that ISPs have deployed multiple censorship techniques. While Jio and Vodafone have implemented HTTP and HTTPS filtering, Airtel uses all three—DNS, HTTP and HTTPS. Interestingly, we also identified DNS injectors specifically deployed by ACT network, a fact yet unreported for India.

In our study we also observed a peculiar trend that was previously unreported. In some ISPs (*i.e.,* Jio and Vodafone), middleboxes intercept a very small fraction of router-level network paths, but still censor vast majority of the websites we tested; this indicates intelligent positioning of censorship middleboxes by these ISPs (ref. §6.1 for details).

Lastly, we present some simple yet effective circumvention schemes (in §4.4) that help in evading the deployed middleboxes. These schemes do not use third-party anti-censorship solutions like proxies. Rather, they send intelligently crafted packets such that these packets can elude the middleboxes (*e.g.,* by disrupting the TCP state within them [18, 32, 33]), and at the

same time elicit responses from the filtered website.

To summarize:

1. We present a first systematic app censorship study of 220 filtered Chinese apps and report interesting findings:
   (a) Indian ISPs are not involved in app censorship, rather app publishers are *themselves* filtering the Indian users.
   (b) App publishers go to great lengths to identify Indian users. As a consequence, we observe a sophisticated three-tiered censorship hierarchy.
   (c) We also found that app publishers even exploit CDN infrastructure to achieve censorship.
   (d) We bypassed the three-tiered censorship for all filtered apps.
2. In a matter of months India witnessed the rise of an effective form of app censorship. Thus, as a natural progression, we investigated if there was any advancement in the web censorship ecosystem in India as well. Through our 15 month long measurement study involving four major Indian ISPs (that together serves over 95% of Indian netizens) we observe that:
   (a) Indeed, the web censorship ecosystem has evolved over time. *E.g.,* three out of the four top ISPs now actively engage in HTTPS (*in addition* to previously reported HTTP) censorship and the fourth one employs DNS injection (along with previously reported DNS poisoning) to achieve effective large-scale censorship.
   (b) However, with our simple yet effective circumvention schemes one can bypass all types of web censorship.
   (c) We gauge the extent of censorship using coverage and consistency metrics *i.e.,* average number of network paths and requested domains the ISPs censors respectively. *E.g.,* in Airtel ISP we observed the coverage to be 70% *i.e.,* 70% of the network paths had presence of middlebox(es) that filter the censored domains.

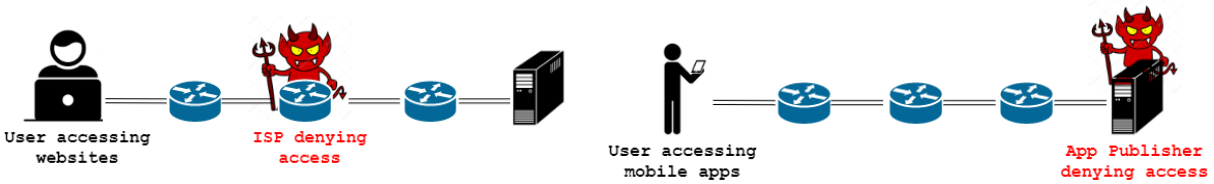
(d) Interestingly, in some ISPs (*e.g.,* Jio), middleboxes intercept a very small fraction of network paths, but still censor vast majority of the websites we tested.

Chapter 2

Background and Related Work

In this work we focus on India that seems ambivalent about its censorship policies [19]. As already mentioned, two different forms of censorship mechanism have evolved in the country—*viz.* mobile app and web censorship.

A. ***App censorship:*** In June 2020, for the first time, Government of India banned 59 Chinese apps claiming that these apps pose a threat to the privacy and data security of Indian users [27]. By November 2020, the number of banned apps increased to 220 [34]. Banning of these apps involved removing them from all the official app stores (*e.g.,* Google and Apple). Moreover, following the ban even the pre-installed apps stopped working. As described ahead, several of these app publishers go to great lengths to identify Indian users so as to not serve content to them. This departs from the traditional model of censorship where ISPs attempt to block content rather than the website maintainer itself. Further, we also confirm that the techniques used for censoring apps are quite different from those used in web censorship.



(a) Web Censorship

(b) App Censorship

Figure 2.1: Types of Internet censorship in India: (a) Web censorship is achieved by an ISP (b) App censorship is achieved by the app publisher themselves.

Such forms of censorship are yet unreported in literature. Thus it becomes important to study such app censorship and to identify possible circumvention solutions. In future other censoring countries could adopt similar censorship techniques to censor mobile apps apart from traditional website filtering.

*B. **Web censorship:*** There exist plethora of studies that reported Internet censorship across the globe [4, 5, 35, 36]. Researchers have reported censorship in various countries like Syria [37], Iran [38], Greece [39], Italy [40], Pakistan [41], Thailand [42], Afghanistan [43], Russia [29], Spain [44] *etc.* But these studies together are far less compared to those conducted for the Great Firewall of China [9, 15, 16, 18, 45–55].

In this work, we focus on India—the largest democracy with more than a billion Internet users [31]. Anecdotal evidences of web censorship in India date back to 1999, when popular Pakistani daily, Dawn's website was censored in India [56]. In the year 2017, Gosain *et al.* [19] conducted the first (preliminary) study and reported that Indian ISPs follow a "federated model of censorship" *i.e.,* they have inconsistent censorship policies that results in huge differences in the censorship experienced by the netizens. Later, in 2018, Yadav *et al.* [20] conducted a detailed study of web censorship in India and reported its multi-faceted aspects. They confirmed the previous observations [19] and reported the presence of various censorship middleboxes positioned in India. They further demonstrated that using specially crafted web requests, similar to those reported in [33, 57], it is possible to bypass such middleboxes. However, Yadav *et al.* did not report filtering of HTTPS websites. But recently, in 2020, Singh *et al.* [21] reported that one Indian ISP has started blocking HTTPS websites using TLS SNI extension.

There exist other large-scale measurement projects like ICLab [5], OONI [58], and CensoredPlanet [4] that track and report censorship events across the globe, including India. These measurement projects are extensive in scale and report a breadth of important information about censorship, but do not provide detailed insights about the mechanics involved. Thus, in this work we present a closer examination of Internet censorship within India. We specifically attempt to answer important questions like—what type of middleboxes have ISPs deployed to achieve HTTPS blocking? Do previous anti-censorship approaches [20] still work? If yes, using them can we bypass middleboxes that cause HTTPS censorship as well? Is the placement of middleboxes

by the ISP ad-hoc or intelligent *etc.*? We answer these questions in §4.

Chapter 3

App Censorship Investigation

Mobile app censorship is a recent phenomenon in India; around 220 apps were banned in the country by the end of November 2020. Our objective was to study how exactly these apps were blocked in the country. More precisely, we attempted to answer the following questions:

- Are the apps available in app stores?
- If not, how can we obtain these apps?
- Are these apps accessible/usable once we install them from alternate sources?
- After installation, if the apps are not usable, what are the mechanisms used to censor them?

Accessing censored apps: The mobile apps were gradually banned in India in three stages, first in June, then in September and lastly in November 2020. We began our research by curating a list of the banned apps from the press releases of the Ministry of Electronics and IT, Government of India [59] [60] [61]. The banned apps belonged to different categories *viz.* gaming, social media, dating *etc.* (as reported by the app publishers).

We began by searching for these apps on popular app stores (*e.g.,* Google and Apple). As expected, these apps were removed from them. Thus, to obtain the installation packages of the blocked apps (*e.g.,* `apk` files), we relied on our overseas contacts in *uncensored* countries where these apps were likely available. Interestingly, 60 among these 220 apps were unavailable

even in their app markets, indicating that they were likely not operational. Additionally, we verified that the installation packages of these apps were unavailable even on third-party sources *e.g.,* `apkmirror.com` (ref. §6.3 for details). Finally, our overseas contacts downloaded the packages for the remaining 160 apps from their official stores and shared them with us.

After installing these apps on our mobile phones in India, we manually tried accessing them. We found that 136 out of 160 apps (that were unavailable in the Indian app stores) could be trivially accessed, and none of their functions or features were censored. Thereafter, we tried to investigate the censorship mechanisms employed for the rest of the 24 apps.

Initial observations: We commenced our research by analyzing a few popular blocked apps like TikTok, PUBG and Vmate *etc.* Upon accessing TikTok, we observed a censorship notification issued in compliance with the government's orders and were also unable to load the content. Next, when we attempted to open PUBG game, it showed no explicit censorship message; instead it simply reported a connection error and prevented us from launching the game. In addition, several other apps like Vmate and UVideos neither showed any censorship notification, nor any connection error. Rather, they showed no contents whatsoever. This indicated that different apps were likely being censored in different ways. These apps rely on standard web protocols (*i.e.,* HTTP and HTTPS). It is possible that Indian ISPs might be filtering the traffic of these apps, much like traditional web censorship [20, 21].

3.1 Investigating ISP level filtering

Indian ISPs generally rely on keyword filtering (*e.g.,* DNS, HTTP and HTTPS headers *etc.*) [19–21] to filter traffic. Thus we began by inspecting the network traffic of TikTok app via `pcap` files. However, we observed only the standard network protocol messages:

1. The domain `tiktok.com` resolved to public IP addresses belonging to AS 23903 (Akamai). We resolved the same domain from other VPSs in uncensored countries and found that the IP addresses obtained, belonged to Akamai as well. Thus, DNS censorship was likely not being used as all IP addresses belong to the same hosting service.

