

October 2022 Internet Freedom ECP Update

Roger Dingledine

October 31 2022

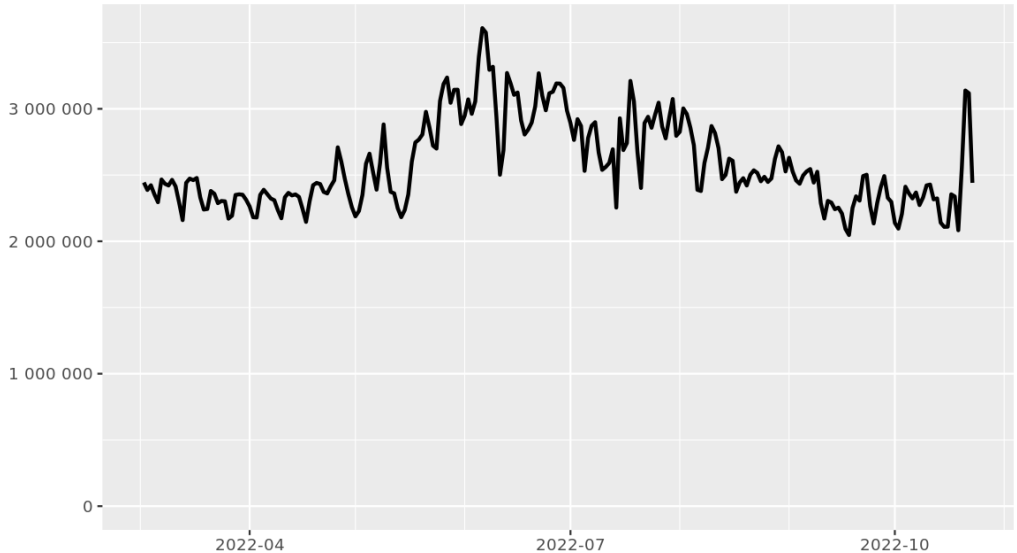
The Tor Project



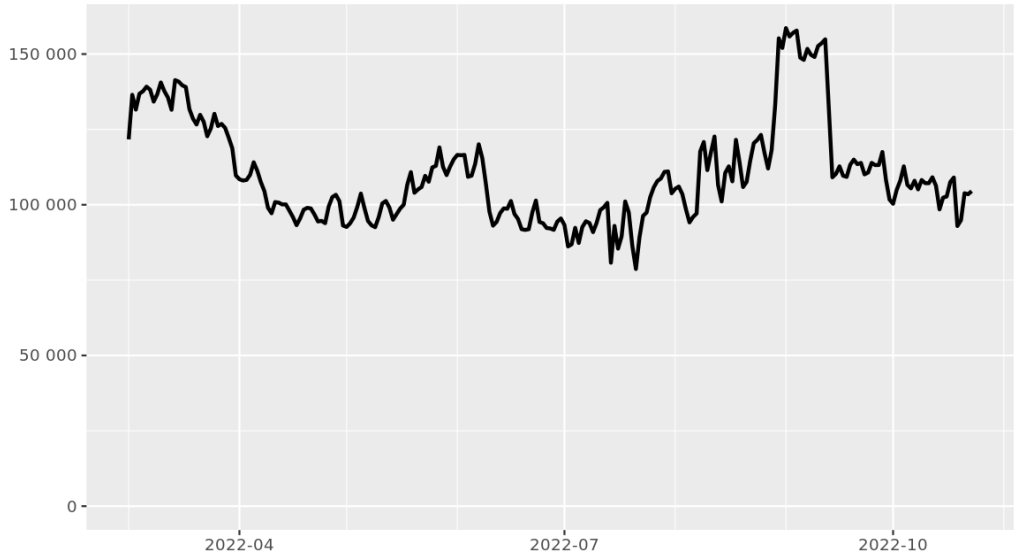
(1) Russia follow-ups since March

- (2) the new Iran chapter
- (3) obfs4 flaws and fixes
- (4) dynamic bridge reachability experiment
- (5) building blocks toward better bridge distribution

