Towards a Censorship Analyser for Tor

Philipp Winter
The Tor Project & Karlstad University

Abstract

Analysing censorship incidents targeting popular circumvention tools such as Tor can be a tedious task. Access to censoring networks is typically difficult to obtain and remote analysis is not always possible. Analysis is however feasible if users behind the censoring networks are given the opportunity to help.

In this paper, we propose a lightweight censorship analyser for Tor which is meant to be run by volunteering users. The analyser automatically gathers relevant data and the final report is sent back to the Tor developers. Our design builds on existing software and should be easy to bundle and deploy.

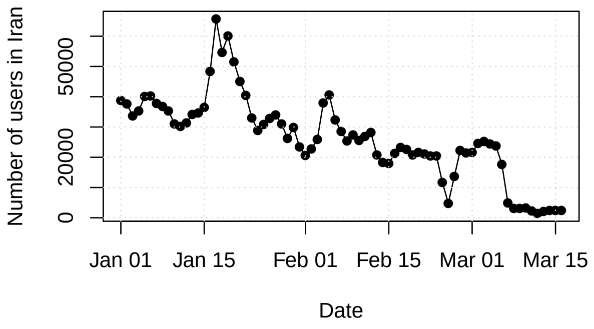

Figure 1: The amount of Iranian Tor users since the beginning of 2013 [4]. In February, the country started deploying a new strategy to block Tor.

1 Introduction

Anonymity and censorship resistance make a great couple. A good example for this is the Tor anonymity network [1]. While Tor's core competence is low-latency anonymity, over the years it developed a reputation as being a practical tool to safely circumvent censorship. Unsurprisingly, this reputation did not remain unnoticed among censors.

Due to Tor's popularity—Iran alone once accounted for more than 30,000 users as can be seen in Figure 1—censoring countries came up with techniques to identify and block Tor connections. As of March 2013, Tor was or is documented to be blocked in China, Iran, Syria, Ethiopia, the United Arab Emirates and Kazakhstan [2]. And these are only the documented censorship incidents. We expect the dark number to be higher. All these blocks greatly differ in their sophistication and range from simple IP address blacklisting to a sophisticated hybrid consisting of deep packet inspection (DPI) and active probing [3].

Censorship evasion can be seen as a *feedback loop*: circumvention tool developers learn from blocks and adapt their tools whereas censors also learn from circumvention tools and refine their blocking techniques. This feedback loop is, however, *highly asymmetric*, meaning that circumvention tool developers learn significantly less than censors. For example, the Tor project is traditionally very open in its circumvention work [2]. Code, designs, and data are available and discussed in public. Censorship systems such as the Great Firewall of China (GFW), on the other hand, are a black box. Typically, these black boxes are repeatedly queried by developers and researchers in order to unravel their inner workings. The purpose of the censorship analyser discussed in this paper is to *reduce this information asymmetry*. That way, we strive to ease censorship analysis and boost circumvention tool development.

The task of censorship analysis requires a way to observe the respective censorship system. Unfortunately, analysis is not always possible from outside the censoring networks. As a result, two popular analysis strategies are to either *1)* obtain network traffic traces for manual inspection and/or to *2)* gain access to machines inside the censoring regime. Both of these approaches can turn out to be difficult in practice. Network traces require the cooperation of users in the censoring country and are difficult to anonymise which poses a problem of operational security. Further, remote access to machines is problematic due to the lack of volunteers or appropriate proxies such as PlanetLab, open SOCKS proxies or VPSes for rental.

The above should highlight that there is a strong need for a lightweight and easy to use tool which can assist in the process of analysing blocking incidents. In a nutshell, this tool should be run by volunteering users within the censoring country and conduct a number of networking tests in order to gain an understanding of how and if Tor is blocked in a given country or network. The analysis report should then reach the Tor developers in a safe way and thus facilitate circumvention and documentation [2]. The main contribution of this paper is the design and software architecture for such a lightweight censorship analyser for the Tor anonymity network.

The remainder of this paper is structured as follows. We discuss related work in §2. The technical requirements for our analyser are presented in §3. We proceed with the software architecture in §4 and review our concept in §5. We finally conclude this paper in §6.