2. With the said IPs, we were able to successfully complete TCP handshake. This largely ruled out TCP/IP filtering.

3. Following that, we were also able to successfully complete the TLS handshake with the same said IPs. We observed a legitimate certificate signed by `DigiCert Inc.` bearing the common name as `*.tiktokv.com`. This confirmed that HTTPS censorship (using SNI extension of TLS ClientHello) was also not employed.

4. Eventually, we observed that encrypted data was exchanged between the app and the TikTok server.

We analyzed the network traffic for all other blocked apps and our observations were consistent—*the responses received by the apps were from the app server itself, and not manipulated by the ISP (as it usually happens with web censorship)*. Ironically, this indicated that the app publishers themselves, and not the ISPs, were restricting the Indian app users from accessing the app content.

3.2 Investigating censorship mechanics used by app publishers

After ruling out the role of Indian ISPs in app censorship, our next concern was how app publishers were selectively filtering users from India, and not from elsewhere. We hypothesized that these blocked apps could be using a well-known technique, *IP geo-blocking* [62] for the same. IP geo-blocking involves web-servers rejecting requests originating from specific regions, identified through their source IP address. This step often involves looking up geo-IP databases to map IP addresses to countries. However, an alternative could be to filter requests from specific regions. A large number of such app publishers might use CDNs, much like almost all popular web services [63–65]. In such a case, the app publisher could restrict content from being available to edge-servers that serve requests from specific regions (*e.g.,* India). Thus, app publishers might geo-block clients largely in two ways:

1. *Geo-blocking based on source IP address:* The app publisher would restrict the users based

on the IP address of the incoming requests.

2. *Geo-blocking by restricting content on CDN edge-servers:* The app publisher may have the control to restrict content on edge-servers that serve requests from India.

For those apps that do not rely on CDNs, the edge-server based blocking could be directly ruled out. For others (that use CDNs), we would need to distinguish between the aforementioned two possibilities. Thus, we first identified whether the blocked apps were relying on CDNs or not.

Identifying the apps that use CDNs: Identifying the use of CDNs requires distinguishing between different types of CDNs. Broadly there are two types of CDNs *viz. DNS* based, and *anycast* CDNs [66, 67]. In DNS based CDNs [68] (*e.g.,* Akamai), DNS queries for web services are resolved often to the nearest edge-servers, generally identified from the clients' DNS resolvers' locations. However, in anycast based CDNs [67] (*e.g.,* Cloudflare), edge-serves in different locations use the same IP address, which is announced through different BGP advertisements from different geographic locations. A client's web request is directed to the closest possible edge-server, based on BGP policies of the client's ISP.

Our approach to identify the use of CDNs and their types relies on how DNS and anycast CDNs work. We now describe the same. For each of the 24 apps:

1. We recorded the app's network traffic as a `pcap` file.
2. Using the `pcap` file we identified the unique domains, to which apps communicate.
3. We resolved the same set of domains from 5 different uncensored countries (We used VPSes in these countries for the same)., and recorded the IP addresses obtained from each location.
4. Across each location, for each of the domains, we checked if the resolved IP addresses were the same or different.
   (a) If from each of the 5 locations we observed different IP addresses corresponding to the same domain, we classified the domain to be using DNS based CDN.

(b) Else, if a domain is resolved to the same IP address at every location, then two possibilities exist:

i. The domain is unicasted (*i.e.,* it is hosted on a non-CDN infrastructure): To confirm the same, we ran `traceroute` from the five geographically diverse VPSs to the same IP addresses. If all the `traceroute` paths end in the same country (Maxmind geolocation database), we classified it as unicasted.

ii. The domain is anycasted: If the `traceroute` paths end in separate countries (likely the ones where they originate), we classified the domain as anycasted.

We observed that on average each of the apps was communicating with 8 unique domains and majority of these (> 6) were using DNS based CDNs. A few of them were anycasted (or unicasted). We found that all the 24 app publishers were hosting content on some form of CDNs. After confirming the role of CDNs with blocked apps we revisit our problem of how app publishers censor Indian users and role of CDNs (if any).

Geo-blocking mechanisms employed: Our goal was to identify how app publishers are selectively filtering Indian users—on the basis of source IP, or simply denying access on edge servers catering the Indian users. Accessing the apps via VPNs with end-points abroad could be an easy way to confirm if the apps are accessible or not. This may likely help circumvent IP geo-blocking. However, our aim was to first identify how exactly geo-blocking was implemented, but VPNs not only change the source IP addresses of apps' requests, but also the edge-servers to which they communicate (depending on the DNS resolvers the VPNs used). This makes it hard to discern how the requests are being filtered.

Thus, we devised heuristics involving changing a single factor at a time. Corresponding to these two factors, we examined four possible scenarios. For instance, one scenario is accessing apps by selecting foreign edge-servers, while still using an Indian IP address. Since the apps were using DNS based CDNs, switching to DNS resolvers in uncensored countries could force the apps to communicate with foreign edge-servers, without changing the Indian source IP address. Alternately, another case would be to access apps with foreign source IP and connecting to Indian edge-servers. For this, our overseas contacts (whose phones bear foreign IPs) used

Indian DNS resolvers to communicate with Indian edge-servers. We now elucidate all the four possible scenarios.

***Case 1**: Indian Source IP and Indian edge-server:* The blocked apps were accessed directly from our Indian mobile phone, configured to use Indian resolvers. This is the typical situation where a regular user tries to use the app. Thus while 136 apps were trivially accessible after installation, the 24 apps we identified fell in this category and were inaccessible (ref. Table 3.1).

***Case 2:** Indian Source IP and Foreign edge-server:* The apps were accessed from phones configured to use open resolver in non-censoring countries (*i.e.* that likely do not block apps based on government orders). This enabled the apps to connect to an edge-server likely located in such countries. We verified this by inspecting the `traceroute` path from an Indian mobile phone to the resolved IPs of the edge-servers. The last few IP addresses in the `traceroute` paths belonged to the same (uncensored) country as that of the DNS resolver.

Unfortunately, the 24 apps were still inaccessible, even when they communicated to foreign edge-servers. While the DNS resolvers of the phones were changed, their IP addresses weren't. This indicated that the app publishers were filtering requests based on source IP address, through the edge-servers, even when the latter were outside India.

***Case 3:** Foreign Source IP and Indian edge-server:* To use a foreign source IP, we set up a VPS in an uncensored country and ran our own `OpenVPN` service on it. We configured our Indian mobile phone (with blocked apps installed) to use the said `OpenVPN` service. This ensured that even if we accessed apps from the mobile phone (in India), the requests would bear a foreign source IP address. Both the `OpenVPN` service and the VPS host were configured to use an open DNS resolver in India. This forced the apps to connect to edge-servers that cater to Indian users. We found that 15 among the 24 apps, mentioned above, were accessible; their traffic bore foreign source IPs and was destined to Indian edge-servers. Thus based on the hitherto 3 cases, we observed that:

1. All the 24 apps were censored when their requests bore Indian source IPs and connected to Indian edge-servers.

2. They were censored even when they were connected with foreign edge-servers, while using Indian source IPs.

3. However, 15 of them *were accessible* when connected to Indian edge-servers, but used foreign source IPs. *This confirmed IP geo-blocking for these apps.*

Case 4: *Foreign Source IP and Foreign edge-server:* Finally, we accessed the apps through VPN with end-points in foreign countries. This resulted in the requests bearing foreign source IPs, and terminating at foreign edge-servers. We expected all the 24 censored apps to be accessible. To our surprise, we were able to access only 16 out of 24 apps! These 16 apps included the previously accessible 15 apps.

Interestingly, the additional app *Chess Rush,* was both IP geo-blocked as well as unavailable at the Indian edge-server. It was only accessible when using both foreign source IP address and foreign edge-server. These 16 apps with their censorship mechanisms are enlisted in Table 3.1 (rows 1–16). To understand the censorship mechanics of the remaining 8 apps we ran additional experiments.

Investigating the blocking of remaining eight apps: As previously mentioned, we were unable to access these apps using VPNs. Likely there were additional location revealing parameters sent by the apps to their server, which might have lead to censorship. In general, mobile phones (both Android and iOS) present multiple interfaces that reveal the location *e.g.,* GPS, time-zone information, language to the apps. Before conducting our experiments we ensured that all such user configurable interfaces were turned off from revealing location, *e.g.,* we turned off the GPS, changed the time-zone of the phone to a foreign country *etc.* But, we were still unable to access these apps whether we used VPNs or not.

To investigate further, we once again selected TikTok for the detailed analysis (as it was one of the remaining 8 apps). We relayed its traffic via our MITM proxy [69], so as to see if location identifying parameters were being relayed via the requests (ref. §5 for the details). Interestingly, TikTok app was sending the country code "IN" as a part of query string in multiple HTTP requests to the app server, even when all configurable location revealing attributes were turned off. Thus, we changed *some* of the parameters in the requests to a different country (*e.g.,*

"US"), on the fly using the MITM proxy. But, still we suffered censorship.

