Do You See What I See?
Differential Treatment of Anonymous Users

Sheharbano Khattak*, David Fifield†, Sadia Afroz†, Mobin Javed†
Srikanth Sundaresan‡, Vern Paxson‡†, Steven J. Murdoch†, Damon McCoy‡

*University of Cambridge †University of California, Berkeley
‡International Computer Science Institute, Berkeley †University College London

Sheharbano.Khattak@cl.cam.ac.uk,{fifield,sadia.afroz,mobin.javed,vern}@berkeley.edu
s.murdoch@ucl.ac.uk,{srikanth,mccoy}@icsi.berkeley.edu

***Abstract*—The utility of anonymous communication is undermined by a growing number of websites treating users of such services in a degraded fashion. The second-class treatment of anonymous users ranges from outright rejection to limiting their access to a subset of the service's functionality or imposing hurdles such as CAPTCHA-solving. To date, the observation of such practices has relied upon anecdotal reports catalogued by frustrated anonymity users. We present a study to methodically enumerate and characterize, in the context of Tor, the treatment of anonymous users as second-class Web citizens.**

We focus on first-line blocking: at the transport layer, through reset or dropped connections; and at the application layer, through explicit blocks served from website home pages. Our study draws upon several data sources: comparisons of Internet-wide port scans from Tor exit nodes versus from control hosts; scans of the home pages of top-1,000 Alexa websites through every Tor exit; and analysis of nearly a year of historic HTTP crawls from Tor network and control hosts. We develop a methodology to distinguish censorship events from incidental failures such as those caused by packet loss or network outages, and incorporate consideration of the endemic *churn* in web-accessible services over both time and geographic diversity. We find clear evidence of Tor blocking on the Web, including 3.67% of the top-1,000 Alexa sites. Some blocks specifically target Tor, while others result from fate-sharing when abuse-based automated blockers trigger due to misbehaving Web sessions sharing the same exit node.

I. Introduction

Anonymity networks serve an important purpose on the Internet. They often provide the only means for citizens to access or distribute censored or restricted content without a threat to their privacy or even safety. A predominant example of such a network is Tor [6], the 'king of high-secure, low-latency Internet anonymity' according to the NSA [25]. The success of such networks depends on their *utility*, i.e., the degree to which they provide not only acceptable performance but also unfettered access to the global Internet.

Traditional threats to Tor involve deanonymization attacks that reduce user privacy, or governments blocking access to the Tor network. A different kind of threat, which we explore in this paper, involves websites providing Tor users with degraded service, resulting in them effectively being relegated to the role of second-class citizens on the Internet [4]. Such differential treatment ranges from websites employing wholesale blocking of Tor-related IP addresses to requiring Tor clients to complete CAPTCHAs before continuing. The problem becomes amplified when 'bottleneck' web services (e.g., CloudFlare, Akamai) whose components are used by many other websites block or discriminate against Tor users, or when third-party blacklists used by a large number of websites include Tor infrastructure (in particular, exit node) IP addresses.

Addressing the Web's second-class treatment of Tor users begins with enumerating and characterizing its prevalence. Thus far, efforts to do so have been ad hoc; effectively by cataloging reports from frustrated users about services that routinely employ such practices [27]. In this work, we carry out a broad, systematic enumeration and characterization of websites and IP addresses that treat Tor users differently from normal connections. We run two complementary measurement campaigns. *1*) At the network layer, we scan the entire IPv4 address space (with a small exclusion list) using a modified version of ZMap [8]. We run the scans from a select number of active Tor exit nodes as well as non-Tor control nodes. *2*) At the application layer, we probe the top 1,000 Alexa Web sites using Exitmap [29]. We fetch home pages of these sites using Tor and non-Tor nodes and analyze the responses to uncover evidence of Tor blocking.

We demonstrate the existence of differential treatment of Tor users at both the network and application layers. At the network layer, we estimate that at least 1.3 million IP addresses that would otherwise allow a TCP handshake on port 80 block the handshake if it originates from a Tor exit node. We also show that at least 3.67% of the top 1,000 Alexa web sites block Tor users at the application layer.

We explore the reasons and techniques used by these websites, and how much of this differential treatment is due to explicit decisions to block Tor versus the consequence of fate-sharing due to automated abuse-based blocking. We identify two kinds of network-layer blocking: wholesale blocking by Autonomous Systems (ASes) such as access ISPs, and more targeted (likely abuse-driven, and thus implicit) blocking practiced by content hosting sites and service providers.

NDSS '16, 21-24 February 2016, San Diego, CA, USA
Copyright 2016 Internet Society, ISBN 1-891562-41-X
http://dx.doi.org/10.14722/ndss.2016.23342

While wholesale blocking of Tor as a matter of ISP or national policy will likely always exist, our results highlight a growing concern that anonymity networks are being negatively affected beyond explicit blocking, due to the desire of web services to block abuse. In this work we contribute a systematic methodology and measurement study of the scale of blocking of anonymity networks, both at the network layer (Section IV) and the application layer (Section V). Our work provides a first step towards addressing this problem by methodically uncovering and characterizing the nature of blocking of anonymity networks as seen at scale.

II. BACKGROUND

For our anonymity system case study, we analyse Tor [7], the most widely used anonymous communication system, with over 2 million daily users [22]. Tor was designed to allow users to access TCP-based services (predominantly websites) privately and securely, preventing any intermediate agent from linking the user's identity to their activities. However, many Tor users primarily seek to circumvent censorship rather than to obtain privacy. Blocking access from Tor imposes serious limitations for Tor users, and has significant implications for Tor itself, potentially reducing its utility substantially. We provide a brief background on Tor's design and the different ways it is blocked.

A. Tor

Tor works by routing users' traffic over a three-hop 'circuit', with each hop being a volunteer-operated 'node' running the Tor software in server mode. Tor uses both per-link and end-to-end cryptography to provide confidentiality, integrity, and unlinkability between incoming and outgoing traffic at each hop. Tor users typically install the Tor Browser Bundle, which consists of a hardened Firefox-based browser and the Tor software configured as a client. When a user makes a request, the Tor client selects three nodes out of those available to form a circuit, connecting first to the 'entry guard', then through it to the 'middle node' and finally to the 'exit node'.

The exit node makes the TCP connection to the desired service and so is also the first target for abuse complaints from operators. For this reason, not everyone is willing to operate an exit node, and the Tor server configuration allows operators to set an 'exit policy' stating to which IP addresses and ports the node will carry exit traffic. When a node activates (and periodically afterwards), it publishes a 'descriptor' to each of the 'directory authorities' which includes the IP address and port at which circuits can connect to the node, its exit policy, and its public key. The directory authorities together form and digitally sign the 'directory consensus', which they make available to clients both directly and via Tor nodes that act as 'directory mirrors'.

The directory consensus includes the information from each node's descriptor, but also includes a set of flags indicating in which positions a node can serve in the circuit (only sufficiently fast and stable nodes can serve as entry guards, and only nodes with a sufficiently permissive exit policy as exit nodes). Furthermore, the consensus includes a 'consensus weight' for each node, which is an integer proportional to the node's bandwidth capacity as measured by a set of 'bandwidth authorities'. When selecting a node for each position in the circuit, clients first identify all the nodes that can take the respective position, and then select from these randomly, but biased by the consensus weights such that in aggregate they place a network load on Tor nodes in proportion to their capacity [23].

Due to its ability to circumvent censorship, the Tor network itself is subject to censorship. The simplest form consists of blocking access to the entry nodes by their IP addresses (which are easily found from the directory consensus). To counter this threat, Tor maintains a set of Tor nodes ('bridges') that act as entry points to the network but are not publicly listed in the consensus. Bridges are instead distributed to individuals in censored countries, making them harder to block reliably [6].

In reaction to this move, some countries fingerprint Tor traffic to block it, so Tor now allows the integration of 'pluggable transports' [24] that disguise the characteristics of Tor traffic. The use of bridges and/or pluggable transports does not affect how traffic exits the Tor network, so for the purposes of our study we do not deal with them specially.

B. Blocking/Filtering Tor

It is technically easy for Internet sites to block traffic from Tor relays on a wholesale basis, as there exist readily accessible and regularly updated lists of Tor relays. Internet services may have different reasons to apply such blocking: to discourage contributions by anonymous users, or avoid abuse such as comment spam. Inevitably, some well-meaning users will be excluded due to how widely Tor shares exit nodes across many users.

The first step to construct a Tor-specific blacklist is to collect the IP addresses of exit nodes. The easiest approach is to collect the IP addresses from the node descriptors in the directory consensus. However, these addresses denote the incoming IP address for nodes, and for nodes with multiple IP addresses this will not necessarily be the IP address for outgoing connections. As a result, using the IP addresses from the consensus could lead to both overblocking (by blocking the incoming IP address even though it is never used for outgoing exit traffic, but may have other uses) and underblocking (by failing to block the outgoing IP address because it is not an incoming address for any node). A more robust approach is 'active probing' by making Tor circuits that use each exit node in turn to establish a connection to a test server, and observing the originating IP address. This approach increases the accuracy of the list but puts more load on the network and reduces the frequency at which the list can be easily updated.

The second decision is which nodes to consider to be exits. The easiest option is to use the 'exit' flag assigned by the directory authorities if the node's exit policy permits at least two ports from 80 (HTTP), 443 (HTTPS) and 6667 (IRC) to at least one /8 IP address range [23]. Relying on the exit flag results in overblocking because it is possible that an exit node will never be selected for a connection to a particular service using the blacklist even if it has the exit flag set (perhaps the service's IP address and/or port is excluded by the node's exit policy). Therefore non-Tor users of the computer hosting the exit node will be blocked from accessing the service even though there is no possibility that this computer will be the origin of Tor-originated abuse. There may also be underblocking if the node does not meet the criteria for the exit flag but its exit policy still permits connecting to the service in question.

Finally, the blacklist operator may decide to include some non-exits in the list (e.g., including nodes that have a 'deny all' exit policy and so can only be entry guards or middle nodes, or including IP addresses on the same netblock as Tor nodes). This approach is especially pernicious, as it leads to blocks of bystander IP addresses in ways that have little to do with Tor-sourced abuse. Motivations for doing so may include a desire to deter people from running Tor servers, or to mitigate underblocking that may occur as a result of missing Tor server configuration changes or mismatches between incoming and outgoing IP addresses for the node.

Examples of publicly available Tor blacklists include `dan.me.uk` [1], which optionally includes non-exit Tor nodes, and `dnsbl.sectoor.de` [19], which includes all IP addresses on the same /24 as the Tor exit by default. The Tor project itself maintains TorDNSEL [21], which uses active probing to increase accuracy, and also takes into account the specific service using the blacklist so as to reduce overblocking and underblocking.

