International Journal For Technological Research In Engineering
Volume 7, Issue 4, December-2019 *ISSN (Online): 2347 - 4718*

WEB CENSORSHIP

Shivam Goyal[1], Kshitiz Chauhan[2], Vaibhav Chauhan[3], Ms Gurpreet[4]
BMCEM, Haryana

ABSTRACT: In this work I would like to a present a study about the Internet censorship mechanism in India. I consolidated a list of potentially blocked websites from various public sources to assess censorship mechanisms used by major ISPs. To begin with, I would demonstrate that existing censorship detection tools like OONI are inaccurate. I developed various techniques and heuristics to correctly assess censorship and study the underlying mechanism used by these ISPs. We fortify our findings by adjudging the coverage and consistency of censorship infrastructure, broadly in terms of average number of network paths and requested domains the infrastructure censors. Our results indicate a clear disparity among the ISPs, on how they install censorship infrastructure.
KEYWORDS: Censorship, OONI, Deep Packet Inspection

I. INTRODUCTION

Free and open communication over the Internet, and its censorship, is a widely debated topic. It is not surprising that an overwhelming majority of prior studies on censorship activities and their mechanism, primarily center around overtly censorious nations like China.

In April this year, several Jio users were puzzled [1] to find that Reddit and Telegram were being blocked by the ISP. Around the same time, Sushant Sinha was perplexed to note that those using Jio connections were unable to access IndianKanoon.com, the legal database he founded and runs.

These experiences of arbitrary web censorship are the natural conclusion of an opaque legal framework that allows the Government of India to order ISPs to block certain websites for its users. The Central Government draws such powers from sections 69A and 79 of the Information Technology (IT) Act and the rules issued thereunder. Notably, the "blocking rules" issued under Section 69A [2] describe an executive-driven process, and further mandate the confidentiality of blocking orders issued to intermediaries. These rules have meant that it is next to impossible for netizens to know the complete list of websites blocked in India and the reasons for such blocking.

Pertinently, the blocking rules do not mandate ISPs to use any particular technical method to block websites. This has meant that Indian ISPs are at liberty to pick whatever filtering mechanism they wish, which has had implications for how internet users experience and circumvent web censorship.

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

Indian ISPs are majorly using two methods:

Domain Name System (DNS) based blocking

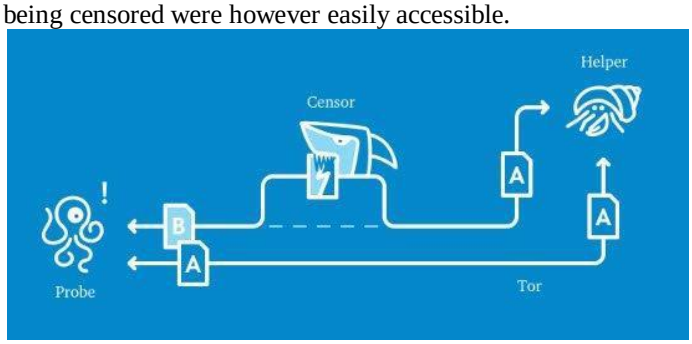
Users trying to access websites usually contact the ISP's DNS directory to translate a human-parseable address like 'example.com' to its network address '93.184.216.34'. Some ISPs in India, like BSNL and MTNL, respond with incorrect network addresses to the users' queries for websites they wish to block.

Hypertext Transfer Protocol (HTTP) header based blocking

HTTP is the most popular way to transmit web pages. Since classic HTTP communication is unencrypted, ISPs can monitor for the website's name that is attached
(the HTTP Host header field) to such traffic. ISPs like Jio, Airtel and Vodafone monitor this field for names of websites they wish to block, intercept such requests, and return anything they wish as a response.

II. OONI TOOL

Open Observatory of Network Interference (OONI) [5], is an open source tool under the Tor project and is designed to detect censorship. We executed OONI on five popular ISP networks, using the PBW, and recorded the results. To corroborate our findings we also manually checked the sites that were reported by OONI as being censored. To our surprise, we observed very few true positives. An exceedingly large number of sites which were reported as being censored were however easily accessible.



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

III. DNS BLOCKING

DNS based blocking can be achieved by (1) DNS poisoning [46]— whereby a corrupt(-ed) resolver replies with the

incorrect IP for specific DNS queries. (2) DNS injection [22] — where some middlebox between the client and resolver intercepts the DNS query and deliberately responds with a forged response bearing an incorrect IP address.