As a last resort, we reverse-engineered the TikTok app to identify the potential censorship logic (if any), embedded within the app's code. We used the `jadx` decompiler [70] for the same. We obtained a partly decompiled code of the app. Careful inspection of TikTok's code, revealed the use of functions like `getSimCountryISO()`. This function is a part of the Android `TelephonyManager` API and is used to access SIM information. This revealed that TikTok might be fetching country related information from the SIM card.

Thereafter, we confirmed that only when an Indian SIM is installed in the mobile phone, TikTok sends a `carrier_region=IN` parameter in HTTP requests; otherwise this parameter was absent. Using the MITM proxy, we changed this parameter's value from "IN" to "US" on-the-fly, and finally we were able to bypass the censorship. Suppressing the parameter also worked, hence an alternative is to simply remove the installed SIM card. It must be noted that there are other parameters like `op_region=IN` that are also sent in different HTTP requests. But only changing the `carrier_region` parameter resulted in circumvention.

Simply accessing the app without the SIM (through a WiFi network), was sufficient to bypass censorship for each of the remaining 8 apps. This confirmed that these apps were identifying requests from India by probing the installed SIM card. To further confirm our deductions, we ran some additional tests. (1) When our overseas contacts (in uncensored countries) accessed these eight apps with Indian SIMs in their phones, their requests were also censored. However, with foreign SIMs, they were able to freely access the apps. (2) Indian mobile phones installed with foreign SIM cards (in our case Singapore), had no problems accessing these apps.

Additionally, in dual SIM phones, the apps inspect location information only from the primary SIM. Thus using an Indian SIM in the secondary slot, while leaving the primary slot empty (or installing with a non-Indian SIM) allows uncensored access.

To conclude, the apps transmit country information to the servers by probing the primary SIM. The app servers use this to identify and censor Indian users, irrespective of their actual geographic location. These 8 apps with their censorship mechanisms are enlisted in Table 3.1 (rows 17–24).

Sl No.	App Name	App Type	Censorship Technique Used		
			Client Source IP	CDN Edge server	Client SIM Card
1	PUBG	Gaming	✓	✗	✗
2	ShareIt	Tools	✓	✗	✗
3	Shein	Shopping	✓	✗	✗
4	Baidu	Tools	✓	✗	✗
5	Tantan	Social	✓	✗	✗
6	VooV	Productivity	✓	✗	✗
7	RomWe	Shopping	✓	✗	✗
8	Ludo	Gaming	✓	✗	✗
9	Rangers of Oblivion	Gaming	✓	✗	✗
10	Ali Suppliers	Business	✓	✗	✗
11	Baidu Express	Tools	✓	✗	✗
12	DingTalk	Productivity	✓	✗	✗
13	MangoTV	Video Players	✓	✗	✗
14	Heroes Evolved	Gaming	✓	✗	✗
15	Singol	Dating	✓	✗	✗
16	ChessRush	Gaming	✓	✓	✗
17	TikTok	Social	✗	✗	✓
18	Likee	Video Players	✗	✗	✓
19	Kwai	Social	✗	✗	✓
20	UC Browser	Browser	✗	✗	✓
21	FaceU	Photography	✗	✗	✓
22	Hago	Social	✗	✗	✓
23	V-Fly	Tools	✗	✗	✓
24	MICO Chat	Social	✗	✗	✓

Table 3.1: Censorship mechanisms employed by different blocked apps (before the permanent ban).

Permanent ban on the apps: At the end of January 2021, Indian government imposed a permanent ban on the said 220 apps. Interestingly, soon after the permanent ban was enforced, we observed changes in the censorship mechanisms of *only* the previously mentioned eight apps. Out of the eight blocked apps (based on SIM card's location) we were able to access only one app (MICO Chat) after removing the SIM. The remaining seven apps were inaccessible, even when the Indian SIM card was not present in the phone.

This change in censorship mechanism could be attributed to two possibilities: (1) The app servers (for these seven apps) have now adopted geo-blocking (using IP addresses or disabling the content on edge-servers serving Indian users), or (2) instead of fetching the country information from the SIMs, the apps might be accessing locale information via other parameter.

To check the possibility of geo-blocking, we removed the SIM card and repeated our previously mentioned four cases that involved around changing the source IP address and the CDN edge-servers. We confirmed that these 7 apps were now censored using IP geo-blocking.

Further, to check if the location information from SIM card was still being used or not, we plugged the SIMs back into the phones before accessing the apps. Thereafter, we accessed the apps via VPNs, to ensure foreign source IPs and edge-servers. To our surprise, the app servers were still filtering the apps' requests. Examination of the network traffic using MITM proxy revealed that presence of the earlier location parameter (`carrier_region=IN`). Like earlier, masking this parameter (either via the MITM proxy or by simply uninstalling the SIM) makes it difficult for the app servers to identify the country of origin.

We conclude that following the permanent ban, the app servers use both the source IP address, as well the location revealing parameter (fetched by the app from the SIM card), to identify Indian users. We tabulate these 7 apps with their censorship techniques in Tab. 3.2.

Sl No.	App Name	App Type	Censorship Technique Used		
			Client Source IP	CDN Edge server	Client SIM Card
1	TikTok	Social	✓	✗	✓
2	Likee	Video Players	✓	✗	✓
3	Kwai	Social	✓	✗	✓
4	UC Browser	Browser	✓	✗	✓
5	FaceU	Photography	✓	✗	✓
6	Hago	Social	✓	✗	✓
7	V-Fly	Tools	✓	✗	✓

Table 3.2: Censorship mechanisms employed by different blocked apps (after the permanent ban).

Chapter 4

Web Censorship Investigation

The steep progress in app censorship (in a matter of months) further motivated us to revisit the current state of web censorship in India. Previous studies [20, 21] have already demonstrated that in India, different ISPs use various types of web censorship techniques *viz.* DNS, HTTP and HTTPS filtering. In this work, we focus on four popular ISPs that cumulatively capture more than 95% of the Indian clients—Reliance Jio (Jio), Airtel, Vodafone-Idea (Vodafone) and ACT network and analyze the current state of Internet censorship and report how it has evolved over the past few years.

Identifying potentially blocked websites (PBWs): For assessing the extent of censorship in the said ISPs, we required a set of potentially blocked websites. Unlike other countries (like Greece [39]), India does not release an official black list [20]. Thus, we obtained the lists used by previous authors [20, 21], that they curated from different sources like court orders and various Internet sites (*e.g.,* Herdict [71] and Citizen Lab [72] *etc.*). The list contained 6491 unique domains.

We resolved these domains from a VPS in a non-censoring country but obtained the IP addresses corresponding to only 4619 domains. Next, from the same VPS, we attempted accessing these 4619 domains but ended up with only 2895 domains that replied with legitimate responses. Interestingly, for these domains both port 80 and 443 were open. We used these 2895 unique domains for our censorship tests. As per `Fortiguard` web filter [73], they spanned

across 53 different categories *e.g.,* pornography, illegal, streaming *etc.*

Vantage Points within ISPs: To perform our large-scale censorship studies, we used our own client machines connected to various ISPs. SIM cards of the three ISPs *viz.* Jio, Airtel and Vodafone were used. These machines were connected to the Internet through these SIM card, attached via USB dongles. Unfortunately, ACT ISP only provides broadband Internet but no mobile data connectivity. Thus we used ACT's residential network connections to perform our experiments.

To ensure that no third-party would be implicated for whistleblowing against censorship, we performed measurements ethically. Throughout our study, the hosts, mobile phones and SIM cards used, all belonged to the authors. We discuss in detail the ethical considerations in App. §5.1. We now present our approach to determine type of censorship (DNS, TCP/IP, HTTP and HTTPS) and mechanisms behind them.

4.1 DNS Censorship

In general, DNS resolution is the primary step for accessing any website. URL supplied to the client's browser is first resolved to its correct IP address. Thus often censors exploit this step and respond with an incorrect IP address. This leads to the website unavailability for the client. DNS censorship is usually achieved in two ways—*DNS poisoning* [74] and *DNS injection* [75]. DNS poisoning involves DNS resolvers that are under the censor's control, which respond with incorrect IP addresses when queried with censored websites' domains. However in DNS injection, a censorship middlebox positioned between the client and the resolver, intercepts the DNS query and forges fake DNS responses, spoofing the source IP of the resolver.

Identifying websites filtered using DNS censorship: For India, Yadav *et. al.* [20] confirmed DNS based censorship using the two heuristics. A censored ISP:

1. Either responses with bogon IPs [76], corresponding to the DNS query for the filtered domains.

2. Or, responses with an IP address belonging to its own AS. This was also later confirmed by Singh *et. al.* [21] that the Indian ISPs send forged DNS responses, each bearing an IP address from a set of their own fixed IP addresses.

Out of the four ISPs under consideration, we observed DNS censorship in only two, Airtel and ACT. In these, we observed the prevalence of only the above two heuristics for DNS censorship. In Airtel 1525, while in ACT 2435, websites were being censored.

DNS poisoning vs injection: After identifying the blocked websites due to DNS filtering, we investigated which DNS blocking technique was in place *i.e.,* DNS poisoning or injection. We began by identifying all open DNS resolvers of the ISP under consideration. We used DNS module of `zmap` [77] to scan the IPv4 address space of the said ISP (in a rate limited manner). Following the standard ethical practices [4], we only selected those resolvers that likely belonged to the ISPs infrastructure (ref. §5.1 for more details). This provided us 134 open resolvers in Airtel and 247 in ACT.