To avoid complications resulting from these different approaches to blacklisting, we run our control probes from systems that did not share a /24 IP address with any Tor node, and our Tor-based probes from exit nodes that had the exit flag for at least a month, as well as permitting access to almost all IP addresses on port 80 (the destination port for our probes).

III. Related Work

We consider Internet censorship relevant to Tor from three perspectives: direct censorship of content, censorship of traffic *entering* Tor, and censorship of traffic *exiting* Tor. A large and growing body of literature focuses on the first two classes, but the latter category has seen little in the way of study; our work aims to fill this gap.

Much existing work has focused on measuring and evading direct content blocking in different countries. Prior work has also focused on government blocking of censorship circumvention systems. Dingledine et al. [5] discuss when and how different governments tried to block access to Tor; governments mainly use address-based blocking of requests to the Tor website, relays, and bridges, and protocol-based blocking of TLS connections to the Tor network identified by Tor specific characteristics (for example, cipher suite).

Our work focuses on a different aspect of the censorship problem: we examine *server-side* blocking of clients; that is, blocking by the server based on the characteristics of the source, not blocking by an intermediate firewall based on characteristics of the destination. In the classical Internet censorship scenario, the server would be happy to accept connections from a client, but some network device near the client prohibits it. We, on the other hand, look at cases where the client's connection arrives at the server unimpeded, but the server (or something working on its behalf) rejects it.

In our work we make use of data from the Open Observatory of Network Interference (OONI) [17]. (Despite a similarity in purpose and acronym, this project is separate from the OpenNet Initiative discussed above.) The OONI dataset has a crucial feature for studying differential treatment of Tor users: it consists of many simultaneous downloads both with Tor and without Tor. While the intent behind these measurements is to highlight content that is inaccessible from certain locations *unless* one uses Tor, we can employ the same information to identify destinations inaccessible *because* one uses Tor.

Developing robust techniques to detect blocking is also important. We need to know when an application is being blocked, and we also need to distinguish genuine network interference from benign or transient failures. Jones et al. tested automated means of detecting censorship block pages in an OpenNet corpus [13]. A metric based on page length proved the best-performing of several options. Our experiments necessitate different ways of detecting blocks at different network layers. In Section IV we use repeated scans across space and time, and in Section V we compare test downloads against simultaneous control downloads.

IV. Measuring Network-Layer Discrimination

As we discuss in Section II, a straightforward technique for services to block Tor is to filter traffic from publicly listed exit nodes. To broadly assess this, we measure Tor filtering using ZMap probing from both Tor exit nodes and from control (non-Tor) nodes to see how their access to remote addresses differs. For convenience we term these measurements as assessing 'network-layer' discrimination, though from a technical perspective they combine measurement of layer-3 and layer-4 blocking, since we restrict our measurements to attempts to connect to TCP port 80 services.

A. ZMap

ZMap is a high-performance network scanner capable of scanning the entire IPv4 address space in as little as 45 minutes, much faster than traditional scanners such as Nmap [8]. ZMap achieves this efficiency by incorporating multiple optimizations, including randomized target selection and maintaining no connection state. Because ZMap does not maintain state, it also does not retransmit probes in case of loss. We used ZMap for test runs of the entire IPv4 address space starting in Spring 2015. Over the course of repeated experiments, we uncovered several bugs (for some of which we contributed fixes, while others were fixed by the ZMap team), addressed measurement considerations (for avoiding measurement loss), and added extra functionality as discussed below. For our measurements we recorded both TCP SYN-ACKs and RSTs. We configured ZMap to run at 100 Mbps rather than at 1 Gbps to avoid saturating our local networks. Doing so results in one scan taking about 7 hours rather than 45 minutes.

B. Overview of measurements and block detection

We run our scans from Tor exit nodes and from two sets of control nodes: university nodes and a Tor middle node. We compare responses to our Tor scans with those from the baseline control scans and flag deviations as potentially reflecting discriminatory blocking. Target hosts respond to ZMap probes

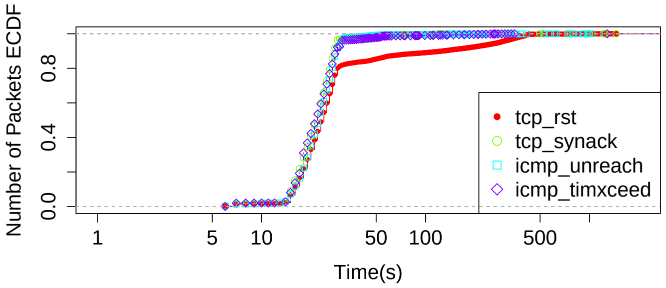


Fig. 1: Distribution of time until receiving a response packet since the last probe was sent for a full scan of IPv4.

(TCP SYNs) in one of three ways: *a*) sending a SYN-ACK, which we term a *successful* response; *b*) sending a RST, which we term an *unsuccessful* response; or *c*) not responding, which we also deem an unsuccessful response. ZMap, by default, only records successful responses; we modified it to record RSTs as well. We note that for an individual probe it is not possible to distinguish a lack of response from packet loss.

We might in simple terms think we can identify Tor blocking by observing destination addresses that respond to probes from our control nodes but not those from our Tor exit nodes. However, this reasoning has two main limitations: 1) Unsuccessful responses could arise due to *packet loss* along either the packet forward or return path; and 2) Destinations can respond inconsistently to probes due to factors unrelated to discriminatory blocking, such as servers only operating during certain hours of the day or days of the week.

More generally, we need to consider issues of *churn*: how Internet service reachability varies, in both spatial and temporal terms. By *spatial churn* we mean the notion that simultaneous probes sent from topologically separate clients to the same server might yield different outcomes, for example due to network congestion or a network outage blocking the path from one of the clients but not the other. By *temporal churn* we refer to the reachability from the same client to the same server varying over the course of time, for example due to day-of-the-week effects governing when the server is accessible.

Thus, to understand how to soundly compare probe outcomes seen at our control nodes versus from Tor nodes, we need to incorporate consideration of how to distinguish probing results that differ due to churn versus those that actually reflect discrimination. Note that through the rest of our discussion, the underlying assumption is that services either completely block a Tor exit node or allow it. We do not deal with selective blocking or rate-limiting in this paper.

C. Mitigating the Impact of Packet Loss

As noted above, ZMap does not allow us to distinguish between a single non-response and a packet loss event. To account for this limitation, we take care to minimize measurement loss in our measurements and to account for potential packet loss in the network.

1) Mitigating Measurement Loss: We first ask whether ZMap accurately sends all the packets it is configured to send, and whether it correctly logs packets and responses.

We profiled ZMap using an experimental setup that consists of a well-provisioned machine running ZMap, and a separate machine running a packet capture. All the ZMap packets are directed to the second machine via a Gigabit Ethernet cable. Separating packet transmission and packet capture allows us to account for losses occurring due to both ZMap itself and the underlying network card. It also avoids the scenario where the two processes compete with each other for CPU cycles. When ZMap runs with its default configuration, we see a 6.7% failure rate—this failure is completely eliminated when we throttle our sending rate down from 1 Gbps to 100 Mbps. (During this process, we also identified and reported a bug in ZMap that caused it to not send certain packets due to the interaction between scan targets, the blacklist and thread-level sharding.)

In addition, we need to configure a timeout for ZMap to deem that a packet did not receive a response. Figure 1 shows the distribution of the time measured between sending the last scan packet and receiving a response for a full scan of IPv4. To generate this plot, ZMap logged response packets for 25 minutes after sending the last scan packet. More than 95% of all replies (excluding RSTs), and 80% of RSTs arrive within the first 30 seconds, while the rest trickle in up until 500 seconds. Though unusual, late responses could arise due to backed-off timers in the case of SYN-ACKs, huge bufferbloat, or initial latency incurred by extensive setup requirements of cellular wireless devices [18]. Given this data, we chose a conservative cooldown value of 10 minutes for responses to come in.

2) Network Packet Loss: An unsuccessful response can be due to loss on the paths between the scanner and the destination, caused by transient network issues such as congestion or network failure. We reduce such noise by sending redundant probes per destination. If any of the probes elicits a SYN-ACK from the destination, we treat it as a successful response, because a single response suffices to inform us that the target server does not block Tor traffic.

We can introduce probe redundancy in many ways; the simplest is by conducting back-to-back scans from the same vantage point. However, since a single scan takes about 7 hours to complete, such an approach introduces a large gap between the redundant probes, which can lead to inconsistent responses due to temporal churn. We ran 3 back-to-back scans from one of our control vantage points. We observed a temporal churn between the first two scans of 13.30%, which increased to 21.61% when computed across the three scans. We repeated the experiment at another of our control vantage points and made similar observations. This finding means that servers respond quite inconsistently across large intervals of time.

This high temporal churn motivates us to incorporate redundancy at shorter timescales in our measurements. Although ZMap allows us to send multiple probes per target in a single scan, it does so back-to-back without any delay between them. This approach only helps if loss events are independent; however, transient network issues mean that loss events are presumably not independent.

Control nodes	
Number of control nodes	3
Number of IPv4 scans	7 per control node
Time span of scans	Aug 7–13
Scanned IP addresses per measurement	3,662,744,599
Average hit-rate per measurement	1.91% (σ=0.01%)
Average (estimated) network loss	0.84% (σ=0.18%)

Tor exit nodes	
Number of exit node	4
Number of scans	4 per exit node
Time span of scans	Aug 10–13
Average hit-rate per measurement	1.87% (σ=0.03%)

TABLE I: Summary of control and exit node data. For all scans, we filtered out IP addresses included in the largest blacklist, that is, one employed by the last scan. Network loss per measurement is estimated as the percentage of IP addresses inaccessible from a node but accessible from at least one other node.

Since ZMap does not keep state, we cannot retransmit only those for which we did not receive a response. We therefore follow the simple strategy of sending K probes, resending them, sending another K probes, resending them, and so on. For $K = 1,000,000$ and with a sending rate of 100 Mbps, this means that the retransmitted probe follows 6.7 sec after the original. This approach allows us to maintain the sending bandwidth and allows us to keep ZMap as a single threaded process; however, and although, as expected, it doubles the length of a full scan. Across three sites and four scans, we found that factoring in responses to retransmitted probes increases the response rate for original probes by 1.04% (we can distinguish these by sending retransmissions from different ports). We further observe that there is a temporal churn of 1.93% between 6.7 sec apart scans, which is significantly lower than 13.30% churn for scans run back-to-back (effectively ≈7 hours apart).

D. Data

We run our measurements from a set of three control nodes and a set of four Tor exit nodes. Two control nodes are located in US universities and one in a European university. The control node measurements serve a dual purpose: they allow us to calibrate and understand our measurement method and the data, and they serve as the baseline measurements against which we compare the Tor exit node measurements.