2 Related Work

Lots of measurement studies have been conducted in the past in order to create "one-time snapshots" of censorship. This involves understanding the GFW [3, 5, 6, 7, 8], the backbone infrastructure of Iran [9], large-scale Internet shutdowns [10] or censorship as it is conducted world-wide [11]. These studies do not—or only to a limited degree—consider censorship as it evolves over time. This is not surprising since the censor could systematically narrow down the measurement devices and tamper with the results.

The time component was first captured by Crandall *et al.* in 2007 [12]. The authors proposed a censorship monitor called ConceptDoppler which is able to determine which keywords are filtered over time. The authors further made use of latent semantic analysis to infer keywords to probe.

In 2011, Sfakianakis, Athanasopoulos and Ioannidis proposed CensMon, a distributed web censorship monitor [13]. CensMon requires PlanetLab access in order to function. A central server receives URLs as input and instructs several distributed agents to probe these URLs in order to detect censorship.

Most recently, Filastò and Appelbaum proposed the open observatory of network interference (OONI) in 2012 [14]. OONI, which is developed by the Tor project, has broader goals than ConceptDoppler or CensMon: in addition to censorship, it is also meant to detect and document surveillance and network interference. OONI is still under heavy development but was already used to expose several censorship incidents[1].

The OpenNet Initiative [15]—a collaboration between Harvard University, the University of Toronto and the SecDev group—maintains censorship-related country profiles. In contrast to OONI, the OpenNet Initiative only publishes its gathered reports; their methods as well as the developed tools are not publicly available.

Finally, it is important to note that Internet censorship and related data sets are also of interest outside computer science. Research in the social sciences, for example, strives to understand Internet censorship on a policy level. Unfortunately, poorly documented, raw and unprocessed data sets can be a major hurdle to social science researchers as pointed out by Asghari *et al.* [16]. In particular, the authors discussed their difficulties in working with the Glasnost data provided by M-Lab[2] [17]. Based on these difficulties, Asghari *et al.* presented several suggestions for data set creators which would, when addressed, facilitate working with such data sets. We strive to consider their suggestions in the following sections.

Our approach differs from the work discussed above in that *1)* our tool is designed specifically for Tor and *2)* intended to be run by users rather than developers. This user-centric design comes with several challenges we aim to address in the following sections. By including users, we aim to be able to "shine light in dark places" and expose censorship which cannot be measured from outside the respective network.

3 Requirements and Design

At first glance, one would expect a censorship analyser to gather as much data and conduct as many experiments as possible. In practice however, this can be a bad idea since the analyser will be run by non-technical computer users who might even face repercussions for running such a "noisy" tool. Besides, users should not be linkable to the data, their copy of our tool generated. As a result, it is important to find a balance between *meaningful data* and respecting the *privacy and security* of users volunteering to run the analyser.

In §3.1 and §3.2, we will enumerate a variety of features we consider desirable in a Tor censorship analyser. These features are motivated by prior experience in debugging censorship incidents as well as by proactively probing for what censors might implement next. Naturally, some features are harder to implement than others. Therefore, the list is organised in ascending order based on the difficulty of the respective feature. While these features were designed specifically for Tor, some might be interesting for other circumvention tools as well.

[1]URL: https://ooni.torproject.org/archives.html.
[2]URL: http://www.measurementlab.net.

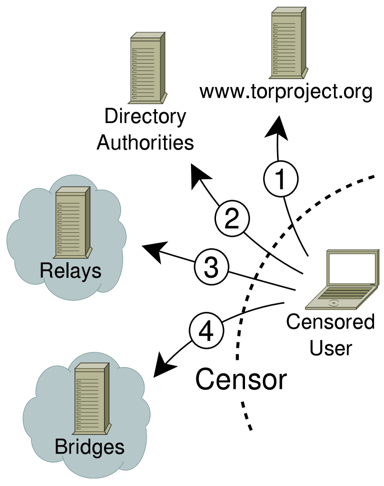

Figure 2: The Tor censorship analyser *1)* probes the official website, *2)* tries to download the consensus, *3)* tries to connect to relays *4)* and to bridges.