To identify the censoring resolvers, we sent 4619 DNS queries (corresponding to PBWs) to each of the DNS resolvers. Based on the aforementioned heuristics, we identified revolvers which responded with invalid IP addresses. We classified them as censored. To further discern DNS poisoning from injection, similar to [20, 48], we used, "Iterative Network Tracer" (INT) tool, that is a derivative of standard `traceroute` utility. Similar to `traceroute`, we sent our crafted DNS queries with increasing TTLs to the resolvers. If the DNS response arrived from any network hop other than the last one, it confirmed *injection*, otherwise it was DNS *poisoning*.

The ACT ISP: ACT only provides landline Internet to different residential and commercial establishments. Thus, using a few of our authors' residences (with ACT connection) as vantage points, we performed our tests. In total we had access to three such vantage points in densely populated Indian cities. In this ISP we observed both DNS poisoning and injection.

(i) DNS poisoning: At one location, we used INT and saw only DNS poisoning at our default resolver. Thus, from the same location, we performed our tests involving different resolvers (that belong to ACT). The resolvers which responded with incorrect IP addresses (as per previous heuristics) were considered poisoned. Previous authors [20] highlighted that different poisoned

resolvers maintain disparate lists of blocked websites. Thus, depending on the resolver selected, one would experience different sets of websites being censored. We also noticed similar trends—*e.g.,* for some poisoned resolvers, we observed more than 52% of the PBWs to be censored, whereas for others, this number was under 24%. In total $\approx 11\%$ of the resolvers were poisoned and each such resolver censored on average 50.6% (S.D. 6%) of the websites.

(ii) DNS Injection: From other two locations we used INT and observed only DNS injection. Interestingly, on sending INT DNS queries to different ACT resolvers, we obtained a forged DNS response (*e.g.,* bogon IPs) always from the 6^{th} hop (and never from the resolvers). This indicated that injector is intelligently positioned, such that it can intercept DNS queries sent to diverse set of resolvers (within ACT).

Next we asked questions like, are injectors placed intelligently so that they intercept paths to non-ACT resolvers also? If not, can one bypass these injectors by selecting some resolvers in other ISPs (or open resolvers)? To answer such questions, we curated the list of 100 resolvers belonging to different Indian and foreign ISPs, such that they are geographically distributed across the globe. We included open resolvers like $8.8.8.8$ and $1.1.1.1$ as well. We sent 2895 DNS INT queries, with PBWs, to these resolvers. Along the network paths to *all* such resolvers, for 1574 ($\approx 54\%$) PBWs we observed DNS injection from the same 6^{th} hop. This confirms the strategic placement of injectors that are hard to bypass by merely choosing resolvers in other ISPs.

This further indicated that clients residing in regions where injectors are deployed, could observe a stricter censorship, compared to the ones where no injectors are placed.

The Airtel ISP: Unlike ACT, Airtel offers both mobile and broadband Internet connectivity. We sent DNS INT queries for both type of connections to Airtel's resolvers and observed only DNS poisoning. In total $\approx 40\%$ of the resolvers were poisoned and each resolver blocked on average 25% (S.D. 12%) of the websites.

Effectiveness of DNS censorship: In both the ISPs we observed DNS censorship. However, in ACT we observed DNS injection, which seems to be an efficient form of censorship. This is because, our results indicate that injectors are placed at a few choke points, where they

intercept a large fraction of DNS traffic, which results in effective censorship. However, for some regions where injectors were not positioned, DNS poisoning was in place. In Airtel, although we observed DNS poisoning only, but it was for a significant fraction (40%) of resolvers.

4.2 TCP/IP Filtering

ISPs can prevent clients from connecting to the IP addresses of blocked websites. They could simply install firewall rules on the routers to drop all packets destined to the blacklisted IP addresses. Additionally, they can also restrict traffic destined to specific services running on such IP addresses. For instance, they could specifically block HTTP packets (bearing destination TCP port 80) for certain IP addresses.

Thus, we used a simple technique to detect TCP/IP blocking. We attempted multiple TCP 3-way handshakes with the PBWs. If TCP handshake failed in all the attempts, it implied TCP/IP filtering. Previous authors [20] tested for only TCP port 80, whereas we tested for port 443 as well. However, we never obtained this form of censorship, for the tested ISPs.

4.3 HTTP and HTTPS Filtering

Various ISPs use deep packet inspection (DPI) to filter Internet traffic by placing sophisticated middleboxes within their networks [18, 29, 32]. Such middleboxes intercept the Internet traffic to identify the plain-text domain names, that they wish to censor. For HTTP, they may inspect the `Host` field of the HTTP GET request. But in HTTPS, they may inspect the TLS ClientHello messages that bear `Server Name Indication` (SNI) extension field [13, 21, 78]. Both these fields bear the domain name in plaintext.

Detecting censorship middleboxes: In order to detect the presence of such middleboxes, we used INT (as already explained in §4.1). From our vantage point within the ISP, for each of the PBW, we first completed TCP 3-way handshake (on port 80) and then sent HTTP GET request bearing the PBW in the `Host` field, with increasing TTLs. If the HTTP response (potentially

bearing the censorship notification message) arrived from any network hop other than the last one, it confirmed the presence of a middlebox.

For identifying HTTPS filtering middleboxes, we repeated the above steps (using INT). But after successful TCP 3-way handshake (on port 443), we sent TLS ClientHello bearing a PBW in the SNI field. If we received TCP connection termination packets (*e.g.,* RSTs), from any intermediate hop other than the last one, we confirmed the use of a middlebox.

We observed both HTTP and HTTPS filtering *i.e.,* responses from middleboxes in the three prominent ISPs, *viz.* Jio, Airtel and Vodafone. We summarize our findings in Table 4.1.

ISP	**HTTP Blocking** (% of PBWs)	**HTTPS Blocking** (% of PBWs)	**Type of Middlebox Used**
Jio	68.35	63	WM
Airtel	72.67	70.43	IM
Vodafone	70	62.6	IM

Table 4.1: HTTP and HTTPS filtering in different ISPs.

Mechanics and evolution of HTTP(S) censorship: After confirming the presence of such filtering, we were interested in analyzing the detailed mechanics of web censorship. Previous authors [20] explored the presence of two types of middleboxes that inspect HTTP traffic in India—interceptive (in-path) and wiretapping (on-path) middleboxes.

Interceptive middleboxes (IMs) are inline network elements that intercept requests from the client to the server, and drop them in transit. Thus, these requests never reaches the server, and clients do not receive actual web responses from the blocked server. Further, after intercepting the clients' requests, IMs respond with connection termination packets (*e.g.,* FIN) to achieve censorship. These connection termination packets signal the client's browser to disconnect from the server.

However, wiretapping middleboxes (WMs) are devices connected to the ISPs routers via wiretaps. It receives copies of all packets traversing the router, rather than intercepting their flow. For example, when filtering HTTP traffic, WMs inspect the copies of HTTP GET request that they receive. If a blocked domain is present in the `Host` field, WMs send connection termination

packets to the client. These packets signal the client's browser to tear down the TCP connection with the server. Notably, with WMs, the original requests are left unaltered, and they elicit the actual web responses from the blocked sites as well. Thus at the client there is always a "race-condition" between the connection termination packets and the actual web responses. If the latter arrives first, the website is rendered normally in the client's browser, else the it suffers censorship. Thus, WMs seem less efficient than IMs.

To identify the type of middleboxes, we adopted the methodology similar to the one presented in [20]. From our vantage points we sent requests (HTTP GET and TLS ClientHello) bearing PBWs to our VPS host. In case of WMs, we observed that the original requests reaching our VPS and connection termination packets (*e.g.,* RST) at our vantage point. However, for IMs, we only received RST (or FIN) at our vantage points, but not the original requests to our VPS.

Interestingly, our findings suggest that Indian ISPs have surreptitiously evolved by employing more effective censorship techniques. For instance, in 2018, Airtel had WMs. However, in our research we presently find only IMs, and no WMs. Earlier WMs were censoring only HTTP traffic, but the newer IMs censor not only HTTP, but HTTPS as well. We observed the similar type of IMs within Vodafone ISP (censoring both HTTP and HTTPS traffic). The only exception is Jio, that still uses WMs for achieving both HTTP and HTTPS filtering. In 2020, Singh *et al.* also reported that both these types of traffic were being filtered by Jio, but they did not report the type of middleboxes used. Further, they did not observe HTTPS filtering in neither Airtel, and nor Vodafone. Table 4.2 summarizes our findings.

ISP	**2018**				**2021**			
	HTTP		**HTTPS**		**HTTP**		**HTTPS**	
	WM	**IM**	**WM**	**IM**	**WM**	**IM**	**WM**	**IM**
Jio	✓	✗	✗	✗	✓	✗	✓	✗
Airtel	✓	✗	✗	✗	✗	✓	✗	✓
Vodafone	✓	✗	✗	✗	✗	✓	✗	✓

Table 4.2: Evolution of HTTP(S) filtering in Indian ISPs.

Efficacy of HTTP/HTTPS censorship: Previous authors [20] proposed two metrics that were seemingly effective in gauging the extent of censorship—*coverage* and *consistency*.

Coverage is the fraction of ISP's router-level paths that are intercepted by the middleboxes (also called *poisoned-paths*). Whereas, **consistency** is the average of fraction of PBWs blocked along each poisoned path. Ideally, if the ISP is consistent about censorship, one should observe the same set of PBWs to be filtered along all its poisoned paths.