Our first goal is to develop a global ‘web footprint’, a set of IP addresses that respond to our scans on port 80. On average, a control node sees a hit rate of 1.91% (σ=0.01%) per measurement scan (translating to ≈ 70 million IP addresses). We note that each scan consists of two probes per target IP address (Section IV-C2); and a ‘hit’ consists of SYN-ACK response to our SYN for at least one probe. This number is roughly constant across the three locations. However, due to multiple reasons, including routing and transient failures, network policies, time-of-day effects, and regular usage patterns, no two scans return the same set of IP addresses (the issue of *churn* discussed previously).

We first conducted extensive preliminary ZMap scans (on the order of 90 scans over a period of 3 months) in order to calibrate the accuracy of our measurement methodology and address problems that arose. The scans all employed a blacklist excluding IP addresses, which we added to whenever we received a request. We run web servers on our control nodes that identify our scanning activity as research and provide an email address for sites to opt out of further scanning. . During our measurements from March to August 2015, we received scan exclusion requests from a total of 134 unique email addresses for 426 networks (covering a total of 3,532,751 hosts). Note that this number provides an upper bound as the machines at Michigan and Berkeley use site-wide scan notices, implying that a complaint could have been triggered by any of the scans running from these sites .

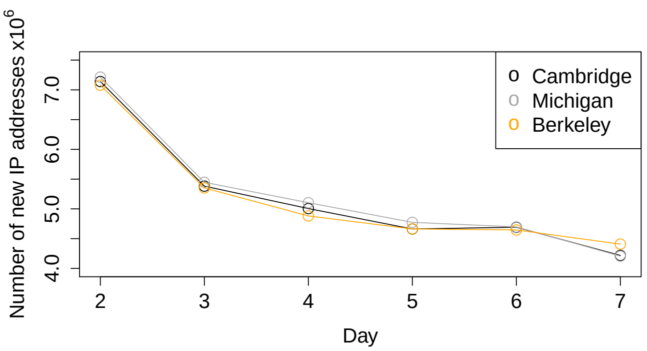


Fig. 2: Number of new IP addresses each control node sees per day.

Once fully developed and debugged, for our final analysis we gathered 37 full IPv4 scans over a period of 7 days, conducting 16 from four Tor exit nodes. Table I shows the breakdown of the measurements run from the control and Tor exit nodes. We now turn to analyzing the final data to understand temporal churn—how the footprint changes across scans spanning multiple days—and spatial churn—how our view of the global web footprint set changes across the three control locations.

a) Temporal Churn: For the same location, we see significant differences in the number of IP addresses that successfully respond, even between consecutive days, ranging up to 17%. Figure 2 shows the number of new IP addresses that each site successfully contacts per day. Using the first day as the baseline, this value somewhat gradually drops from a peak of about 7 million on the second day to about 4 million on day 7. The slow convergence rate indicates that temporal churn remains high even for the same location, and that obtaining a true underlying web footprint for a given location may not be well-defined. Temporal churn is likely caused by nodes that only come online occasionally; however we do not investigate the reasons in this paper.

b) Spatial Churn: Not all IP addresses respond to all three control locations, even though we initiated the control scans all at the same time for each run. One potential cause for this phenomenon is wide-area routing issues. We identify IP addresses that only successfully responded to one or two locations (not all three) as reflecting spatial churn, corresponding to about 3.66% (about 3.7 million) of responding IP addresses across the footprints from the three control nodes. Upon further investigation, we observed that 52% of this spatial churn arose from IP addresses accessible from only *one* of the control

Exit Node	Location	Uptime	Bandwidth (MB/s)
Axigy1	USA	35 days	31.09
Axigy2	USA	76 days	31.46
NForce2	Netherlands	35 days	31.46
Voxility1	Romania	1 day 17 hr	16.99

TABLE II: Description of Tor exit nodes from which IPv4 scans were conducted.

nodes. We tested a handful of these IP addresses manually and confirmed this behavior, ruling out that it reflected a ZMap problem.

Given the significant amount of spatial and temporal churn, we settle on two definitions of web footprint to use for our analysis. *1*) a Lax definition, where we only remove cases of spatial churn. For this set, we consider the set of IP addresses for which all control nodes see a response at least once across the seven days. *2*) A Strict definition, where we remove cases of both spatial and temporal churn. We include in this set only IP addresses for which all control nodes received a successful response on all days. We find that the Raw footprint contains 103,329,073 IP addresses (2.82% of the probed set). Lax footprint is 96% of the Raw footprint, whereas Strict reduces the Raw footprint to 50%. For the purpose of reference and understanding the effect of network loss on our measurements (§ IV-F), we also report the numbers for the Raw footprint (response to any control node on any day).

E. Assessing Network-Layer Discrimination

Having gained confidence in our measurement methodology, we now turn to analyzing the resulting data. We conducted the scans from four high-bandwidth Tor exit nodes for 4 days (Aug. 10–13, 2015) (Table II). These represent 3% of aggregate Tor exit bandwidth. We note that each exit node hosts 2–3 Tor processes on the same interface. As our 100 Mbps scans use the same IP address as the Tor exit node, we turned off all but one Tor process on these machines for the duration of the experiment to minimize load on the interface and potential packet loss on the interface and/or the outgoing link. These preventive measures helped reduce our reported `pcap` loss on the exit nodes to 0.001% of the typical number of responses seen per scan. We also chose Tor instances that use the same IP address for incoming and outgoing Tor traffic to allow our scans to trigger even 'lazy' blacklists.[1] For three of the exit nodes, we displayed our scan notice page on port 8080 instead of the usual port 80, as the latter already displayed a separate Tor abuse complaint page.

Our basic technique for flagging network-layer discrimination of Tor is to identify the part of the Web footprint that *never* produces a successful response to a Tor exit node. We examine this separately for each exit node, as we do not assume that all the exits are blocked consistently. Once we have extracted this subset for an exit node, we scan the suspicious IP addresses 5 times from the corresponding exit node and discard IP addresses that respond successfully at least once, effectively reducing our false positives. As a result of this last step, the blocked IPs per exit node reduce on average by 7.70% (σ=2.82%) for Raw footprint, 8.94% (σ=3.23%) for Lax footprint, and 1.05% (σ=0.74%) for Strict footprint. We note that our approach does not account for transient IP layer blocking such as abuse-based filtering. However, assuming that transient IP blocking is enforced for a time window smaller than 4 days, we may still observe a successful response in scans conducted before or after the transient block. Using this methodology, we characterize Tor blocking for both Lax and Strict Web footprints.

Table III shows the breakdown of the Tor blocking we detect. We detect a significantly higher rate of blocking for the Lax footprint compared to Strict (13.01–16.14%, and 1.23–2.59%, respectively). This discrepancy could be caused by multiple factors. First, the Lax footprint is more than double the Strict footprint, due to the weaker selection criteria. This means that it is likely to see larger churn and therefore has a larger potential for false positives. Second, as we see next, the Lax footprint exposes large access ISP networks, which potentially block Tor across the whole network. Due to the transient nature of nodes in such networks, they are less likely to be seen in the Strict footprint.

Tables IV and V show the breakdown of the ASNs that block Tor. Tables IV shows the distribution by the number of IP addresses in an ASN that block Tor, for both the Lax and the Strict footprints. We see that the ASNs in the Strict footprint are dominated by hosting services, which suggests that could be policy or abuse-driven. The Lax footprint contains ASNs that are potentially access and mobile ISPs, such as CHINANET, BSNL, and Airtel. These ISPs likely enforce symmetric blocking of Tor. Because they are access ISPs, nodes in these networks are more likely to go offline, which explains their absence in the Strict list. We note that ASes of IPs in Lax footprint that block Tor traffic mostly originate in countries that are notoriously known for their censorship practices, such as China and Iran. Thus far these countries have been reported to block *access* to Tor network, but our results suggest that traffic coming *from* Tor network may also be blocked either as a policy or as an unintended effect of the mechanism of censorship chosen.

Table V shows a similar result sorted by the proportion of servers within a given ASN that block Tor. We see a higher prevalence of hosting sites in both Lax and Strict.

F. Calculating effect of network loss

The possibility of losing our probes or their responses due to packet-loss introduces uncertainty as to whether a given IP address specifically blacklists Tor traffic. In this section we develop a Bayesian analysis of this uncertainty so as to provide error bounds on the estimates we derive from our measurements of Tor-blocking.

For the purposes of error calculation we assume that IP addresses fall into one of four categories: allowing responses to all probes received (A), denying responses to all probes (D), blacklisting probes from Tor nodes, but otherwise responding (B), and whitelisting probes from Tor nodes, but otherwise

[1] The easiest approach to blacklist Tor is to block IP addresses from the node descriptors in the directory consensus that denote the incoming IP address for nodes. This blacklisting approach fails to cover nodes that use a different IP address for outgoing traffic, per Section II-B.

Footprint	IP Addresses	Axigy1 (%)		Axigy2 (%)		NForce2 (%)		Voxility1 (%)	
		orig.	ret.	orig.	ret.	orig.	ret.	orig.	ret.
Raw	103,329,073 (2.82%)	16.05	15.48	15.45	14.01	17.66	16.18	16.20	14.65
Lax	99,547,512 (2.72%)	14.09	13.50	13.68	12.19	16.14	14.59	14.63	13.01
Strict	52,148,437 (1.42%)	1.91	1.91	1.25	1.23	2.59	2.55	1.88	1.82

TABLE III: Web footprint blocked across exit nodes. We show footprint as % of probed IP addresses (3,662,744,599). For each exit node, we present the original (*orig.*) block proportion of the footprint and that retained (*ret.*) after weeding out false positives using 5 verification scans.