3.1 Analysis-centric Requirements

3.1.1 Network Trace of Analysis

Optionally, the analyser should be able to capture a network trace in pcap format. A network trace enables minute inspection of censorship techniques such as spoofed TCP reset segments injected into the user's connection. The trace must be limited to the network traffic generated by our censorship analyser and must not contain network traffic other than that. While clearly useful, pcaps contain IP addresses and can pose a threat to users having strong threat models. Techniques such as Crypto-PAn [18] could be used to pseudonymise IP addresses but packet payload could still contain identifying information. As a result, this feature should be turned off by default. If a user decides to activate it, she should be warned that she gives up a large part of her anonymity by doing so.

3.1.2 Difficult to Detect

An effort should be made to avoid obvious network activity which would make it straightforward for censors to isolate users running our analyser. Obfuscation strategies should involve random sleep periods between and during network tests and randomising the order of executed tests. We also note, however, that a fully undetectable analyser would be very difficult to implement; it would include emulation of typical user behaviour as well as steganography to transmit the final report without censors noticing.

3.1.3 Probe the Website

The official website, www.torproject.org, is sometimes found to be blocked [2]. With the website being unreachable, users should be prevented from downloading the Tor Browser Bundle (TBB) [19]. To detect website blocks, the analyser should try to download the index page by emulating an HTTP GET request of a modern browser such as Firefox or Chrome. We note that it is important to craft the GET requests to resemble one of these browsers. Otherwise, the GET requests could fail if censors whitelist user agents. If the website is believed to be blocked—because the index page could not be fetched or the TLS connection failed—the following subsequent analysis steps are performed.

First, the domain `www.torproject.org` must be resolved by querying the user's configured DNS server. The returned IP addresses (if any) are then compared to the expected IP addresses which are hard-coded in our analyser. That way, we can detect DNS poisoning. Other open and "well-behaved" DNS servers such as Google's 8.8.8.8 could further be queried. That way, we hope to be able to detect transparent DNS rewriting assuming that we trust 8.8.8.8 to return the right DNS records.

Second, another website request should be issued for one of the official Tor mirrors. In particular, for mirrors without the strings "tor" or "torproject" in their domain name. While the official website might be blocked, mirrors could very well be reachable.

Third and finally, in a subsequent step the analyser should verify whether a TCP connection to port 443 of `www.torproject.org` is possible. Besides, a TLS connection with a modified Server Name Identifier (SNI) could unravel censorship based on SNI inspection in the TLS client hello.

The same method should be applied for bridges.torproject.org which is used to distribute so-called bridges (cf. §3.1.6) to users who find themselves unable to connect to the public Tor network.

3.1.4 Probe the Directory Authorities

As of March 2013, nine directory authorities serve the consensus for the Tor network. The consensus contains the approximately 3,000 relays which form the network. These authorities are a natural *choke point*. If a client is unable to contact any of them, a direct connection to the public Tor network can not be bootstrapped. Therefore, the authorities are sometimes blocked on the IP layer [3]. To detect such censorship attempts, our analyser should be able to connect to the authorities and try to download the consensus. Should the analyser be unable to contact any of the hard-coded authorities, two subsequent checks should be performed.

First, ICMP echo requests should be sent to the authorities. Most of the authorities reply to ping requests. If no reply is received, this could be an indicator of blocking. Note that this test could easily yield false positives if the user's network disallows ICMP in general.

Second, traceroutes should be run to the potentially blocked authorities. In case of filtering, traceroutes could help narrow down the hops on the network path responsible for the block. Further, traceroutes can be conducted using a variety of protocols such as TCP, UDP and ICMP to increase the likelihood of this test to succeed.

3.1.5 Relay Reachability

After a client successfully downloaded the consensus, the next step in bootstrapping a Tor connection consists of selecting relays which together form a circuit. The first relay in a circuit—the so-called entry guard—must be reachable for the client.

Another straightforward way to block access to the Tor network lies in periodically downloading the consensus and blacklisting all IP addresses found within. In fact, it is sufficient to blacklist only the entry guards. We can easily detect this censorship strategy by trying to initiate a TLS connection to several entry guards. However, it would also be interesting to learn whether connections to pure middle or exit relays succeed. If not, then this could be an indicator that a censor is blindly blacklisting all IP addresses found in the consensus.