To compute coverage and consistency, we require diverse router-level paths within an ISP. Thus we curated a list of different popular unfiltered domains (*e.g.,* Alexa top websites, university websites, government websites *etc.*) as suggested in [20]. In total we selected 500 such websites hosted across 226 unique global ASes.

Running `traceroutes` from our vantage point (in the ISP under consideration), to these 500 websites, yielded diverse router level paths. After obtaining the router-level paths and the corresponding hop numbers to these websites, we ran INT using HTTP(S) packets to detect the middleboxes.

Running INT requires establishing TCP connection between the vantage point and the Internet destinations. Thus, we first established the TCP connection with one of the said 500 websites. To detect HTTP censorship middleboxes, we sent regular GET requests with the `Host` field pointing to PBWs. For HTTPS, we sent a crafted TLS ClientHello with a PBW as SNI field to the said website. We set the TTL value to the penultimate hop of the destination. This ensured that such packets were seen by the middlebox, but not the server. We recorded the corresponding censorship responses (if present). We repeated these steps by sending all the PBWs in SNI, to the same popular unblocked website. Thereafter, we repeated the entire process for each of the 500 popular domains as destinations. From vantage points in each ISP, we sent ≈ 2.9 Million requests that bore the PBWs' domains in plaintext *i.e.,* 500 (popular websites) x 2895 (PBWs) x 2 (HTTP GET and TLS ClientHello requests). These measurements were conducted across a period of more than one year to avoid overloading the network and the Internet destinations.

We observed similar results for both HTTP and HTTPS tests. For Airtel, we observed that out of 500 different paths, along 350 paths we observed atleast one instance of censorship, *i.e.,* coverage is about 70%. Corresponding to each of these 350 poisoned paths we observed on an average $\approx 90\%$ (S.D. 21%) of PBWs to be blocked (consistency). But, for Reliance Jio and

Vodafone we observed very low coverage values. For Jio, the coverage was less than 7% and consistency was $\approx 21\%$ (S.D. 20%). For Vodafone, coverage was only 6.4% and consistency was 70.25% (S.D. 31%).

4.4 Anti-censorship solutions

Evading DNS censorship: We now describe how we bypassed DNS poisoning and injection.

Evading DNS poisoning: Bypassing DNS poisoning was simple. Rather than using ISPs corrupted resolver, we simply used some open resolver *e.g.,* $8.8.8.8$. With such open resolvers we evaded DNS poisoning in all instances. Moreover, resolvers in uncensored countries could also be used for the same.

Evading DNS Injection: The strategy to evade poisoned DNS resolvers did not work with injectors. Thus to circumvent injectors, we attempted to identify if they were interceptive or wiretapping. In case of wiretapping, there would be two responses at the client, one from the injector and other from the actual DNS resolver. One could simply ignore the DNS responses from the injector bearing the bogon IPs, and accept the legitimate responses sent by the actual resolver. But with interceptive, this evasion would not work, as one would never receive the legitimate responses from the resolver.

When we sent DNS queries with blocked domains to various ACT resolvers, we always saw only one DNS response bearing the bogon IP, indicating the presence of IM. To ascertain the same, from our vantage point (in ACT), we sent DNS queries with filtered domains to our VPSs, but we did not receive them there. This confirmed that the DNS injector was indeed *interceptive*.

To evade the injector, we used numerous techniques to obfuscate the domain name in DNS query such that it is correctly interpreted by the resolver but not by the injector. We first tried appending white spaces before (or after) the domain name. Next, we sent multiple DNS queries within a single request *etc.* Finally, we sent the fragmented DNS query packets, *but nothing worked.*

Further, it is known that censorship middleboxes maintain network flow information [18, 32]. Thus, we hypothesised that DNS injector could also be maintaining UDP four-tuple information (source IP, source port, destination IP and destination port). Similar to [18, 32], we sent a query for an uncensored domain (*e.g.,* `unblocked.com`) to a resolver. Using the same four-tuple used by this request, we immediately crafted and sent a fresh query to the same resolver, but this time for a censored domain (*e.g.,* `blocked.com`). We expected that injector might not inspect the second query as it bore the same four-tuple. Unfortunately, the injector intercepted our second request and responded with a bogon IP. We tried several permutations, *e.g.,* we first sent `blocked.com` followed by `unblocked.com`, and then again `blocked.com`; but none worked, indicating that injector is inspecting UDP packets with dst. port 53.

Lastly, it is known that DNS resolvers also respond to queries on TCP port 53 in accordance with `RFC 7766` [79]. Thus we sent DNS queries (with censored domains) to resolvers over TCP port 53. These were the same 100 resolvers that we tested earlier for identifying the injectors (ref. §4.1). We saw legitimate responses from all the resolvers, evading the injector. This confirmed that injector was only inspecting UDP packets with destination port 53, and not TCP.

Evading HTTP(S) censorship: To bypass middleboxes causing HTTP and HTTPS censorship, we adopted techniques used in [20]. All previously mentioned anti-censorship approaches still work.

Evading Wiretapping Middleboxes: Previous authors [20] demonstrated that by simply changing the case of `Host` keyword (*e.g.,* `HOST`, `HoSt`, *etc.*) in HTTP GET request, one could evade WM (within the Indian ISPs), but would still receive response from the actual blocked website. Moreover, other simple schemes like adding white spaces before (or after) the keyword `Host` in the GET request, allows GET request to go undetected by the WM, at the same time, also elicits responses from the blocked website. Additionally, at the client if we simply drop the connection termination packets (*e.g.,* RST, FIN *etc.*) from the middlebox, we can evade censorship.

We confirm that all these techniques still work for evading WMs. Interestingly, we also report for the first time, that same techniques work for bypassing HTTPS filtering as well.

Merely adding white spaces before (or after) SNI extension of the ClientHello, or changing its case, hoodwinks the middlebox, and also yields ServerHello from the filtered website. We also noticed that all responses from WM have an IP-ID value of 242. We used this observation to disambiguate responses of middlebox from the actual webserver.

Evading Interceptive Middleboxes: Unlike Yadav *et al.*'s [20] observation, we confirm that all circumvention approaches for WMs, presently work for IMs as well. Additionally, we also tested the techniques proposed to bypass Great Firewall of China [18, 32]. Such approaches involve sending crafted packets to disrupt the TCP states maintained by the middleboxes.

It is known that after the TCP 3-way handshake with the blocked website, if we send HTTP GET request (or TLS ClientHello) with blocked domains, the middleboxes get triggered and drop them. But, immediately after the TCP handshake, similar to [18, 32], if we send requests bearing *uncensored domains* with TTL values such that, hop count of middlebox $<$ TTL $<$ final hop, the middlebox would assume that request is destined to the unblocked domain and would purge out the TCP state information. Also, these requests would expire even before reaching the server. Thereafter, requests bearing the blocked domain (with appropriate TCP header fields) with standard value of TTL, *i.e.,* 30, would go undetected by the middlebox and yield a response from the blocked site. We confirm that this scheme works for evading IMs.

On the use of ESNI: We also explored the possibility of using Encrypted-SNI (ESNI) to circumvent HTTPS filtering. But we found that currently only 187 (out of 4691) domains support the use of ESNI (ref. 6.8 for details). As expected all of them are hosted on Cloudflare as it was instrumental in developing ESNI [80]. Our empirical results reveal that use of ESNI is scarce. We hope that proposals like Encrypted Client Hello [81] may soon be standardised and would be largely adopted by the website publishers and the browsers alike [82].

The curious case of SNIs: We further conducted some more experiments specifically to evade HTTPS censorship (both for WMs and IMs). The goal of these experiments was to assess how strictly various web servers inspect the SNI extension before responding with TLS ServerHello. If most web servers ignore the SNI field, then TLS ClientHello with empty SNI fields would not only elicit ServerHello from the blocked webserver, but it would go undetected by the middlebox.

For all three ISPs (Jio, Airtel and Vodafone), when we sent TLS ClientHello packets with empty SNI fields (or with unblocked domain like `google.com`), we bypassed the censorship. At the same time, we received the ServerHello from the blocked web server and were able to complete the TLS handshake, and accessed the website. Surprisingly, web servers associated with more than 70% of the PBWs, do not inspect SNI field. For the remaining 30% of the websites, which require the exact domain name in the SNI, we can use any of the earlier mentioned HTTPS circumvention techniques.

Overall, we evaded all types of web censorship using intelligently crafted packets.

Chapter 5

Experimental Setup

We now present the details of how we conducted the tests to analyze both app and web censorship.

Setup for app censorship tests: We installed apps on our own mobile phones (running latest Android version 10 and iPhone iOS version 14). Likewise, our overseas contacts (authors of this work) used their mobile phones (using the same Android version) to install such apps and performed tests.

For analyzing the app traffic we required installing `mitmproxy` [69], that helps intercepting, decrypting and analyzing TLS traffic. This required installation of self-signed certificates within the mobile phones. However, the apps in Android versions newer than 7 cannot use certificates signed by untrusted issuers. This is called as *certificate pinning* [83]. Thus to bypass certificate pinning, only when analyzing app traffic, we used an older version of the Android (*i.e.,* 6). The `mitmproxy` was run on a Linux host running Ubuntu 20.04.1, equipped with quad core x64 processor, and 8 GB of RAM.