Axigy1 (13.50%)	Axigy2 (12.19%)	NForce2 (14.59%)	Voxility1 (13.01%)
CHINA169-BACKBONE,CN (11.33)	CHINA169-BACKBONE,CN (11.73)	CHINA169-BACKBONE,CN (11.02)	CHINA169-BACKBONE,CN (12.53)
CHINANET-BACKBONE,CN (7.42)	CHINANET-BACKBONE,CN (8.20)	CHINANET-BACKBONE,CN (7.30)	CHINANET-BACKBONE,CN (7.93)
Uninet S.A.,MX (3.43)	DCI-AS(ITC),IR (3.26)	AIRTELBROADBAND-AS-AP,IN (4.31)	DCI-AS(ITC),IR
DCI-AS(ITC),IR (2.94)	Uninet S.A.,MX (3.00)	BSNL-NIB,IN (4.30)	Uninet S.A.,MX
BSNL-NIB,IN (2.94)	DTAG Deutsche Telekom,DE (2.89)	DCI-AS(ITC),IR (2.73)	DTAG Deutsche Telekom,DE

(a) Lax Web Footprint (99,547,512 IP addresses forming 2.72% of probed IPv4)

Axigy1 (1.91%)	Axigy2 (1.23%)	NForce2 (2.55%)	Voxility1 (1.82%)
MCCI-AS,IR (11.91)	MCCI-AS,IR (18.44)	MCCI-AS,IR (8.92)	OCN NTT,JP (20.90)
RMH-14-Rackspace,US (10.87)	DREAMHOST-AS,US (13.07)	RMH-14-Rackspace,US (8.14)	MCCI-AS,IR (12.46)
RACKSPACE-Rackspace,US (9.92)	KUNET-AS,KR (3.59)	RACKSPACE-Rackspace,US (7.43)	DREAMHOST-AS,US (8.83)
DREAMHOST-AS,US (8.44)	REDSTATION,GB (2.66)	DREAMHOST-AS,US (6.32)	GO-DADDY-COM-LLC,US (3.51)
Rackspace Ltd.,GB (5.85)	SINGLEHOP-INC,US (2.09)	BBIL-AP BHARTI Airtel,IN (5.59)	FBDC FreeBit,JP (2.56)

(b) Strict Web Footprint (52,148,437 IP addresses forming 1.42% of probed IPv4)

TABLE IV: ASN distribution (top 5) of IP addresses that block Tor across exit nodes for Lax and Strict. For each exit node, we show the percentage of footprint that blocks it, and ASN distribution (%) of blocking IP addresses in the footprint.

Axigy1	Axigy2	NForce2	Voxility1
MCCI-AS,IR	DREAMHOST-AS,US	RMH-14-Rackspace,US	OCN NTT Communications,JP
RMH-14 - Rackspace,US	KUNET-AS,KR	RACKSPACE - Rackspace,US	DREAMHOST-AS, LLC,US
RACKSPACE - Rackspace,US	REDSTATION,GB	AIRCEL-IN Aircel Ltd.,IN	KUNET-AS,KR
DREAMHOST-AS, LLC,US	LLC-SK-CONTINENT,RU	DREAMHOST-AS, LLC,US	BEKKOAME INTERNET INC.,JP
CNNIC-SGATHER-AP,CN	tropicalweb-as,MZ	Rackspace Ltd.,GB	tropicalweb-as,MZ

(a) Lax Web Footprint

Axigy1	Axigy2	NForce2	Voxility1
MCCI-AS,IR	MCCI-AS,IR	MCCI-AS,IR	OCN NTT Communications,JP
RMH-14 - Rackspace,US	DREAMHOST-AS,US	RMH-14 - Rackspace,US	MCCI-AS,IR
RACKSPACE - Rackspace,US	KUNET-AS,KR	RACKSPACE - Rackspace,US	DREAMHOST-AS,US
DREAMHOST-AS,US	REDSTATION,GB	DREAMHOST-AS,US	KUNET-AS,KR
Rackspace Ltd.,GB	AS-INTERMEDIA,US	Rackspace Ltd.,GB	BEKKOAME INTERNET INC.,JP

(b) Strict Web Footprint

TABLE V: ASN distribution (top 5) by fraction of IP addresses in their subnet that block Tor across exit nodes for Lax and Strict. As multiple ASNs block Tor 100%, we further order them by ASN size (the number of IP addresses in an ASN).

	$\mathrm{P}(T, NT\} \mid A)$	$\mathrm{P}(T, NT\} \mid W)$	$\mathrm{P}(T, NT\} \mid B)$	$\mathrm{P}(T, NT\} \mid D)$	$\mathrm{P}(T, NT\})$
$\mathrm{T} = 0$	n^2	n^2	1	1	$(a+w)n^2 + b + d$
$\mathrm{T} \in \{1, 2\}$	$(1-n)^2 + 2n(1-n)$	$(1-n)^2 + 2n(1-n)$	0	0	$-\left(n^2 - 1\right)(a+w)$
$\mathrm{NT} = 0$	n^2	1	n^2	1	$an^2 + bn^2 + d + w$
$\mathrm{NT} \in \{1, 2\}$	$(1-n)^2 + 2n(1-n)$	0	$(1-n)^2 + 2n(1-n)$	0	$-(a+b)\left(n^2 - 1\right)$

TABLE VI: Likelihood of scan outcomes: conditioned on category of target IP address and unconditional.

denying (W).[2] In this context, we treat responding with an error the same as denying responses.

We cannot directly observe which category a node falls into, but we send two probe packets from either a Tor node or non-Tor node and then count the number of responses (0, 1 or 2) from each: T for the number of successful responses from the Tor node, and NT for non-Tor node. To estimate the category given a count we must know the likelihood $\mathrm{P}(T \mid c)$ and $\mathrm{P}(NT \mid c)$ for each category $c \in C$ where $C = \{\mathrm{A}, \mathrm{W}, \mathrm{B}, \mathrm{D}\}$.

[2] While the whitelisting case seems unlikely, we need to consider it for a complete analysis.

	$\mathrm{P}(A \mid T, NT\})$	$\mathrm{P}(W \mid T, NT\})$	$\mathrm{P}(B \mid T, NT\})$	$\mathrm{P}(D \mid T, NT\})$
$\mathrm{T} = 0$	$\frac{\mathrm{an}^2}{(\mathrm{a}+\mathrm{w})\mathrm{n}^2+\mathrm{b}+\mathrm{d}}$	$\frac{\mathrm{n}^2\mathrm{w}}{(\mathrm{a}+\mathrm{w})\mathrm{n}^2+\mathrm{b}+\mathrm{d}}$	$\frac{\mathrm{b}}{(\mathrm{a}+\mathrm{w})\mathrm{n}^2+\mathrm{b}+\mathrm{d}}$	$\frac{\mathrm{d}}{(\mathrm{a}+\mathrm{w})\mathrm{n}^2+\mathrm{b}+\mathrm{d}}$
$\mathrm{T} \in \{1, 2\}$	$\frac{\mathrm{a}}{\mathrm{a}+\mathrm{w}}$	$\frac{\mathrm{w}}{\mathrm{a}+\mathrm{w}}$	0	0
$\mathrm{NT} = 0$	$\frac{\mathrm{an}^2}{(\mathrm{a}+\mathrm{b})\mathrm{n}^2+\mathrm{d}+\mathrm{w}}$	$\frac{\mathrm{w}}{(\mathrm{a}+\mathrm{b})\mathrm{n}^2+\mathrm{d}+\mathrm{w}}$	$\frac{\mathrm{bn}^2}{(\mathrm{a}+\mathrm{b})\mathrm{n}^2+\mathrm{d}+\mathrm{w}}$	$\frac{\mathrm{d}}{(\mathrm{a}+\mathrm{b})\mathrm{n}^2+\mathrm{d}+\mathrm{w}}$
$\mathrm{NT} \in \{1, 2\}$	$\frac{\mathrm{a}}{\mathrm{a}+\mathrm{b}}$	0	$\frac{\mathrm{b}}{\mathrm{a}+\mathrm{b}}$	0

TABLE VII: Posterior probability of IP addresses falling into each category given a scan outcome.

If n gives the probability of the loss of a probe packet or its corresponding response, then, assuming independence, the probability of losing 0, 1 or 2 packets is $(1-n)^2$, $2(1-n)n$, and n^2 respectively. Although the two probes are sent with a 6.7 s delay to reduce correlated loss, we will show how to drop the assumption of independence later. We assume that for W, $NT = 0$; for B, $T = 0$; and for D, $T = 0 \wedge NT = 0$, which corresponds to the definition of the RAW web footprint—a single non-Tor response means that the IP address is not in D, and a single Tor response means that the address is not in B. Given these definitions, Table VI shows the likelihood of each response given a category, and we can then calculate $\mathrm{P}(T) = \sum_{c \in C} \mathrm{P}(T \mid c)\mathrm{P}(c)$ and $\mathrm{P}(NT) = \sum_{c \in C} \mathrm{P}(NT \mid c)\mathrm{P}(c)$. If we know the network loss rate, receiving 2 packets in response to a pair of probe packets does not add any more information than receiving 1 so we combine these two cases. We define the prior probability of an IP address being in each category as a for P(A), b for P(B), d for P(D) and w for P(W).

Using Bayes law, we can now calculate the probability that an IP address is of each category, given the count of probe responses: $\mathrm{P}(c \mid T) = (\mathrm{P}(T \mid c)\mathrm{P}(c))/\mathrm{P}(T)$ and $\mathrm{P}(c \mid NT) = (\mathrm{P}(NT \mid c)\mathrm{P}(c))/\mathrm{P}(NT)$, as shown in Table VII. We can drop the assumption about independent packet loss probabilities by noting that the value for n is never used directly, only n^2, so we can specify n^2 directly as the probability of the loss of two consecutive packets, taking into account any dependence between loss probabilities.

We perform the probability calculation for each IP address by selecting a row from Table VII based on the type of scan (Tor or non-Tor) and number of probe responses, then applying each formula to update the estimate of P(A), P(W), P(B) and P(D). The formulae depend on the network loss rate n^2, as well as the previous estimates of P(A), P(W), P(B) and P(D) (a, w, b and d). The first time we calculate the probabilities we must provide prior probabilities (for which we use the uniform distribution), but the effect of this choice diminishes as we consider each new set of probe results.

Note that if we receive at least 1 response to a non-Tor probe, $w = d = 0$ and thus $\mathrm{P}(\mathrm{B} \mid NT \in \{1, 2\})$ becomes equal to $\mathrm{P}(\mathrm{B} \mid NT = 0)$. Therefore the probability that a target IP address is blacklisting Tor depends only on n^2, the prior probability estimates, whether we ever receive a response to a Tor probe (as then $\mathrm{P(B)} = 0$); whether we receive a response to at least one non-Tor probe (then $\mathrm{P(W)} = \mathrm{P(D)} = 0$); and the total number of scans we performed.

To estimate n^2 we average how often an IP address appears to be inaccessible from one site, despite being accessible from at least one of the others. This is always less than 1% so we take this as a conservative estimate for n^2. We can then compute P(B) for each category of IP address: those outside of the RAW footprint ($NT = 0$ for all 7 non-Tor scans), and those inside the footprint with $T = 0$ for all 4 Tor scans have non-zero P(B). IP addresses with $T > 0$ for any scan have $\mathrm{P(B)} = 0$. From these probabilities we can estimate the expected number of blacklisting nodes by multiplying the number of IP addresses of each category (count) by P(B) for that category, and taking the normal approximation of the binomial distribution we can find the 95% confidence interval as 3 standard deviations of the normal distribution with $\mu = \mathrm{count} \times \mathrm{P(B)}$ and $\sigma^2 = \mathrm{count} \times \mathrm{P(B)}(1 - \mathrm{P(B)})$.