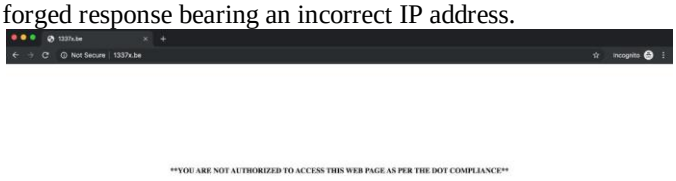


Image: The notice served by Jio (through HTTP-header based filtering and injected response) when a user tries to access a blocked website.

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

IV. HTTP FILTERING

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

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

V. SNI INSPECTION

Transport Layer Security (TLS) [3] is a cryptographic protocol for providing communication confidentiality and authenticity, commonly used for encrypting web traffic (as done in HTTPS). The SNI, defined first in RFC 4366 and then in RFC 6066, is an extension to TLS designed to facilitate the hosting of multiple HTTPS websites on the same server. While establishing a secure connection (a TLS Client Hello), a client just fills in the SNI attribute with the hostname of the website it wishes to connect to.

SNI, unfortunately, travels on the network in cleartext, i.e. network operators can not only see the websites you're visiting, but also filter traffic based on this information. The use of SNI inspection in state-directed web censorship was not very common until recently. Only this year, the use of SNI inspection to censor websites was documented in China and South Korea. In the Indian context, the aforementioned paper, the researchers note that in Indian ISPs they investigated (including Jio), they "observed fewer than five instances of HTTPS filtering which were actually due to manipulated DNS responses [...], and not because of SNI field in TLS [...]." However, as the next section documents, Jio is now in fact using SNI-inspection based filtering.

VI. THE TEST

To run our tests, we can take advantage of the fact that Google's server is configured to respond successfully to TLS connection attempts even if we send an SNI with a website's name that it does not host on that server.

Using OpenSSL's [4] s_client utility, we attempt to establish a TLS 1.3 connection with an IP address (216.58.196.174) corresponding to google.com. However, instead of specifying 'google.com' in the SNI, we specify a potentially blocked website (PBW) 1337x.be. openssl s_client -state -connect 216.58.196.174:443 - servername 1337x.be -tls1_3

Two important notes here:

- We are not connecting to the PBW at all! This simple approach is allowing us to rule out other censorship methods (like DNS, HTTP, and even IP/TCP-level blocking) from interfering with our results.
- We're using TLS 1.3 to make our connections. This is because in older versions of TLS, the server passes its certificate to the client in cleartext. ISPs may also be using that information to block websites if older TLS versions are used. Using TLS 1.3 allows us to ensure that ISPs are indeed using SNI inspection to block websites.

We notice that when we specify a PBW in the SNI, we receive a TCP packet with the RST (reset) bit set almost immediately after the connection is established, which closes the established connection. Of course, a plausible explanation could be that the Google server itself might be resetting the connection upon realising that it does not host the PBW. However, this is neither the expected behaviour as per RFC 6066, nor do we notice the server doing so in all cases where we specify a SNI for a website that it is not hosted on the server. For example, when we specify facebook.com as the SNI, not only are we able to complete the TLS handshake but we're also able to make subsequent requests to the server after completing the handshake (albeit receiving an expected "not found" error in response).

VII. ANTI-CENSORSHIP APPROACHES

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

Evading DNS poisoning: In order to circumvent poisoned DNS resolver, we tested using OpenDNS, Google's public DNS (8.8.8.8) and many other non-poisoned resolvers which belong to noncensorious countries like Ireland, Canada, and

Sweden. With each of them, we were able to bypass the DNS based censorship.

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

VIII. CONCLUDING REMARKS

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

REFERENCES

[1] https://in.reuters.com/article/us-india-internet-idINKCN1RF14D
[2] https://indiankanoon.org/doc/10190353/
[3] https://tools.ietf.org/html/rfc5246
[4] https://www.openssl.org/
[5] https://www.thewindowsclub.com/ooni-open-observatory-of-network-interference-project-tor
[6] https://github.com/citizenlab/test-lists/blob/master/lists/in.csv.
[7] http://rti.gov.in/.