Setup for web censorship tests: As previously mentioned, for our tests we purchased our own Indian SIM cards. These SIM cards were installed in USB dongles or mobile phones and connected to our Linux machines running Ubuntu 20.04.1, equipped with a quad core x64 CPU and 8 GB RAM. For measurements in residential networks like ACT and Airtel Broadband, we used hosts (Raspberry Pi v3.0) placed in the authors' home that connected to the Internet via such ISPs.

Our experiments involved several tests that required sending multiple requests destined to various Internet destinations. To that end, from our vantage points we spawned a maximum of 4 parallel threads at any point of time. Each thread managed a separate TCP connection with the Internet destinations. Thus, to not overload the web servers, we first established TCP connection, sent only one measurement packet (HTTP GET or TLS ClientHello) with increasing TTL (till the penultimate hop) and eventually terminated the connection. Before the next measurement we provided a "cool-down" period of 5 seconds and then repeated the same process.

Overall our study was conducted for a span of 15 months, starting from January 2020.

5.1 Ethical Considerations

Censorship measurement studies often require accessing blocked websites that are deemed objectionable by the nation sates. Thus, accessing these websites from the censored country may raise some important ethical considerations. Thus, for this study, we carefully devised our experiments following the recommendations given by Belmont [84] and Menlo [85] reports.

As already mentioned (in §4), we obtained access to Indian ISPs, by purchasing their SIM cards. We were extremely careful at this step; only those author(s) of this study, who are citizens of India, purchased the SIM cards. No third-party was involved this step. Later, all our experiments involving mobile Internet connectivity were performed using these SIM cards only. Similarly, for tests involving ISPs that cater residential landline networks, we required residential vantage points. For this we again used the residential network of author(s) of this study.

Further, while conducting DNS censorship studies, we required scanning various ISP prefixes for open DNS resolvers. We only selected those that belong to ISPs' infrastructure and avoided non-ISP resolvers. This minimizes the risk of placing the maintainers of non-ISP resolvers in jeopardy.

Thus, from the list of all available open DNS resolvers in the ISP under test, we identify those that are likely authoritative nameservers for any domain. As suggested in [29], we performed a reverse DNS PTR lookup and selected only those resolvers whose PTR began with the regular

expression “ns[0-9]+nameserver[0-9]”.

Our experiments involved sending multiple packets (like DNS requests, HTTPS requests, and TLS client hello) to various Internet destinations. We were extremely cautious to rate limit our measurements to reduce the network load as well as on the end destinations. In all our experiments, involving HTTP and HTTPS censorship tests, we first completed a TCP 3-way handshake with the web server and then sent the subsequent requests with increasing TTLs. Once we recorded the censorship response (if present), we gracefully closed the connection with the web server.

Moreover, our experiments for coverage and consistency, required sending HTTP requests and TLS client hello bearing potentially blocked domains to the unfiltered websites. In such cases, to reduce the load on the unfiltered website, we ensured that the requests were sent with TTL value set to the penultimate hop. Thus, webservers of unfiltered website would not observe any web (or TLS) request that was not intended for it.

Chapter 6

Insights and Analysis

6.1 Topologically aware positioning of middleboxes

In Jio and Vodafone we observed a large fraction of the PBWs ($> 60\%$) to be blocked (ref. Table 4.1). But at the same time, we also record poor coverage values ($< 7\%$), when destinations were the 500 unblocked sites. This led us to a few pertinent questions—if the coverage is low *i.e.,* middleboxes are intercepting only a few router level paths, why did we observe such a large number of websites (> 1800) being blocked within these ISPs? Have these ISPs intelligently positioned the middleboxes so that they likely intercept router level paths to PBWs and not the unblocked websites? Such topologically aware placement of middleboxes would greatly reduce the inspection load on the middleboxes as they would have to inspect less traffic (*i.e.,* mostly destined to PBWs).

To answer these questions, we devised a simple experiment. In both Jio and Vodafone, we first sent the previously confirmed 1800 known blocked websites (BWs) in the SNI field of ClientHello packets to the previously selected 500 *unblocked* websites. We then recorded the fraction of these BWs that were blocked along the paths to each of the unblocked websites.

Secondly, we repeated the same experiment but with ClientHello packets destined to randomly selected 500 *blocked* websites as destinations. We again recorded the fraction of blocked websites along each path.

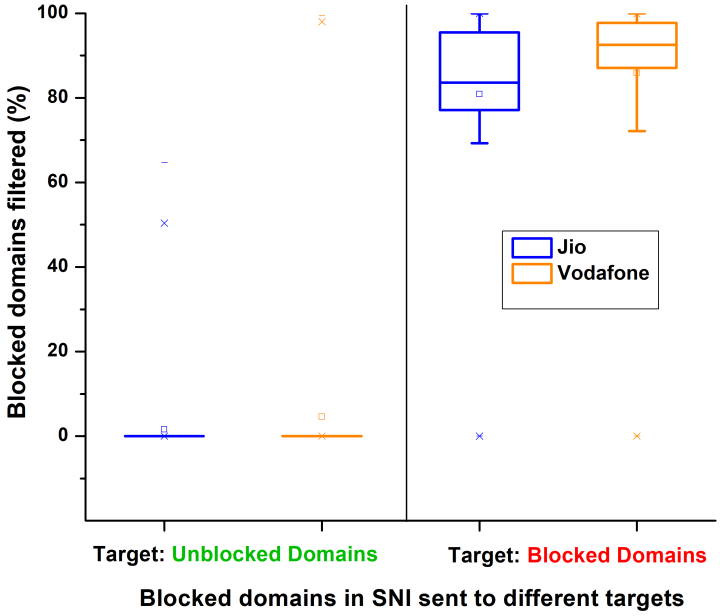


Figure 6.1: Network partitioning within Jio and Vodafone: Fraction of PBWs (as SNIs) that were censored along the ISPs paths. We observe disparate blocking for two different set of destinations—500 benign and 500 blocked destinations.

To our surprise we observed that in Jio, along the paths to 500 unblocked websites we rarely observed any blocking (barring a few exceptions); on an average, 1.6% of the 1800 BWs were filtered along each of these paths. But at the same time, we found a large majority ($\approx$ 80% of the 1800 BWs) to be blocked, along the paths to the 500 BWs. We repeated the above experiment by selecting a different set of BWs (from the known 1800 BWs), as well as unblocked websites as destinations. Yet again, we observed the similar trend. We noticed the similar phenomenon in Vodafone as well. Fig. 6.1 depicts the two scenarios for both the ISPs. Thus, depending upon the destinations, we observe completely different censorship trends within the same ISP.

To further obtain more insights about ISP's network paths and positioning of middleboxes, we first ran `traceroutes` from our vantage points (within these two ISPs) to the set of 500 unblocked domains, as well as to 1800 BWs.

We observed an interesting pattern. In Vodafone, from the same vantage point, `traceroutes` paths to both set of domains (blocked and unblocked) share a significant segment of the network path, *i.e.,* upto the first four network hops. However fifth hop onwards the paths change. Further, three IP addresses, corresponding to the fifth, sixth and seventh hop of the `traceroute`

paths to the blocked domains, always remained the same. However, for unblocked websites as destinations, we never observed the said triplet.

We observed the similar trend in Jio as well, just that the `traceroute` paths changed after the seventh hop. Moreover, similar to Vodafone, three IPs corresponding to the eighth, ninth and tenth hop remained the same for all the paths corresponding to the blocked domains alone.

Interestingly, using INT, we found that in Vodafone and Jio, the middleboxes appeared at the sixth and ninth hop respectively. This strongly indicates that middleboxes are intelligently positioned within these ISPs such that they largely intercept paths towards the blocked websites and not the unblocked ones. However, we did not observe this interesting trend in Airtel.

One may further ask that how ISPs are identifying the locations for such intelligent positioning of middleboxes? One plausible explanation could be that if the filtered websites are hosted within a few ASes (or prefixes), the middleboxes could be positioned along with those routers that route traffic to handful of such destinations. We identified that two ASes *viz.* `Teaminternet` (ASN 61969) and `Trellian` (ASN 133618) together host $\approx 50\%$ of the blocked websites. This indicates that this could be one of the factors in intelligent positioning of the middleboxes. Although, the exact answer can not be ascertained without the cooperation from the ISPs.

6.2 Censorship disparity between residential clients and VPSs

It has been reported [5] that censorship is more prominent for residential networks, in comparison to VPSs (and cloud hosts). Thus to study the same, we accessed PBWs from three popular hosting services (in India)—Digital Ocean (DO), Microsoft Azure, and Amazon AWS. We also observed the similar disparities. From a residential network (e.g., Airtel), we observed more than 70% of PBWs to be filtered. However, for the aforementioned hosting services, we observed comparatively less fraction of websites to be blocked. In Azure and AWS we practically observed no censorship, whereas in DO we observed 40% of the PBWs to be blocked. Interestingly, from the DO VPS, using HTTP(S) INT, we confirm that censorship notification-cum-disconnection packets (for the blocked websites) were sent from IP addresses belonging to Airtel (the upstream

provider of DO). Furthermore, in the `traceroute` paths to the remaining 60% PBWs, we did not observe Airtel as the upstream provider. This clearly indicates that DO does not itself filter the websites. Rather, clients suffers censorship due to its upstream provider. To conclude, there is a clear disparity in censorship for residential networks, compared to VPSs.