Of the 3,662,744,599 IP addresses scanned, 103,329,073 are inside the RAW footprint leaving 3,559,415,526 outside. For those IP addresses outside the footprint $\mathrm{P(B)} = 5 \times 10^{-15}$ and so the expected number of blacklisting nodes missed is 0.000018 ± 0.012. Of the IP addresses inside the footprint which never responded to any of the Tor-scans, $\mathrm{P(B)} = 1 - 1 \times 10^{-8}$. Therefore the expected number of blacklisting IP addresses is between 0.15 and 0.16 less than the count for each exit node studied, with 95% confidence interval between ±1.16 and ±1.22. So overall we can conclude that the number of scans performed are sufficient to almost completely eliminate the effect of network loss.

V. APPLICATION-LAYER DISCRIMINATION

We have seen that Tor exit nodes encounter a restricted Internet at layers 3/4. In this section we describe our experiments to measure layer 7, i.e., application-layer blocking of Tor users. We base our observations on two data sources: *1*) five days of our own intensive scans of 1,000 URLs from a control server and through every Tor exit node; and *2*) a year's worth of paired Tor/non-Tor scans of over 2,300 URLs from the Open Observatory of Network Interference (OONI). OONI is a global network measurement platform aimed at detecting censorship and surveillance, one of whose tests is particularly suited to our study.

There are two main ways Tor users may find themselves blocked by a server. The server may block Tor users specifically, using a blacklist of Tor exit node addresses. The only maintenance required is keeping the blacklist up to date. Alternatively, Tor users may simply be caught up in an automated blocking system that does not target Tor per se, but merely reacts to the consolidated traffic of the many users that come from an exit node. Perhaps the most conspicuous current example of this phenomenon is CloudFlare's 'Attention Required!' CAPTCHA page. CloudFlare is a large content delivery network (CDN) that by default assesses the 'reputation' of each client IP address in terms of how much malicious traffic it has been observed to send, and blocks attempted access by clients with sufficiently poor reputations.

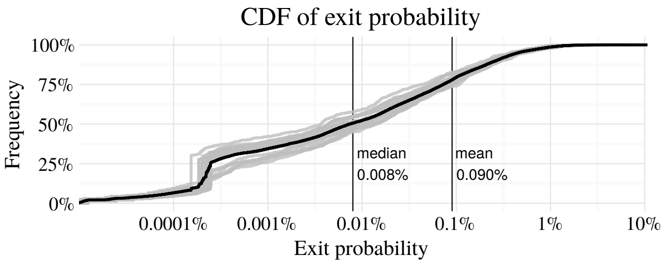


Fig. 3: The cumulative distribution of exit probability from 20 randomly sampled Tor consensuses since September 2014. The bulk of exits have between 0.001% and 1% probability of being selected. The largest exits tend to have an exit probability around 5%. The notable rise just above 0.0001% is an artifact of the bandwidth measurement process; when a node's bandwidth is unmeasured for whatever reason, it receives a default bandwidth of 20 KB/s, giving it a very low exit probability of around 1 in 500,000.

A CloudFlare support page explains that while they do not specifically target Tor users, 'due to the behaviour of some individuals using the Tor network (spammers, distributors of malware, attackers, etc.), the IP addresses of Tor exit nodes generally earn a bad reputation' [3].

Some sites—mainly larger web properties, though not exclusively—apparently implement their own detection algorithms and custom block pages. Notable examples in this category are Craigslist and Yelp. Many other sites simply inherit the blocking behavior of their web host or content delivery network, which may or may not offer customers control over the severity of blocking. In this latter case, a single provider's policy can affect many websites.

Our first experiment—contemporary scans of Alexa URLs—provides broad coverage across all Tor exit nodes over a short time period. The second experiment—analysis of historic OONI scan data—covers a long time span and more URLs, but lacks a longitudinal comparison across all exits for each URL. Figure 3 illustrates the rationale for conducting our own scans in addition to analyzing past data. Tor clients do not choose exit nodes with equal probability; each exit is weighted according to its bandwidth [23, §3.8.3]. Faster nodes have a greater probability of being used (subject to some other constraints such as exit policies). The OONI data reflects Tor circuits made in the ordinary fashion; therefore low-probability exits are rarely represented. Measuring low-probability exits is important because it helps to distinguish the two kinds of blocking: slow exits will appear on blacklists but will have fewer users and thus be less likely to exceed abuse thresholds.

Our measurements are limited to home pages, except for about 3% of OONI URLs that include a path component. The Alexa URLs are only home pages. We know through experience that some application-layer blocking only becomes apparent when accessing certain deeper features or pages of websites. For example, Wikipedia allows Tor users to read but not to edit articles [28]. Google allows access to its home page but may present a CAPTCHA or block page when doing a search. Bank of America does not permit logging in. We did not undertake to discover such deeper blocks, which would require extensive additional methodology to conduct in a large-scale fashion.

A. *Contemporary Scans*

To measure the differential treatment of Tor users, we visit Alexa top 1,000 URLs once from all available Tor exit nodes and once without Tor. For the former, we used Exitmap [29], a fast and extensible Python-based scanner for Tor exit nodes. Exitmap uses Stem [12] to connect to the Tor network, and enables running a module over all available exit nodes. It is designed to monitor the reliability and trustworthiness of exit nodes [30] but its basic architecture is generic and it can be used to run any query.

Exitmap downloads a Tor consensus and extracts the currently available exit nodes. It then initiates circuits using the selected exit nodes as their last hop. To improve the performance of the scanning process, Exitmap uses two-hop circuits instead of the default three-hop circuits. (Using single-hop circuits is not an option because by default, exit nodes do not allow direct connection from other non-Tor IP addresses [14]; additionally the authors of Exitmap argue that one-hop circuits may permit exit node operators to treat scanning connections differently [30, §3.2].)

To measure discrimination against Tor, we send many HTTP requests per URL: with Tor through every available exit node, and one without Tor. We use Exitmap to build Tor circuits and a Python program to send the HTTP requests. Our experiment of downloading thousands of URLs per exit node stretched Exitmap past its original design parameters, requiring us to overcome some scanning challenges. Downloading a single URL using all the Tor exits requires 45–50 minutes on average; however much of this time consists of circuit-construction overhead. We reduced the total scanning time by running 5 instances of Exitmap in parallel, and downloading 20 URLs at a time through each circuit. With these changes, visiting 100 URLs through every Tor exit node takes around 1–2 hours on average. By default, Tor rebuilds circuits and streams each hour. We set configuration parameters to prevent this.

We collected data for 5 days, from August 10 to August 14, 2015. We selected exit nodes that allow traffic through port 80 and 443. Different runs of Exitmap can select different exit nodes for two reasons: *1*) The Tor directory authorities release a fresh consensus on available nodes every hour. New nodes might appear, old nodes might disappear, and nodes can change their 'exit policy' of allowed ports. Thus the available exit nodes can change every hour. *2*) To build circuits Tor clients need to download 'enough' of the network so that they can construct a sufficiently large number of the possible paths through the network. By default, Tor uses a value chosen by the directory authorities, which can change every hour. Thus, different exit nodes might be selected at different times. During our data collection period, we collected data from 899–915 exit nodes. The distribution of the exit probabilities in our dataset is within the regular range as seen in Figure 3. We have a good mixture of both high- and low-probability exit nodes. We noticed that 83%–89% of the circuits succeeded. This success rate is similar to previous exit scanning studies [30].

Table VIII shows the total number of HTTP requests sent and responses received on each day. We intend to send 1,000 HTTP requests through over 900 exit nodes. The number of HTTP requests per exit node should be 1,000, but the average

Date	# Exits	Request		Response (`200`)		Response (non-`200`)		Non-HTTP Errors	
		Control	Per Exit	Control	Per Exit	Control	Per Exit	Control	Per Exit
Aug 10	908	1000	741.29	992	641.93	1	26.03	7	73.33
Aug 11	915	1000	679.43	985	595.21	3	23.93	12	60.29
Aug 12	905	1000	735.66	986	632.96	6	28.22	8	74.48
Aug 13	915	1000	735.46	989	639.44	4	26.84	7	69.18
Aug 14	899	1000	738.22	989	641.55	2	28.18	9	68.49
Average	908	1000	726.01	98.82%	86.81%	0.32%	3.67%	0.86%	9.53%

TABLE VIII: This table shows the total number of HTTP requests sent and responses received during the 5 days of scanning. On average, over 3.67% of the Alexa top websites discriminate against Tor (p-value = 0.008)

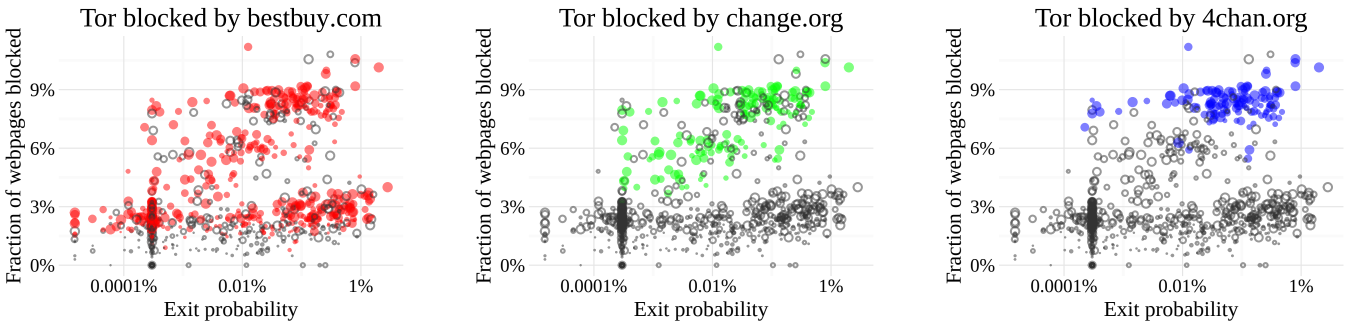


Fig. 4: These graphs show the relationship between blocking rate and exit probability. Each dot represents one exit node. The size of a dot represents the age of an exit (when the node became an exit) in days. Bigger nodes are older than the smaller nodes. The colored dots represent exit nodes blocked by different sites. (a) The leftmost graph shows the number of exit nodes blocked by `bestbuy.com`, which is hosted on Akamai. `bestbuy.com` and other Akamai sites block over 60% of the exit nodes. (b) The middle graph shows the number of exit nodes blocked by a site hosted on CloudFlare (`change.org`). (c) The last graph shows the number of exit nodes blocked by another site hosted on CloudFlare (`4chan.org`). `4chan.org` uses a less strict setting than `change.org`, and thus blocks mostly high probability and older exit nodes.

number of requests per exit node was less than 800. This discrepancy occurs for two reasons: *1*) As mentioned in the previous section, different exit nodes can be selected during different runs of Exitmap because the available exit nodes can change every hour. Since our crawl takes over 10 hours to finish, some of the exit nodes were not available for the whole span of crawling. *2*) Even when the same exit nodes are selected, some exit nodes are incapable of handling 1,000 HTTP requests because of resource limitations.