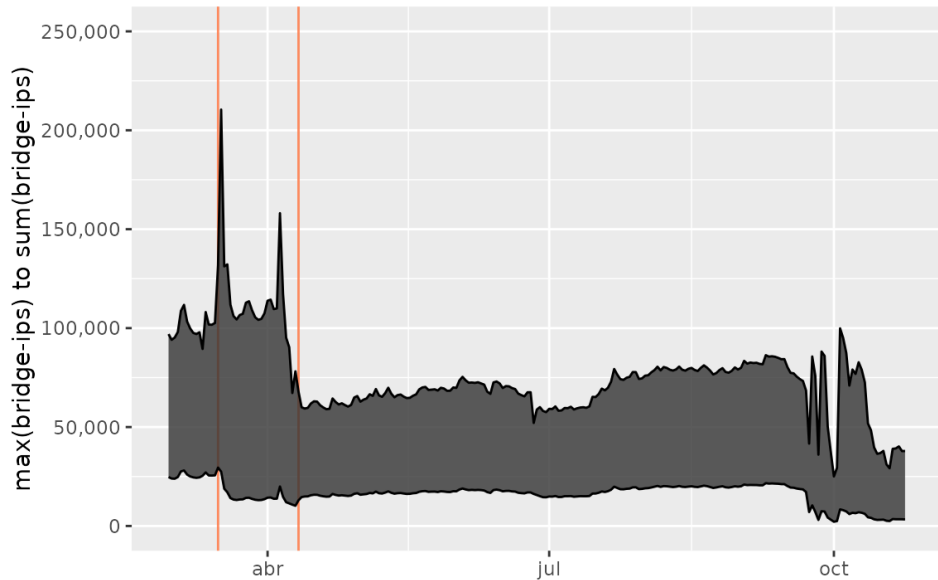
Directly connecting users



Directly connecting users from Russia

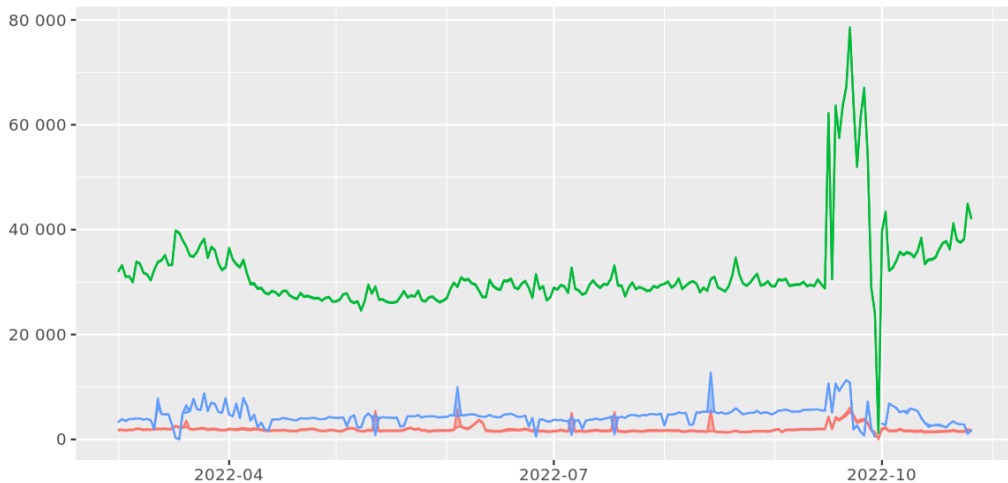


Snowflake unique daily users estimate (Russia only)