6.3 The apps unavailable in official play stores in non-censoring countries

In §3, we mentioned that our overseas contacts (residing in uncensored countries) were also unable to find 60 apps that were blocked in India. Thus, we assumed that they were likely defunct, else they would be available atleast in uncensored countries.

However, it could be argued that some of these apps might be functional yet unavailable in uncensored countries. This is because, some of these apps may have been launched specifically for Asia (or India). Thus, to confirm that this was not the case, and 60 apps were likely defunct, we searched them on third-party sources *e.g.,* `apkmirror.com`. Barring a few, the installation packages of most of these apps were unavailable on such sites as well. For the few that we found, they were last updated around four to five years ago. This indicated that they were no longer operational. Our overseas contacts installed them and confirmed that they were unable to access them. Thus, we ignored these 60 apps and focused on the remaining 160 apps.

6.4 Why app publishers are filtering Indian users

The ban on Chinese apps resulted in the removal of 220 apps from the official apps stores in India and among them 24 apps are restricting access to potential Indian users even when installed. Interestingly, many of these app publishers do not operate from India, yet they adhered to the ban and restricted their services in India. There were anecdotal evidences that the app publishers were hoping that the ban is temporary [86], and thus they obliged with the ban. Some app companies were in communication with the Indian government and were awaiting their feedback and approvals for relaunching their apps [87].

However, in January 2021, the government imposed a permanent ban on all the 220 apps (ref. §3.2). Even then the apps publishers not only continued the filtering, but also imposed stricter censorship for Indian users, while the ISPs continue to have no role in this. Before the ban, 8 apps were using only SIM based censorship, but after the ban 7 of them started adopting IP geo-blocking as well. The precise reason for app publishers (and not the ISPs) filtering the Indian users remains unclear.

6.5 Collateral damage due to disabled Indian edge-server

In §3.2, we reported that one app ChessRush, has removed content from edge-servers that serve requests from India. This may result in collateral damage to the end-users in the neighbouring countries who could unfortunately be mapped to the said edge-servers (by the CDN). Thus they may not be able to access the contents even though the restriction is not imposed for them. This could be one reason why almost all apps do not use this technique to censor Indian users.

6.6 SIM based blocking and the scope of "offline censorship"

The eight apps (ref. §3.2) that suffered SIM based blocking fetched the country name from the SIM card. Later, they sent this information to their respective app servers and in response, servers prevented users from accessing the services. However, this censorship does not necessarily require any intervention from servers and therefore could be enforced without Internet. The application can be implemented in manner that the apps deny services to end-users with devices bearing SIM cards of censoring countries. Thus SIM based censorship can be used for "offline censorship". During our analysis we found identified 41 apps that do not require Internet connectivity to work. These include multimedia players, image viewers, file sharing apps *etc*. Presently, apart from their removal from the official app stores, these applications were fully functional, when fetched from third-party sites. However, in future they could also be censored by the app publishers using this mechanism.

6.7 Obtaining banned apps

Google and Apple app store accounts provide an option to select one's country. When setting up of phone for the first time, the app stores automatically detect the country using IP geo-location of the device [88]. But when we access these app stores over VPN, we get the option to change the country settings in the store to the country of the VPN end-point. Thus, using VPNs, when we changed the app store's country setting from India to any non-censoring country, we found the banned apps. After re-configuring the country setting, we were able to directly download the apps from the official app stores, without requiring VPNs.

6.8 The use of ESNI

To encrypt the plain-text SNI filed of TLS Client Hello, we require a public key from the website publisher. To obtain this, we sent a DNS TXT query targetted towards `_esni.DomainName`. We achieve this using the command, `dig _esni.DomainName TXT`. We observed that 513 websites (out of 4691 PBWs) have associated public keys to encrypt SNI and amongst them only 187 supported TLS 1.3. Thus we are able to access only 187 websites using ESNI.

Additionally the latest version of Firefox (85) does not support ESNI. In our experiments we used the older version 64 to test the support of ESNI by the PBWs.

Chapter 7

Conclusion

Internet censorship has been promulgated by many nations-states in the past. In this work we focused on a democratic country India to analyze the current state of affairs of Internet censorship therein. Our research reveals a novel form of "three-tiered" mobile app censorship, with every tier increasing the censorship sophistication. Notably, India does not use the traditional model of censorship for blocking apps—*i.e.,* filtering performed by the ISPs. Rather, following the orders of the Indian government, app publishers are themselves filtering the Indian users at their servers. They achieve server-side censorship either by geo-blocking the clients based on source IP or by identifying Indian users using the country codes fetched from the SIM cards.

Furthermore, through our 15 month-long measurement study, we highlight some alarming observations regarding web censorship in India. Presently, the major ISPs are not only censoring HTTP, but also the HTTPS traffic. These ISPs have deployed sophisticated middleboxes that employ deep packet inspection to filter traffic destined to blocked domains. Interestingly, we observed in some ISPs, middleboxes appear in relatively fewer network paths, but still they censor a large number of websites. Additionally, for the first time we report the use of DNS injection in India as well. Finally, we show how one could circumvent all types of aforementioned web censorship schemes, using only intelligently crafted packets.

Bibliography

[1] "United nations general assembly, human rights council thirty-second session, third item," https://www.article19.org/data/files/Internet_Statement_Adopted.pdf.

[2] J.-P. Verkamp and M. Gupta, "Inferring mechanics of web censorship around the world." in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[3] W. Scott, T. Anderson, T. Kohno, and A. Krishnamurthy, "Satellite: Joint analysis of CDNs and network-level interference," in *USENIX Annual Technical Conference (ATC)*, 2016, pp. 195–208.

[4] R. Sundara Raman, P. Shenoy, K. Kohls, and R. Ensafi, "Censored planet: An internet-wide, longitudinal censorship observatory," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 49–66.

[5] A. A. Niaki, S. Cho, Z. Weinberg, N. P. Hoang, A. Razaghpanah, N. Christin, and P. Gill, "IClab: a global, longitudinal internet censorship measurement platform," in *IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 135–151.

[6] J. Dalek, B. Haselton, H. Noman, A. Senft, M. Crete-Nishihata, P. Gill, and R. J. Deibert, "A method for identifying and confirming the use of url filtering products for censorship," in *Proceedings of Internet measurement conference*, 2013, pp. 23–30.

[7] R. Clayton, S. J. Murdoch, and R. N. Watson, "Ignoring the Great Firewall of China," in *International workshop on privacy enhancing technologies*. Springer, 2006, pp. 20–35.

[8] P. Winter and S. Lindskog, "How the Great Firewall of China is blocking Tor," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[9] R. Ensafi, P. Winter, A. Mueen, and J. R. Crandall, "Analyzing the Great Firewall of China over space and time," *Proceedings on privacy enhancing technologies*, vol. 2015, no. 1, pp. 61–76, 2015.

[10] J. Griffiths, *The Great Firewall of China: How to build and control an alternative version of the internet*. Zed Books Ltd., 2019.

[11] J. Beznazwy and A. Houmansadr, "How China detects and blocks shadowsocks," in *Proceedings of the Internet Measurement Conference*, 2020, pp. 111–124.

[12] Anonymous, A. A. Niaki, N. P. Hoang, P. Gill, and A. Houmansadr, "Triplet censors: Demystifying great firewall's DNS censorship behavior," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*. USENIX Association, Aug. 2020.

[13] Z. Chai, A. Ghafari, and A. Houmansadr, "On the importance of encrypted-sni (ESNI) to censorship circumvention," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2019.

[14] J. Holowczak and A. Houmansadr, "Cachebrowser: Bypassing Chinese censorship without proxies using cached content," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 70–83.

[15] A. Dunna, C. O'Brien, and P. Gill, "Analyzing China's blocking of unpublished tor bridges," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2018.

[16] R. Ensafi, D. Fifield, P. Winter, N. Feamster, N. Weaver, and V. Paxson, "Examining how the great firewall discovers hidden circumvention servers," in *Proceedings of Internet Measurement Conference*, 2015, pp. 445–458.

[17] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson, "An analysis of China's great cannon," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2015.

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

[19] D. Gosain, A. Agarwal, S. Shekhawat, H. B. Acharya, and S. Chakravarty, "Mending wall: On the implementation of censorship in India," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2017, pp. 418–437.

[20] T. K. Yadav, A. Sinha, D. Gosain, P. K. Sharma, and S. Chakravarty, "Where the light gets in: Analyzing web censorship mechanisms in India," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 252–264.

[21] K. Singh, G. Grover, and V. Bansal, "How India censors the web," in *12th ACM Conference on Web Science*, 2020, pp. 21–28.

[22] "China bans mobile apps," https://www.bbc.com/news/technology-55230654.

[23] "Iran bans signal messaging app," https://www.theverge.com/2018/1/2/16841292/iran-telegram-block-encryption-protest-google-signal.

[24] "Iran bans messaging apps," https://en.radiofarda.com/a/iran-lawmakers-aim-to-fully-ban-all-foreign-messaging-apps/30802448.html.

[25] "Apps banned in Russia," https://www.comparitech.com/blog/vpn-privacy/websites-blocked-russia/.

[26] "Tiktok blocked in India," https://www.nytimes.com/2020/06/29/world/asia/tik-tok-banned-india-china.html.

[27] "Press Information Bureau (Government of India) officially confirms the ban of 59 Chinese apps," https://pib.gov.in/PressReleseDetailm.aspx?PRID=1635206.