We consider a website to discriminate against Tor when it responds with a `200` 'OK' status code when visited without Tor, and some other valid but non-`200` level status code when visited with Tor. On average, around 3.67% of the Alexa top 1,000 websites responded with a non-`200` status code when visited through Tor (Table VIII), whereas only at most 6 sites responded with a non-`200` code when visited without Tor. To check whether this difference is statistically significant, we computed the p-value using permutation test under the null hypothesis of independence. We chose the permutation test because it does not assume that the responses of the experimental units are independent and identically distributed. In our case, many of the top Alexa 1,000 websites are hosted on the CloudFlare and Akamai CDNs. For all of these sites, whether they send a non-`200` response to a Tor exit is not independent. The p-value of the permutation test is 0.008 which shows that web visits through Tor receive different treatment from the websites. We also encountered around 8% non-HTTP errors such as timeouts and connection resets, which can be caused by discrimination at layers 3 and 4.

We noticed that no site blocks all Tor exit nodes (Figure 5). During these five days, on average 15.6 sites out of the Alexa 1,000 top sites blocked over 60% of Tor exit nodes. These sites include `yelp.com` (up to ≈ 70% exit nodes blocked), `macys.com` (up to ≈ 60%), and `bestbuy.com` (up to ≈ 66%). The majority of these sites are hosted on Akamai and Amazon Web Services. The websites on Akamai all show a `403` 'Access Denied' block page, which a user cannot bypass. Yelp and Craigslist have their own block page. Some websites, e.g., `macys.com`, return a redirect error that often leads to an infinite redirect loop. On average, around 69 sites block over 10–50% of the Tor exit nodes (Figure 4). The majority of these websites are hosted on CloudFlare. On average, around 150 websites block less than 10% of the Tor exit nodes and the rest of the websites (over 700) do not block any exit nodes at the home page.

To check whether these blocking events are abuse-based or Tor-specific, we looked at the age and exit probability of exit nodes. We assume that abuse-based blocking is more likely to block old or high-probability exit nodes because they have more opportunity to attract abuse, while blacklist-based or Tor-specific blocking tends to block all exits equally. We downloaded node exit probabilities and ages from Onionoo [20]. We used logistic regression to determine the effect of the exit characteristics on blocking rate.[3] Overall (across all measured sites), we did not find any statistically significant effect; however for specific websites and specific blockers we found significant effects. We manually tested three sites: `bestbuy.com` (on Akamai), `change.org` (on CloudFlare),

[3] We acknowledge that logistic regression requires each observation to be independent which might not be true in our case. We chose logistic regression because it can handle non-linear relationship and can provide an estimation of the effect.

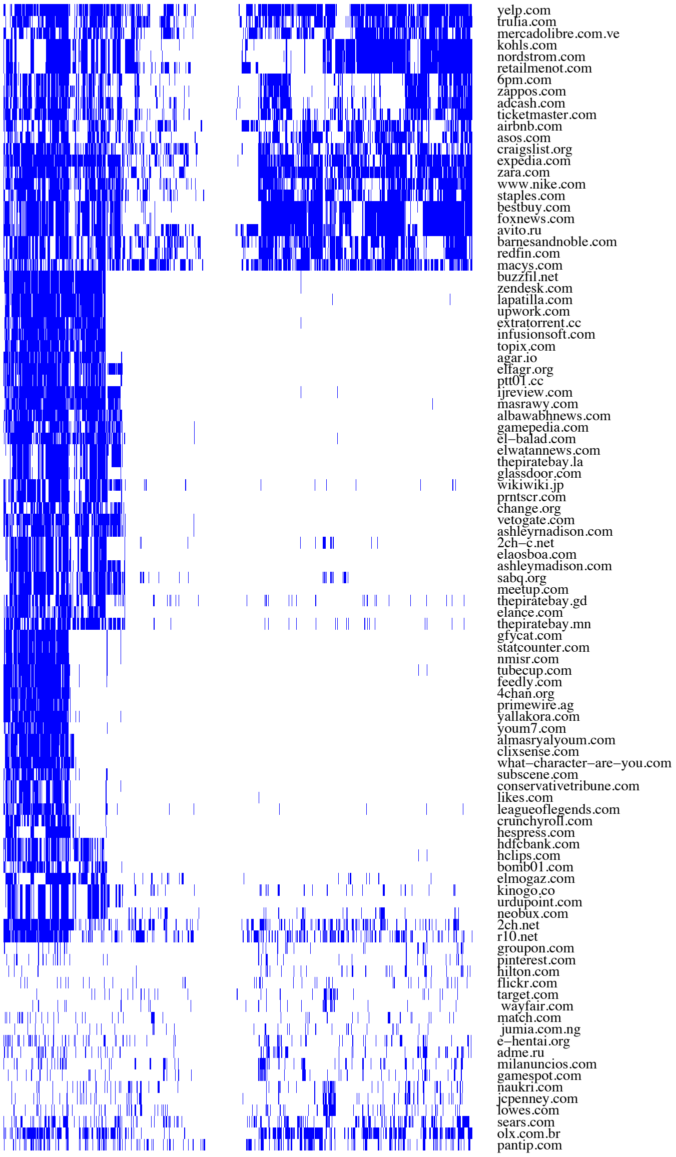


Fig. 5: This figure shows 100 websites among the Alexa top 1,000 sites that block most Tor exit nodes. Each row represents one website. Each column represents one exit node. A blue bar represents a blocking event; that is, the web server responded with a `200` status code when accessed without Tor and another valid but non-`200`-level HTTP response when accessed with Tor. No site blocked every exit node. During our scan, on average 15 sites blocked over 50% of the exit nodes; `yelp.com`, at the top of the figure, is an example of such a site with a high blocking rate.

and `4chan.org` (on CloudFlare with an apparently lower security configuration). For `bestbuy.com`, both exit probability (odds ratio = 2.4 per 1% change in exit probability with p-value = 0.0098) and age (odds ratio = 1.002 per day of age with p-value < 0.001) have an effect on blocking frequency. For `change.org`, the effect of exit probability was not significant, but age had a slightly greater effect (odds ratio = 1.003 per day with p-value < 0.001) than with the Akamai-hosted `bestbuy.com`. For `4chan.org`, exit probability had a moderate effect (odds ratio = 1.9 per 1% with p-value < 0.001) and age had an even greater effect (odds ratio = 1.004 per day with p-value < 0.001) than the other site hosted on CloudFlare. We saw that the two CloudFlare sites did not block any exits younger than about 30 days, while the Akamai site did. Figure 4 compares the different subsets of exits blocked by the three sites.

Some sites do not block Tor on their home page but block other pages or functions. We conducted a small ancillary experiment on search engines with URLs containing search queries (as opposed to the URLs of home pages). The home page of `google.com` is not blocked for any Tor exit node, but searching on Google was blocked for 23–40% of them (varying on different days). We noticed similar behavior in Yahoo!, where searching is blocked for around 1% of the exits, but the homepage is always accessible.

We found 42 exit nodes that were not blocked by any of the Alexa top 1,000 websites during our 5-day crawl. These exits do not appear to be dedicated Tor exit nodes. All the exits have similar characteristics: *1*) All except one are hosted on Amazon EC2; *2*) their node bandwidth was unmeasured; *3*) all of the exits are turned on and off periodically and were never up consecutively for more than a month; *4*) most of exits were started and stopped at the same time. We suspect that these exit nodes were unblocked either because of their low bandwidth (20 KB/s) or low uptime.

B. *Historical Perspective from OONI*

For a historical record of Tor blocking, we drew upon scan data published by OONI, the Open Observatory of Network Interference [17]. Volunteers run a program called ooniprobe that runs a variety of network tests and reports the results to a central collector. The network tests are designed to detect behavior such as DNS tampering, blocking of anticensorship proxies, and manipulation of HTTP headers. The oldest reports are from December 2012 and they continue to the present.

One of ooniprobe's several tests, http_requests [16], suits our purpose of detecting differential treatment of Tor users very well. The test takes as input a list of URLs. It downloads each URL twice: once with Tor, and once without. Both downloads happen within a few seconds of each other. The results consist of a set of (Tor, non-Tor) *request pairs*. Each request in a pair maps to a response: either an HTTP response with status code, header, and body; or an indication that an error occurred, such as a timeout or rejected connection.

The http_requests test was intended to discover blocking by the local network, with the Tor request serving as a control (uncensored request). We turn the intended methodology on its head, using the non-Tor request as a control and observing how the response to the Tor request differs. Within a single execution of http_requests, each URL is downloaded through Tor only once, through a single exit node. The same exit node is reused for multiple URLs, but changes over time (even within a report) as circuits are naturally rotated. Path selection favors exit nodes with higher bandwidth (per Figure 3), meaning that larger exit nodes get tested more often. However, the large number of available OONI reports means that all but the rarest exit nodes receive at least some representation.

The list of tested URLs varies across reports. For the most part, they use the Citizen Lab URL testing lists [2], which consist of about 1,200 'global' URLs, plus up to about 900 additional country-specific URLs that depend on the geographical location where ooniprobe runs. (Versions of ooniprobe before October 2014 tested a static list of 1,000 URLs derived from the Alexa top sites.) There are also reports that use a manually specified URL list. Therefore some URLs are tested more often than others. We only considered URLs tested at least 100 times.

Up through July 20, 2015, the raw OONI http_requests data consist of 2,505 reports, 2,574,326 (Tor, non-Tor) request pairs, and 102,865 distinct URLs. We applied a number of restrictions to the raw data to obtain a subset useful for our analysis:

- We discarded reports before September 2014. Reports from after this date (82% of the total) occur more regularly than those from before it.
- We discarded URLs with a small number (<100) of request pairs. The great majority of distinct URLs were tested only a handful of times and thus not appropriate for our analysis. Though only 2% of URLs occur often enough, they account for 89% of all request pairs.
- We discarded request pairs where one or both responses were missing. A response to an http_requests probe is supposed to be either an HTTP response (i.e., with a status code such as `200`), or else an indication of timeout or rejection. About 20% of request pairs are anomalous and are missing a response data structure, but they are concentrated in a tiny fraction of reports and URLs.

In brief, we sought URLs that had been frequently sampled, at close time intervals, that had meaningful response data. After applying all these restrictions together, there remained 1,969 reports, 1,727,138 request pairs, and 2,387 unique URLs.