Connections to entry guards can fail for a number of different reasons. Perhaps the most popular strategy among censors is to terminate or drop TLS handshakes which appear to be Tor-specific. Over time, Tor's TLS handshake exhibited a number of distinguishing elements such as the client cipher list [3, 20], the server certificates as well as the randomly generated SNIs. TLS-based filtering was or is done in Ethiopia [21], China [3] and Iran [22], just to name a few. To detect Tor-specific TLS filtering, a Tor-specific TLS client hello could be sent to machines which are not Tor relays; for example the host behind `mail.google.com`. Inferring the exact pattern matched by DPI boxes is a difficult task and was sometimes done manually [20]. An automated way would be highly desirable but is beyond the scope of this paper.

3.1.6 Bridge Reachability

Analogous to ordinary Tor relays, *bridges* should be tested as well. Bridges are relays which are not listed in the public Tor consensus. Instead, they are distributed to censored users out-of-band and serve as "hidden" stepping stones into the Tor network.

Furthermore, using *obfsproxy* it is possible to obfuscate the network traffic exchanged between clients and bridges [23]. Obfsproxy is merely an obfuscation framework enabling so-called pluggable transport modules which dictate how traffic is to be obfuscated. At the time of writing, Tor can be used with the pluggable transports obfs2 [24] and obfs3 [25]. More transports are under development [26]. In order to detect DPI targeting these pluggable transports, the analyser should be able to conduct obfs2 and obfs3 handshakes and see if they succeed.

3.1.7 Gather Debug Information

Censorship is typically not homogeneous across a country and can differ on the level of provinces, autonomous systems or even network prefixes [27]. As a result, we are interested in information which can help narrow down the respective censorship infrastructure. Also, this would help ruling out interferences and prevent jumping to wrong conclusions. Of interest would be the following information.

1. What ISP does the user have? Our analyser could obtain the whois record for the user's IP address and discard all IP address material which would otherwise reveal the user's location.
2. What is the autonomous system number? Open IP-to-ASN lookup services[3] could be used.
3. Is the user behind a captive portal? This can be difficult to verify and might require a number of heuristics.
4. Is all traffic forced to go through an HTTP proxy?

3.1.8 Anonymising Reports

Naturally, reports should not be linkable to users submitting them. We have to distinguish between *report content* and *report submission*.

The actual report is straightforward to anonymise. As an option, we can easily discard identifying information in the report by, e.g., never logging the user's IP address and redacting the first hops in traceroutes. A high degree of anonymity could be achieved by completely discarding all IP addresses, prefixes and location information such as autonomous system numbers and whois records. Network captures in the form of pcap files are very difficult to anonymise and should also be completely discarded if strong anonymity is desired.

Problematic however, is report submission. It is the task of tools such as Tor to provide anonymity on the address layer. Unfortunately, we cannot make use of Tor

[3]URL: http://asn.cymru.com

as the very purpose of our tool is to find out why Tor is unreachable. Users could be advised to generate one-time email addresses for submission over email. That way however, the email provider would learn the user's IP address[4]. This would place a lot of control into the hands of an untrusted third party so another—possibly better—option is automatic submission using an HTTPS POST request to infrastructure run by the Tor project. This method could be easier to block and once again, this cannot disguise the user's IP address, unfortunately.

3.2 User-centric Requirements

While the analyser's main purpose is to gather censorship-related data, it is important to recognise that it will frequently—if not mostly—be run by users who *lack technical expertise*. These users will not be able to configure the analyser manually or run tools on the command line. This observation motivates the following requirements.

3.2.1 User-friendly Output

We emphasise that the primary purpose of the analyser is to gather data which can assist Tor developers. Still, as a side task, the analyser should inform users about the gathered data and results while avoiding too much technical jargon. Based on a decision tree inferred from the gathered data, the analyser could inform the user about possible ways to circumvent the respective censoring network.

3.2.2 Cover our Tracks

As mentioned above, it is crucial to protect users' privacy. This implies that temporary files or reports must not be stored in non-obvious locations on the user's hard drive. All temporary data must remain in the unpacked directory containing the analyser. That way, a user can easily delete the entire directory and by doing so also delete gathered data.