[28] "Total downloads of Tiktok app," https://www.theverge.com/2020/4/29/21241788/tiktok-app-download-numbers-update-2-billion-users.

[29] R. Ramesh, R. S. Raman, M. Bernhard, V. Ongkowijaya, L. Evdokimov, A. Edmundson, S. Sprecher, M. Ikram, and R. Ensafi, "Decentralized control: A case study of russia," in *Network and Distributed Systems Security (NDSS) Symposium*, 2020.

[30] A. McDonald, M. Bernhard, L. Valenta, B. VanderSloot, W. Scott, N. Sullivan, J. A. Halderman, and R. Ensafi, "403 forbidden: A global view of CDN geoblocking," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 218–230.

[31] "The Indian telecom services performance indicators report," https://www.trai.gov.in/sites/default/files/QPIR_21012021_0.pdf.

[32] S. Khattak, M. Javed, P. D. Anderson, and V. Paxson, "Towards illuminating a censorship monitor's model to facilitate evasion," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[33] J. Jermyn and N. Weaver, "Autosonda: Discovering rules and triggers of censorship devices," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2017.

[34] "Indian government bans 220 apps," https://bit.ly/31ECpeA.

[35] J. Zittrain and B. Edelman, "Internet filtering in China," *IEEE Internet Computing*, vol. 7, no. 2, pp. 70–77, 2003.

[36] P. Gill, M. Crete-Nishihata, J. Dalek, S. Goldberg, A. Senft, and G. Wiseman, "Characterizing web censorship worldwide: Another look at the Opennet Initiative data," *ACM Transactions on the Web (TWEB)*, vol. 9, no. 1, p. 4, 2015.

[37] A. Chaabane, T. Chen, M. Cunche, E. De Cristofaro, A. Friedman, and M. A. Kaafar, "Censorship in the wild: Analyzing internet filtering in Syria," in *Proceedings of Internet Measurement Conference*. ACM, 2014, pp. 285–298.

[38] S. Aryan, H. Aryan, and J. A. Halderman, "Internet censorship in Iran: A first look." in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[39] V. Ververis, G. Kargiotakis, A. Filasto, B. Fabian, and A. Alexandros, "Understanding internet censorship policy: The case of Greece," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2015.

[40] G. Aceto, A. Montieri, and A. Pescapé, "Internet censorship in Italy: An analysis of 3G/4G networks," in *IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[41] Z. Nabi, "The anatomy of web censorship in pakistan." in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[42] G. Gebhart and T. Kohno, "Internet censorship in Thailand: User practices and potential threats," in *IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2017, pp. 417–432.

[43] H. B. Acharya, A. Ramesh, and A. Jalaly, "The world from Kabul: Internet censorship in Afghanistan," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2019, pp. 1061–1062.

[44] V. Ververis, T. Ermakova, M. Isaakidis, S. Basso, B. Fabian, and S. Milan, "Understanding internet censorship in europe: The case of spain," in *13th ACM Web Science Conference 2021*, 2021, pp. 319–328.

[45] J. Wright, "Regional variation in Chinese internet filtering," *Information, Communication & Society*, vol. 17, no. 1, pp. 121–141, 2014.

[46] J. R. Crandall, D. Zinn, M. Byrd, E. T. Barr, and R. East, "Conceptdoppler: a weather tracker for internet censorship." in *ACM Conference on Computer and Communications Security (CCS)*, 2007, pp. 352–365.

[47] Q. Yang and Y. Liu, "What's on the other side of the great firewall? Chinese web users' motivations for bypassing the internet censorship," *Computers in human behavior*, vol. 37, pp. 249–257, 2014.

[48] X. Xu, Z. M. Mao, and J. A. Halderman, "Internet censorship in China: Where does the filtering occur?" in *Proceedings of Passive and Active Network Measurement*. Springer, 2011, pp. 133–142.

[49] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Scott-Railton, R. Deibert, and V. Paxson, "China's great cannon," *Citizen Lab*, vol. 10, 2015.

[50] J. Knockel, L. Ruan, and M. Crete-Nishihata, "Measuring decentralization of Chinese keyword censorship via mobile games," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2017.

[51] S. Khattak, M. Javed, P. D. Anderson, and V. Paxson, "Towards illuminating a censorship monitor's model to facilitate evasion." in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[52] P. Levis, "The collateral damage of internet censorship by DNS injection," *ACM SIGCOMM CCR*, vol. 42, no. 3, 2012.

[53] Alice, Bob, Carol, J. Beznazwy, and A. Houmansadr, "How China detects and blocks shadowsocks," in *Proceedings of Internet Measurement Conference (IMC 2020)*. ACM, 2020.

[54] Anonymous, A. A. Niaki, N. P. Hoang, P. Gill, and A. Houmansadr, "Triplet censors: Demystifying Great Firewall's DNS censorship behavior," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2020.

[55] Z. Chai, A. Ghafari, and A. Houmansadr, "On the importance of encrypted-sni (ESNI) to censorship circumvention," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2019.

[56] "Rediff," http://www.rediff.com/computer/1999/jul05dawn.htm.

[57] F. Li, A. Razaghpanah, A. M. Kakhki, A. A. Niaki, D. Choffnes, P. Gill, and A. Mislove, "lib• erate,(n) a library for exposing (traffic-classification) rules and avoiding them efficiently," in *Proceedings of Internet Measurement Conference*, 2017, pp. 128–141.

[58] A. Filasto and J. Appelbaum, "OONI: Open observatory of network interference," in *USENIX workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[59] "List of blocked apps," https://pib.gov.in/PressReleseDetailm.aspx?PRID=1635206.

[60] "List of blocked apps," https://pib.gov.in/PressReleasePage.aspx?PRID=1650669.

[61] "List of blocked apps," https://www.pib.gov.in/PressReleasePage.aspx?PRID=1675335.

[62] A. McDonald, M. Bernhard, L. Valenta, B. VanderSloot, W. Scott, N. Sullivan, J. A. Halderman, and R. Ensafi, "403 forbidden: A global view of CDN geoblocking," in *Proceedings of Internet Measurement Conference 2018*, 2018, pp. 218–230.

[63] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan, "Mapping the expansion of Google's serving infrastructure," in *Proceedings Internet measurement conference*. ACM, 2013, pp. 313–326.

[64] J. Xue, D. Choffnes, and J. Wang, "CDNs meet CN an empirical study of CDN deployments in China," *IEEE Access*, vol. 5, pp. 5292–5305, 2017.

[65] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig, "Open connect everywhere: A glimpse at the internet ecosystem through the lens of the Netflix CDN," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 1, pp. 28–34, 2018.

[66] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye, "Analyzing the performance of an anycast CDN," in *Proceedings of Internet Measurement Conference*. ACM, 2015, pp. 531–537.

[67] D. Cicalese, D. Giordano, A. Finamore, M. Mellia, M. Munafò, D. Rossi, and D. Joumblatt, "A first look at anycast CDN traffic," *arXiv preprint arXiv:1505.00946*, 2015.

[68] F. Wohlfart, N. Chatzis, C. Dabanoglu, G. Carle, and W. Willinger, "Leveraging interconnections for performance: The serving infrastructure of a large CDN," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18, 2018, pp. 206–220.

[69] "Mitm proxy," https://mitmproxy.org/.

[70] "Jadx decompiler," https://github.com/skylot/jadx.

[71] "Herdict:help spot web blockages," http://herdict.org/.

[72] "List of potentially blocked websites in India—Citizen Labs," https://github.com/citizenlab/test-lists/blob/master/lists/in.csv.

[73] "Fortiguard webfilter," https://www.fortiguard.com/webfilter.

[74] S. Son and V. Shmatikov, "The hitchhiker's guide to DNS cache poisoning," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2010, pp. 466–483.

[75] Anonymous, "The collateral damage of internet censorship by DNS injection," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, Jun. 2012.

[76] "Bogon ip addresses," https://ipinfo.io/bogon.

[77] "Zmap module of DNS," https://github.com/zmap/zdns.

[78] W. M. Shbair, T. Cholez, A. Goichot, and I. Chrisment, "Efficiently bypassing SNI-based https filtering," in *2015 IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2015, pp. 990–995.

[79] "RFC7766," https://tools.ietf.org/html/rfc7766.

[80] "What is esni?" https://www.cloudflare.com/en-gb/learning/ssl/what-is-encrypted-sni/.

[81] "Rfc for encrypted client hello," https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-08.

[82] "Good-bye esni, hello ech!" https://blog.cloudflare.com/encrypted-client-hello/.

[83] "Certificate pinning," https://www.digicert.com/dc/blog/certificate-pinning-what-is-certificate-pinning/.

[84] T. L. Beauchamp, "The belmont report," *The Oxford textbook of clinical research ethics*, pp. 149–155, 2008.

[85] D. Dittrich, E. Kenneally *et al.*, "The menlo report: Ethical principles guiding information and communication technology research," US Department of Homeland Security, Tech. Rep., 2012.

[86] "App ban in India could be temporary," https://www.businesstoday.in/current/economy-politics/tiktok-denies-plans-for-legal-recourse-against-ban/story/408741.html.

[87] "Pubg to be relaunched in india," https://bit.ly/39viUcu.

[88] "How Google tracks the user location?" https://policies.google.com/technologies/location-data?hl=en-GB.