Our basic analysis technique compares the Tor and non-Tor responses in each request pair. We specifically look for cases where the Tor request is blocked and the non-Tor request is unblocked. We consider a URL 'blocked' if the request *1*) timed out, *2*) was rejected, or *3*) received an HTTP response with status code `400` or higher. We treated redirect status codes like `302` as 'unblocked', though it would have caused us to miss blocks that are effected by a redirect to a block page, because it is common for sites to serve a redirect for the base URL (to add or remove a 'www' domain prefix, or to redirect HTTP to HTTPS, for example). We also special-cased certain other responses, such as HTTP `408` 'Request Timeout', which occurs when the client does not send its request in time, and is more likely measurement error than blocking. This method of classifying based on status codes is crude, but it is tenable precisely because we have paired simultaneous Tor and non-Tor responses. If a Tor request received an HTTP `403` 'Forbidden' response, it does not in itself indicate differential treatment of Tor users. But if, at the same time, a non-Tor request receives an HTTP `200` 'OK' response, it serves as evidence that the server treats Tor users differently. If both requests time out, say, or both succeed, then we would not consider it an instance of discrimination against Tor. For our purposes, we consider the case where Tor is unblocked and non-Tor is blocked (which is what the OONI http_requests test was meant to find) the same as both being unblocked (i.e., no negative Tor discrimination). This method of comparing paired responses does away with some of the difficulties in distinguishing variations that arise due to blocking and benign variations based on geolocation, for example.

There are some limitations to our approach. Sometimes servers return block pages with a non-error status such as `200`, and we will miss those. The results likely partially conflate Tor blocking with general anti-bot blocking; i.e., some blocks may be because of Tor, and others may be because of ooniprobe. We suspect this is the case for `www.amazon.com`, for example (see below). Some installations of ooniprobe run in censored places. Because of how we count responses, the worst that happens in that case is that we miss an instance of Tor discrimination (because Tor and non-Tor both appear to be blocked).

We now quantify the amount of blocking we observed in the OONI data. First, we give the overall rates of Tor versus non-Tor blocking. Recall that each request pair consists of a Tor and a non-Tor request, each of which may be blocked or unblocked, leaving four possibilities. The highlighted row is our focus of interest:

84.4%	Both requests unblocked
6.8%	Tor request blocked only
1.8%	Non-Tor request blocked only
7.1%	Both requests blocked

Drilling deeper, we find that a little more than half of the 6.8% Tor blocking happens at the application layer; i.e., block pages served as HTTP responses. The other blocks are transport-layer rejected connections and timeouts.

$$6.8\% = \begin{array}{ll}\left.\begin{array}{rl} 0.45\% & \text{rejects} \\ +\ 2.82\% & \text{timeouts}\end{array}\right\} & \text{transport layer} \\ \left.\begin{array}{rl} +\ 3.54\% & \text{HTTP}\end{array}\right\} & \text{application layer}\end{array}$$

Finally, we list the organizations that are responsible for the most Tor blocking. To categorize blockers, we wrote regular-expression classifiers and ran them against the OONI HTTP responses. Together these constitute the 3.54% figure in the previous table.

2.507%	CloudFlare (CDN)
0.362%	other
0.349%	custom
0.144%	Bluehost (web host)
0.126%	Akamai (CDN)
0.028%	Site5 (web host)
0.028%	Convio (web host)

CloudFlare is a content delivery network. It offers an abuse-based blocking system (turned on by default) that, when tripped, forces the user to complete a CAPTCHA before continuing to the site. The next row, marked 'other', includes all pages for which we did not write a specific classifier. The 'custom' row encompasses a wide variety of bespoke block pages belonging to one specific web site. Sites in this category include Craigslist and Yelp. Bluehost is a web hosting company. Akamai is a content delivery network. Site5 and Convio are web hosting companies.

Figure 7 illustrates the potential impact of a large centralized provider. Here, the blocking rate of CloudFlare sites suddenly drops, while other forms of blocking remain unchanged. This means it is possible for one company to have a unilateral effect on many users' browsing experience.

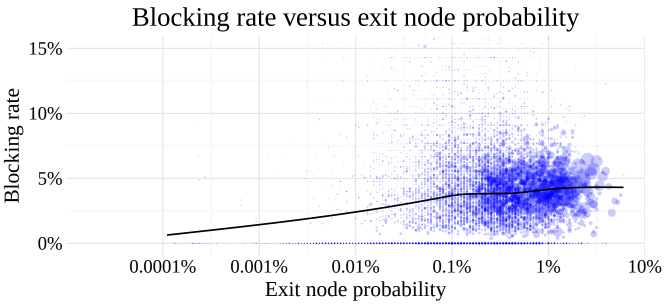


Fig. 6: As Tor exit probability increases, so does the incidence of blocking. Each dot represents a single exit node and its rate of being blocked, as seen from nearly a year of OONI measurements. The dark line shows a smoothed mean of the blocking rate. For clarity, the graph omits some points with a blocking rate above 15%; these constitute only about 0.5% of the data mass.

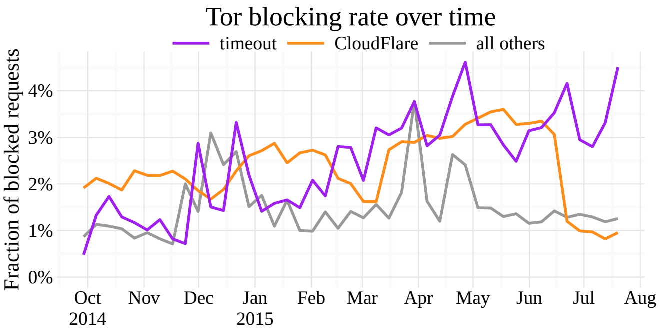
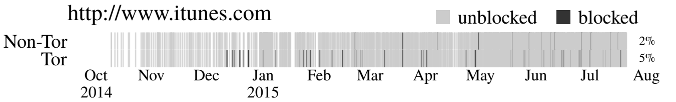


Fig. 7: Tor blocking rates over time. We have separated out CloudFlare blocks to illustrate both the fact that CloudFlare is the most common blocker (at least among the URLs in the OONI set), and how CloudFlare's rate of blocking decreased, possibly reflecting a policy change.

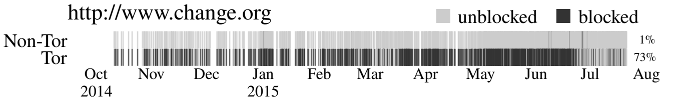
A small number of block pages explicitly targeted Tor users. The hosting company Convio sends a simple `501` (Not Implemented) page that says, 'Not Implemented Tor IP not allowed' and offers no opportunity to continue. The site `ezinearticles.com` serves a custom `403` (Forbidden) block page that says, 'It appears that you are using Tor anonymizing software. No Problem! We just need you to enter a Captcha so we can confirm that you are a person and not a bot.' The site permits browsing after solving the CAPTCHA.

We conclude this section with a sampling of time series that compare the patterns of Tor and non-Tor blocking for selected URLs. These URLs exemplify various types of blocking. It is possible to distinguish sites that employ a Tor blacklist, because they have near 100% rates of Tor blocking. We can readily link sites that share a CDN or web service provider by temporal patterns in their blocking. In the charts below, each request pair corresponds to a vertical strip across two rows: one for the non-Tor request and one for the Tor request. A light shade in the row means the request was unblocked and a dark shade means it was blocked.

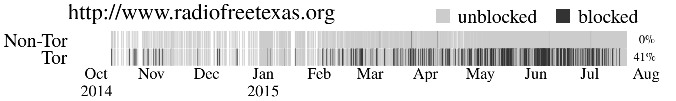
Blocking as a whole is not all that common. Most URLs manifest like this one, where potential blocking is scattered, intermittent, and rare:



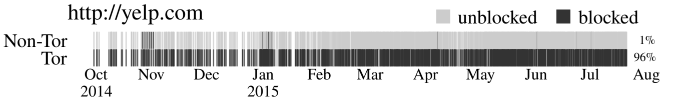
On the contrary, here is an example of a site running on CloudFlare, the largest source of blocking. Non-Tor is almost always accessible, while Tor is often—though not always—blocked. We have found that for sites such as this one, simply retrying the request with a different Tor circuit often makes the site accessible.



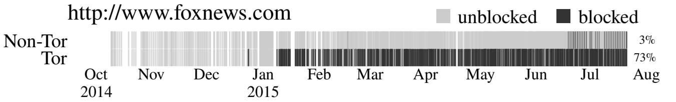
Another common blocker is a web hosting service called Site5. It also disproportionately blocks Tor visitors, though not at as high a rate as CloudFlare does.



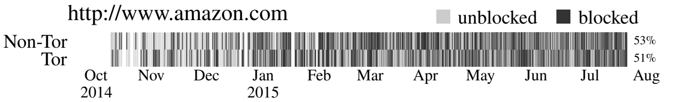
There are a few sites that evidently employ a blacklist of Tor exit nodes. Their rate of Tor blocking is nearly 100%.



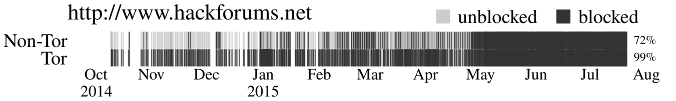
Similarly, some sites now have Tor blocks, but were not always that way. The server `www.foxnews.com` serves an Akamai block page to all Tor clients, but only began to do so in January 2015.



The web server at `www.amazon.com` is an interesting case because of its nearly equal blocking rates of both Tor and non-Tor traffic. We suspect that this kind of blocking is not in terms of abuse or proxy blocking per se, but rather probabilistic anti-bot or anti-crawling detection, that in this case detects ooniprobe as not corresponding to a human with a browser. The text on the block page supports this idea: 'To discuss automated access to Amazon data please contact. . . '



The site HackForums.net used to block both Tor and non-Tor visitors, Tor at a higher rate. In May 2015 the site announced on Twitter that 'Most countries aggressively blocked again. Sorry but the attack traffic, scammers and spam are not worth it.' All of ooniprobe's request became almost completely blocked. This reflects the website deploying anti-bot or anti-crawling detection to mitigate crawling and spamming by bots.



VI. Discussion

Based on our measurements we find instances of both blocking all traffic from Tor exits and cases of fate-sharing, where ASNs and websites block Tor exit traffic due to employing automated abuse-based filters. In the case of entities that preemptively block all Tor exit traffic, there is little that can be done beyond detecting instances of this occurring and publicizing the entities that do so. With abuse-based blocking, the potential for more precise filtering could enable benign users to avoid blocking that targets the abusive actions of other users. Here we discuss several potential methods of reducing this filtering, more precise abuse-based filtering, and minimizing the impact of this growing threat to people that use anonymity networks such as Tor.