Nevertheless, it is difficult to hide the fact that a certain program was executed [28]. Even more so on Windows. What can be hidden is the analysis results by simply deleting them after submission[5]. The very existence of our analyser is difficult to disguise, however.

We argue that *usage diversity* can mitigate the threat arising from simply having a copy of our tool. Users could state that they were simply interested in finding out why their TBB would not bootstrap (cf. §3.2.1) rather than in assisting in circumventing their national firewall.

[4]The content of the report can be protected when the report is encrypted using the hard-coded PGP public key (see §4.1).

[5]Note that simple file removal might not be sufficient when file system forensics is conducted.

3.2.3 Ease of Use and Informed Consent

It is crucial that the analyser is as easy to use as possible. In particular, it should be a self-contained click-and-go executable, just like the TBB. Ease of use also involves the analyser's *bundle size*. Ideally, the analyser would only be a few megabytes in size which would also make is suitable for email distribution over GetTor[6].

Users should be informed in clear words that the analyser is performing network probing and that the gathered data can be submitted afterwards for further inspection. Users should be given the opportunity to abort the process prior to network probing and prior to submission. Note that internationalisation of our analyser will also be necessary since it is unrealistic to expect users to have a decent command of the English language.

4 Software Architecture

There is no need to reinvent the wheel. The analyser will be implemented as a set of tests for OONI [14]. OONI, which was already introduced in §2, is a framework for network censorship analysis and provides a Python API which can be used to develop all of the requirements discussed above. In addition, OONI defines a custom YAML format—YAMLOO—for analysis reports. At the time of writing OONI cannot, however, be invoked by running a single executable. As a result, our analyser must be packaged together with OONI to a single Windows executable. This can be done using tools such as py2exe[7]. The following sections discuss a number of architectural considerations.

4.1 Communicating Results

Eventually, the data produced by a user's analyser—mainly a text file containing YAMLOO data—has to reach the Tor developers. We believe that it would be short-sighted to define a single mechanism for the transmission to happen. This would create a single point of failure which might turn out to be easy to block for censors.

Instead, we envision several reasonable communication channels. Ideally, we would send the report in an anonymous fashion and upload it to a hidden service. But as already discussed in §3.1.8, we expect Tor to be censored and not an option for this. A report could be sent manually over email. While this requires user interaction, some sort of email is typically available; even in countries with pervasive censorship. In particular, free

[6]URL: https://www.torproject.org/projects/gettor.html.en

[7]URL: http://www.py2exe.org.

services such as GMail[8] are frequently reachable over HTTPS.

Another option would be to automatically transmit the report using an HTTPS POST request to a web server operated by the Tor project. While single web servers are straightforward to block, we could increase collateral damage by running the web server inside a cloud provider, the censor is hopefully unwilling to blacklist. Other communication channels could include instant messaging programs or steganographic publishing systems. Unfortunately, all of these methods have in common that the user's IP address will eventually be known by a provider of some sort.

Furthermore, the report should contain a *message digest* which is calculated over the YAMLOO report. This is particularly important when the report is being sent over email since it makes it possible to detect if the report is incomplete or the user accidentally changed parts of it.

Finally, the analyser should contain a hard-coded PGP public key which can be used to encrypt analysis reports. Users who decide to encrypt the report should still have the opportunity to review a report prior to encryption and submission. While encryption comes at the cost of key management, we believe that the additional management overhead is worth the increase in security in certain situations.

4.1.1 Configurable and Testable During Build

It should be possible to pass configuration parameters to the analyser during the build process. That is necessary because certain information will be subject to change. This includes hard-coded IP addresses of relays or the web servers hosting www.torproject.org. If we would be unable to change these parameters, a censor could simply whitelist the hard-coded IP addresses and render our analysis results useless.

All the features discussed in §3 should be testable in an automated way. Otherwise, we might end up shipping code which does not work in real environments or we might not notice if improvements break existing code.

5 Discussion

We point out that motivated and strongly equipped censors would be able to identify users trying to run our analyser. Such censors would also be able to actively falsify our analyser's results. Being undetectable would require a substantially more complex concept. However, we believe that most censors are more motivated to spend their resources on actual blocking technology rather than measurement and analysis technology. Of course, this will change if our analyser should turn out to be a very valuable tool in the arms race.