A. Anonymous Blacklisting Systems

Anonymity networks, such as Tor, make it more difficult for ASNs and website operators to discern abusive users from benign visitors based on IP address, since they cloak the user's true IP address. This causes many automated and list-driven abuse-detection systems to blacklist some or all of the exit nodes' IP addresses.

Anonymous blacklisting systems [10], [26] have been proposed as a method to enable website operators to more precisely allow benign visitors to access their sites and hold abusive users accountable for their actions. The goal of anonymous blacklisting systems is to allow a website, such as Wikipedia, to block access to an individual anonymous abusive user without requiring a trusted third-party that can revoke a user's anonymity. This capability would allow websites to defend themselves against anonymous abusive users using similar methods as against identifiable users. Most anonymous blacklisting systems require users to anonymously register and authenticate with the anonymity network using blind signatures or zero-knowledge proof techniques, and create whitelists of permitted users. The registration process must require anonymous payment or otherwise bind users to scarce resources, such as IP addresses, to mitigate Sybil attacks.

Adoption of anonymous blacklisting systems has been negligible due to issues of degraded user privacy—anonymous blacklisting systems either offer pseudonymity instead of full anonymity or require a semi-trusted third-party to provide anonymity—and additional computational overhead [9]. If these issues could be addressed, anonymity blacklisting systems might be more widely deployed by anonymity networks and website operators, reducing the amount of explicit anonymity network blocking and fate-sharing experienced by users of these systems.

B. Contextual Awareness

It is conceivable that anonymity networks could reduce instances of abuse-based filtering by learning which websites were blocking certain exit nodes and reroute requests for these sites to another exit node that was not blocked. This would likely require application-layer analysis on the exit node that might be overly invasive from a privacy context.

A less privacy-invasive technique could involve the Tor Browser displaying a message when filtering is suspected. This could be done by including block page detection similar to that used in our study. The browser could also offer to retry the request using a different exit node. Both of these techniques could marginally reduce the impact of abuse-based blocking. However, neither of these approaches directly addresses fate-sharing issues caused by abuse-based blocking of Tor exit IP addresses. They also have the potential to trigger an "arms-race" as abusers could benefit from Tor spreading out abusive traffic causing more aggressive filtering of Tor traffic by impacted services.

C. Redesigning Anonymity Networks

Tor and other anonymity networks could attempt to recruit a larger pool of exit nodes that enables each exit to deliver a smaller amount of traffic. Our results find a (weak) correlation between the amount of traffic a node exits and the probability of a node's IP address being blocked due to automated abuse-based filtering. Thus, reducing the amount of traffic each node exits might reduce their probability of being blocked by automated abuse-based filtering. The risk of this and other techniques to fan out traffic to more IP addresses is that it might cause more websites to preemptively block *all* Tor exit traffic. This also does not deter abusive usage of Tor.

We could also consider disincentivizing large-scale abuse by charging Tor users for traffic usage. The BRAIDS [11] system proposed an anonymous payment scheme for improved quality-of-service originally with the goal of disincentivizing users from performing bulk downloads using Tor. BRAIDS could also be used to charge Tor users for traffic usage. This might reduce the amount of abuse, but at the cost of Tor becoming unusable by people that are not willing or cannot pay for usage or improved quality-of-service.

D. Redesigning Automated Abuse Blocking

Basing automated abuse blocking on ratios of abusive to benign requests instead of absolute values might reduce the instances of higher-bandwidth exit nodes being blocked by abuse-based filtering. However, this would allow abusive users to insert benign chaff requests to evade automated abuse filtering based on ratios instead of fixed limits.

Another idea is to never completely block requests and instead display CAPTCHAs to low-reputation IP addresses associated with Tor exit nodes. The risk of websites not blocking Tor exit node IP addresses is that CAPTCHAs are an economic deterrent to large-scale abuse that might be insufficient in cases of profit-driven abuse, such as spam [15]. This highlights the challenges of websites that block Tor exit node IP addresses in self-defense based on automated abuse filtering systems.

VII. Conclusion and Future work

Anonymous communication on the Internet is a critical resource for people whose access to the Internet is restricted by governments. However, the utility of anonymity networks is threatened by services on the Internet that block or degrade requests from anonymous users. In this paper, we measured that at least 1.3 million addresses in the IPv4 address space, and $\approx$ 3.67% of the Alexa top 1,000 websites, either block or offer degraded service to Tor users. Our study provides a first step in illuminating the scale of the problem and identifying centralized mechanisms that impact the usability of many sites for users of anonymity networks. While many websites block Tor to reduce abuse, doing so inadvertently impacts users from censored countries who do not have other ways to access censored Internet content. In the future, we plan to perform large-scale studies of the HTTP layer to discover finer-grained discrimination, such as websites offering restricted services to anonymous users. We will also investigate more effective technical and policy-level solutions to mitigate the second-class treatment of anonymous users. **Data availability:** All relevant data, code, and auxiliary information are available from the University College London database, under the DOI: http://dx.doi.org/10.5522/00/5

Acknowledgements

We thank the sysadmins at the University of California, Berkeley, University of Cambridge and University of Michigan for their help and support throughout this project; and the operators of the Tor exit nodes used in this study, Moritz Bartl and Juris Vetra from project Torservers.net, for their assistance with data collection. We thank Philipp Winter for help in running Exitmap, Arturo Filastò for advice in working with OONI measurements, and George Danezis for advice on Bayesian analysis method to find confidence intervals for our results to mitigate the effect of network loss. We thank Michael Tschantz, Zakir Durumeric, Georg Koppen and Bjoern A. Zeeb for comments and help on various aspects of this work. Likewise, we also appreciate the valuable feedback we received from our shepherd, Lujo Bauer, and the anonymous reviewers.

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/L003406/1]; the US National Science Foundation (grants CNS-1223717, CNS-1237265, CNS-1619620, and CNS-1518918); Intel through the ISTC for Secure Computing and the Open Technology Fund through the Freedom2Connect Foundation and from the US Department of State, Bureau of Democracy, Human Rights and Labor. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] D. Austin. TOR Node List. https://www.dan.me.uk/tornodes.

[2] Citizen Lab. URL testing lists, July 2015. https://github.com/citizenlab/test-lists.

[3] CloudFlare. Does CloudFlare block Tor?, May 2015. https://support.cloudflare.com/hc/en-us/articles/203306930.

[4] R. Dingledine. A call to arms: Helping Internet services accept anonymous users, Aug. 2014. https://blog.torproject.org/blog/call-arms-helping-internet-services-accept-anonymous-users.

[5] R. Dingledine and J. Appelbaum. How governments have tried to block Tor. Chaos Communication Congress, Dec. 2012. https://svn.torproject.org/svn/projects/presentations/slides-28c3.pdf.

[6] R. Dingledine and N. Mathewson. Design of a blocking-resistant anonymity system. Technical Report 2006-11-001, The Tor Project, Nov. 2006. https://research.torproject.org/techreports/blocking-2006-11.pdf.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, Aug. 2004. https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf.

[8] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *USENIX Security Symposium*, pages 605–620, 2013. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric.

[9] R. Henry and I. Goldberg. Formalizing Anonymous Blacklisting Systems. In *IEEE Symposium on Security and Privacy*, pages 81–95, May 2011. http://www.ieee-security.org/TC/SP2011/PAPERS/2011/paper006.pdf.

[10] R. Henry, K. Henry, and I. Goldberg. Making a nymbler Nymble using VERBS. In *Privacy Enhancing Technologies*, volume 6205, pages 111–129. July 2010. http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-05.pdf.

[11] R. Jansen, N. Hopper, and Y. Kim. Recruiting new Tor relays with BRAIDS. In *ACM Conference on Computer and Communications Security*, pages 319–328, 2010. http://www-users.cs.umn.edu/~hopper/braids_ccs.pdf.

[12] D. Johnson. Stem. https://stem.torproject.org/.

[13] B. Jones, T.-W. Lee, N. Feamster, and P. Gill. Automated detection and fingerprinting of censorship block pages. In *Internet Measurement Conference*. ACM, 2014. http://conferences2.sigcomm.org/imc/2014/papers/p299.pdf.

[14] N. Mathewson. #1751: Project: Make it harder to use exits as one-hop proxies, July 2010. https://bugs.torproject.org/1751.

[15] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: CAPTCHAs: Understanding CAPTCHA-solving services in an economic context. In *USENIX Security Symposium*, pages 28–28, 2010. https://www.usenix.org/legacy/events/sec10/tech/full_papers/Motoyama.pdf.

[16] OONI. HTTP Requests Test, July 2013. https://github.com/TheTorProject/ooni-spec/blob/6a3c38f2dc/test-specs/ts-003-http-requests.md.

[17] Open Observatory of Network Interference (OONI). https://ooni.torproject.org/.

[18] R. Padmanabhan, P. Owen, A. Schulman, and N. Spring. Timeouts: Beware surprisingly high delay. In *Internet Measurement Conference*, pages 303–316. ACM, 2015. http://conferences2.sigcomm.org/imc/2015/papers/p303.pdf.

[19] sectoor GmbH. TOR DNSBL - blacklist for Tor servers. http://www.sectoor.de/tor.php.

[20] The Tor Project. Onionoo—a Tor network status protocol. https://onionoo.torproject.org/.

[21] The Tor Project. The public TorDNSEL service. https://www.torproject.org/projects/tordnsel.

[22] The Tor Project. Direct users by country, Aug. 2015. https://metrics.torproject.org/userstats-relay-country.html?graph=userstats-relay-country&end=2015-09-01.

[23] The Tor Project. Tor directory protocol, version 3, Aug. 2015. https://spec.torproject.org/dir-spec.

[24] The Tor Project. Tor pluggable transport specification, 2015. https://spec.torproject.org/pt-spec.

[25] Tor: 'The king of high-secure, low-latency anonymity'. Media leak, Oct. 2013. http://www.theguardian.com/world/interactive/2013/oct/04/tor-high-secure-internet-anonymity.

[26] P. Tsang, A. Kapadia, C. Cornelius, and S. Smith. Nymble: Blocking misbehaving users in anonymizing networks. *IEEE Transactions on Dependable and Secure Computing*, 8(2):256–269, Mar. 2011.

[27] Various authors. List of services blocking Tor. https://trac.torproject.org/projects/tor/wiki/org/doc/ListOfServicesBlockingTor.

[28] Wikipedia. Advice to users using Tor, Aug. 2015. https://en.wikipedia.org/w/index.php?title=WP:TOR&oldid=670864289.

[29] P. Winter. exitmap, 2014. https://github.com/NullHypothesis/exitmap.

[30] P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. Weippl. Spoiled onions: Exposing malicious Tor exit relays. In *Privacy Enhancing Technologies*. Springer, 2014. http://www.cs.kau.se/philwint/spoiled_onions/pets2014.pdf.