There are two additional important features not discussed here due to page constraints. First, our analyser could be enhanced to be able to infer the patterns used by DPI boxes to identify protocols. According to our own experience, DPI boxes do not always just use simple regular expressions but also context-sensitive languages. This is, however, a difficult problem which might require a client-server architecture which runs a grammatical inference algorithm. Precise knowledge of the patterns used for censorship would enable targeted circumvention and could be fed into tools such as format-transforming encryption [29].

Second and equally difficult, our analyser could have features to detect the type or model of DPI box used for censorship. This would make it possible to expose cases similar to the use of Bluecoat in Syria [30]. While well-designed DPI boxes not exposing their management interface can be difficult to identify, it might be possible to define a number of heuristics to cluster DPI boxes; perhaps based on injected TCP reset segments (cf. [31]).

6 Conclusion

In this paper, we outlined the design requirements for a censorship analyser for the Tor anonymity network. While the majority of these requirements discuss network probing techniques, we also consider usability aspects. We plan to implement our concept in the next step.

We believe that involving ordinary computer users in the process of censorship analysis can greatly increase the coverage of censorship documentation and shed light on previously unknown types of censorship. Great care must be taken, however, to not exploit users and inform them about the process. Informed consent must be achieved whenever possible. Furthermore, we hope to start a discussion about how to safely integrate non-technical users into the process of censorship measurement.

Acknowledgements

We want to thank Arturo Filastò, Simone Fischer-Hübner, George Kadianakis, Karsten Loesing, Tobias Pulls, Runa A. Sandvik and the anonymous reviewers for their valuable feedback.

Finally, we want to express our gratitude to Internetfonden of the Swedish Internet Infrastructure Foundation for supporting the author's work with a research grant.

[8]URL: https://mail.google.com.

References

[1] Roger Dingledine, Nick Mathewson, and Paul Syverson. “Tor: The Second-Generation Onion Router”. In: *USENIX Security*. USENIX Association, 2004, pp. 303–320. URL: http://static.usenix.org/event/sec04/tech/full_papers/dingledine/dingledine.pdf.

[2] The Tor Project. *Censorship Wiki*. URL: https://censorshipwiki.torproject.org.

[3] Philipp Winter and Stefan Lindskog. “How the Great Firewall of China is Blocking Tor”. In: *FOCI*. USENIX Association, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final2.pdf.

[4] The Tor Project. *Directly connecting users from Iran*. URL: https://metrics.torproject.org/users.html?graph=direct-users&start=2013-01-01&end=2013-03-15&country=ir&events=off#direct-users.

[5] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. “Ignoring the Great Firewall of China”. In: *PETS*. Springer, 2006, pp. 20–35. URL: http://www.cl.cam.ac.uk/~rnc1/ignoring.pdf.

[6] Sparks et al. “The Collateral Damage of Internet Censorship by DNS Injection”. In: *SIGCOMM Computer Communication Review* 42.3 (), pp. 21–27. URL: http://conferences.sigcomm.org/sigcomm/2012/paper/ccr-paper266.pdf.

[7] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. “Internet Censorship in China: Where Does the Filtering Occur?” In: *PAM*. Springer, 2011, pp. 133–142. URL: http://www.eecs.umich.edu/~zmao/Papers/china-censorship-pam11.pdf.

[8] Jong Chun Park and Jedidiah R. Crandall. “Empirical Study of a National-Scale Distributed Intrusion Detection System”. In: *ICDCS*. IEEE, 2010, pp. 315–326. URL: http://www.cs.unm.edu/~crandall/icdcs2010.pdf.

[9] Collin Anderson. *The Hidden Internet of Iran: Private Address Allocations on a National Network*. Tech. rep. 2012. URL: http://arxiv.org/pdf/1209.6398v1.

[10] Alberto Dainotti et al. “Analysis of Country-wide Internet Outages Caused by Censorship”. In: *IMC*. ACM, 2011, pp. 1–18. URL: http://conferences.sigcomm.org/imc/2011/docs/p1.pdf.

[11] John-Paul Verkamp and Minaxi Gupta. “Inferring Mechanics of Web Censorship Around the World”. In: *FOCI*. USENIX Association, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final1.pdf.

[12] Jedidiah R. Crandall et al. “ConceptDoppler: A Weather Tracker for Internet Censorship”. In: *CCS*. ACM, 2007, pp. 352–365. URL: http://www.cs.unm.edu/~crandall/concept_doppler_ccs07.pdf.

[13] Andreas Sfakianakis, Elias Athanasopoulos, and Sotiris Ioannidis. “CensMon: A Web Censorship Monitor”. In: *FOCI*. USENIX Association, 2011. URL: http://static.usenix.org/event/foci11/tech/final_files/Sfakianakis.pdf.

[14] Arturo Filastò and Jacob Appelbaum. “OONI: Open Observatory of Network Interference”. In: *FOCI*. USENIX Association, 2012. URL: https://www.usenix.org/system/files/conference/foci12/foci12-final12.pdf.

[15] OpenNet Initiative. URL: https://opennet.net.

[16] Hadi Asghari et al. *Making Internet Measurements Accessible for Multi-Disciplinary Research*. Tech. rep. TU Delft and Syracuse University, 2012. URL: http://dpi.ischool.syr.edu/MLab-Data_files/HA-MM-MvE-IMC.pdf.

[17] Marcel Dischinger et al. “Glasnost: Enabling End Users to Detect Traffic Differentiation”. In: *NSDI*. USENIX Association, 2010. URL: http://broadband.mpi-sws.org/transparency/results/10_nsdi_glasnost.pdf.

[18] *Cryptography-based Prefix-preserving Anonymization*. URL: http://www.cc.gatech.edu/computing/Telecomm/projects/cryptopan/.

[19] The Tor Project. *Tor Browser Bundle*. URL: https://www.torproject.org/projects/torbrowser.html.en.

[20] Tim Wilde. *Great Firewall Tor Probing Circa 09 DEC 2011*. 2012. URL: https://gist.github.com/twilde/da3c7a9af01d74cd7de7.

[21] The Tor Project. *An update on the censorship in Ethiopia*. 2012. URL: https://blog.torproject.org/blog/update-censorship-ethiopia.

[22] The Tor Project. *Iran blocks Tor; Tor releases same-day fix*. 2011. URL: https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix.

[23] The Tor Project. *obfsproxy*. URL: https://www.torproject.org/projects/obfsproxy.

[24] The Tor Project. *obfs2 (The Twobfuscator)*. URL: https://gitweb.torproject.org/obfsproxy.git/blob/HEAD:/doc/obfs2/protocol-spec.txt.

[25] The Tor Project. *obfs3 (The Threebfuscator)*. URL: https://gitweb.torproject.org/user/asn/obfsproxy.git/blob/HEAD:/doc/obfs3/obfs3-protocol-spec.txt.

[26] The Tor Project. *Tor: Pluggable Transports*. URL: https://www.torproject.org/docs/pluggable-transports.html.en.

[27] Joss Wright, Tulio de Souza, and Ian Brown. "Fine-Grained Censorship Mapping: Information Sources, Legality and Ethics". In: *FOCI*. USENIX Association, 2011. URL: http://static.usenix.org/event/foci11/tech/final_files/Wright.pdf.

[28] Runa A. Sandvik. *Forensic Analysis of the Tor Browser Bundle on OS X, Linux, and Windows*. Tech. rep. The Tor Project, 2013. URL: https://research.torproject.org/techreports/tbb-forensic-analysis-2013-06-28.pdf.

[29] Kevin P. Dyer et al. *Protocol Misidentification Made Easy with Format-Transforming Encryption*. Tech. rep. Portland State University, RedJack, LLC., and University of Wisconsin, 2012. URL: http://eprint.iacr.org/2012/494.pdf.

[30] Jillian C. York. *Government Internet Surveillance Starts With Eyes Built in the West*. 2011. URL: https://www.eff.org/deeplinks/2011/09/government-internet-surveillance-starts-eyes-built.

[31] Nicholas Weaver, Robin Sommer, and Vern Paxson. "Detecting Forged TCP Reset Packets". In: *NDSS*. The Internet Society, 2009. URL: http://www.icsi.berkeley.edu/pubs/networking/ndss09-resets.pdf.