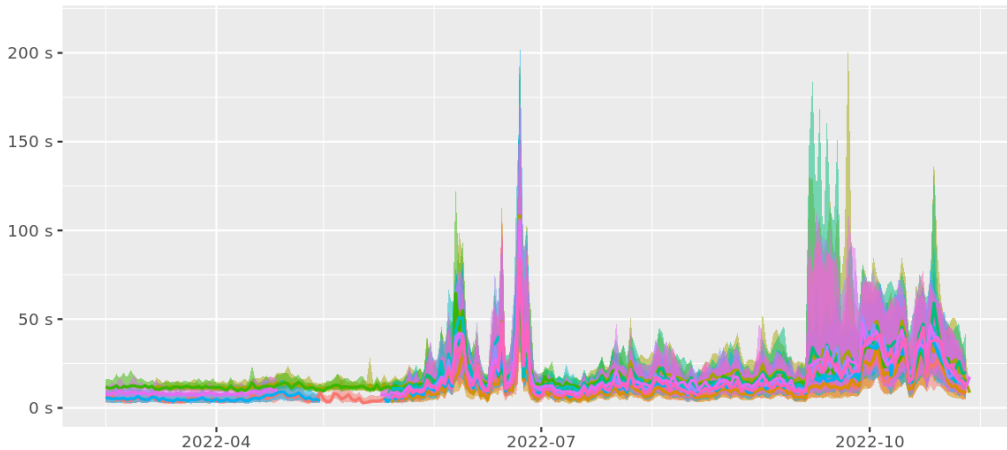


Bridge users by transport from Russia

Top-3 transports ■ <OR> ■ obfs4 ■ snowflake



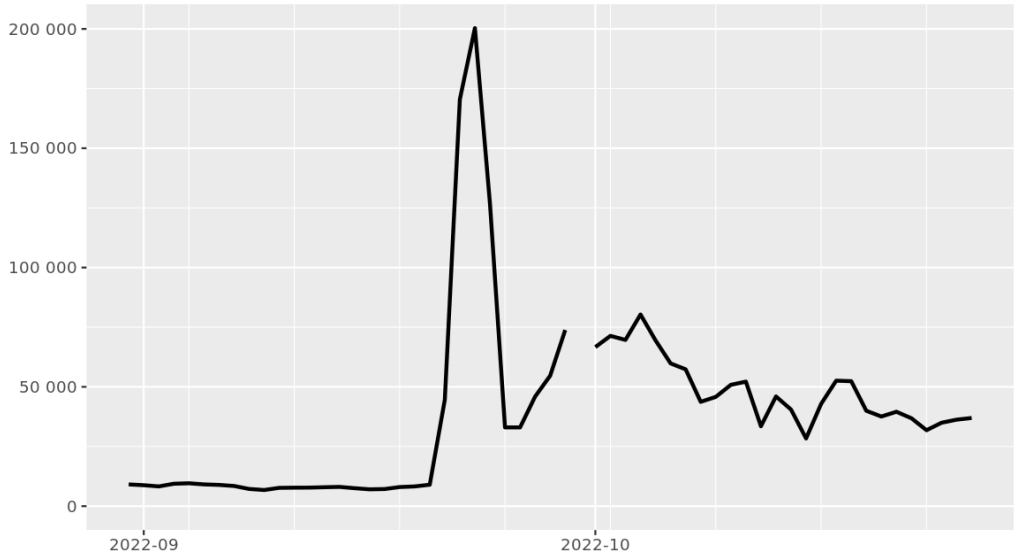
Time to complete 5 MiB request to public server



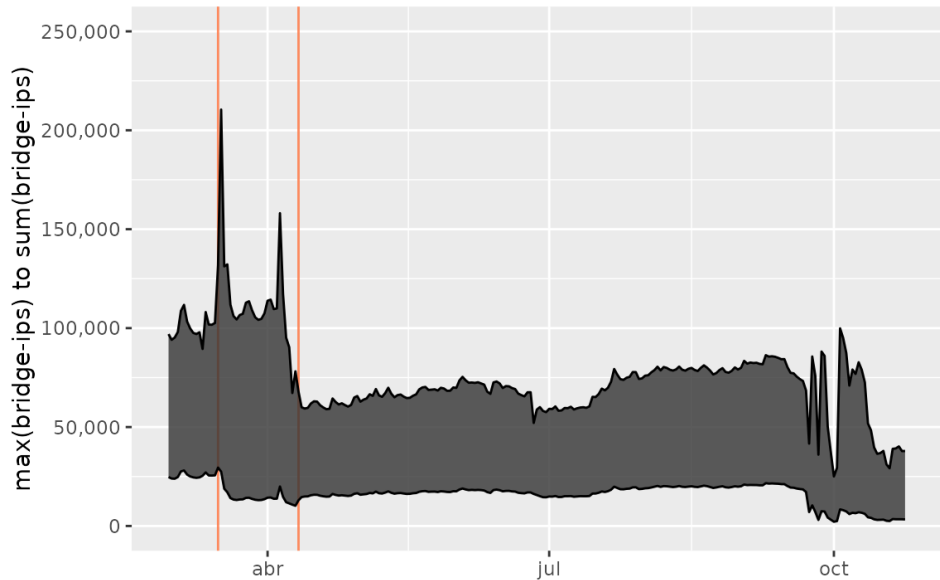
Outline

- (1) Russia follow-ups since March
- (2) the new Iran chapter**
- (3) obfs4 flaws and fixes
- (4) dynamic bridge reachability experiment
- (5) building blocks toward better bridge distribution

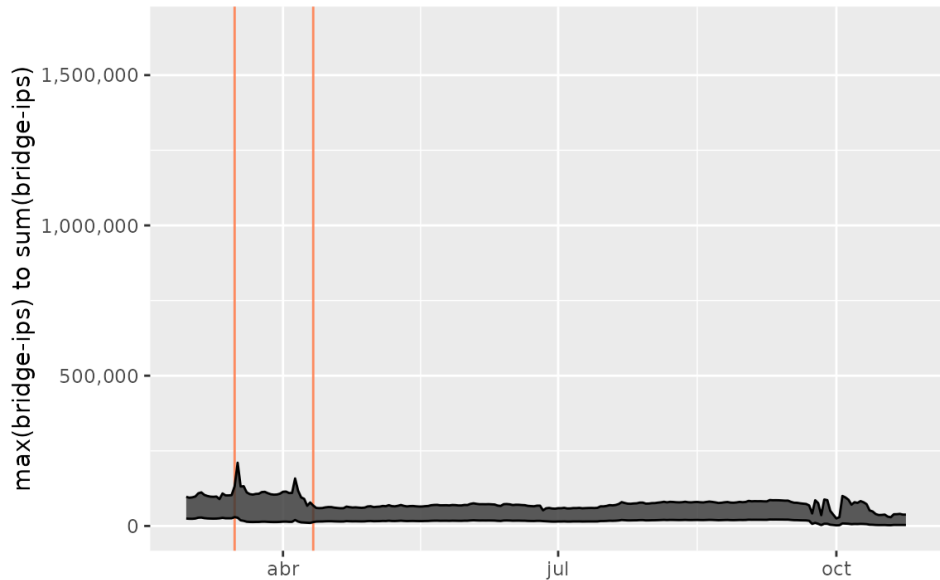
Directly connecting users from Iran



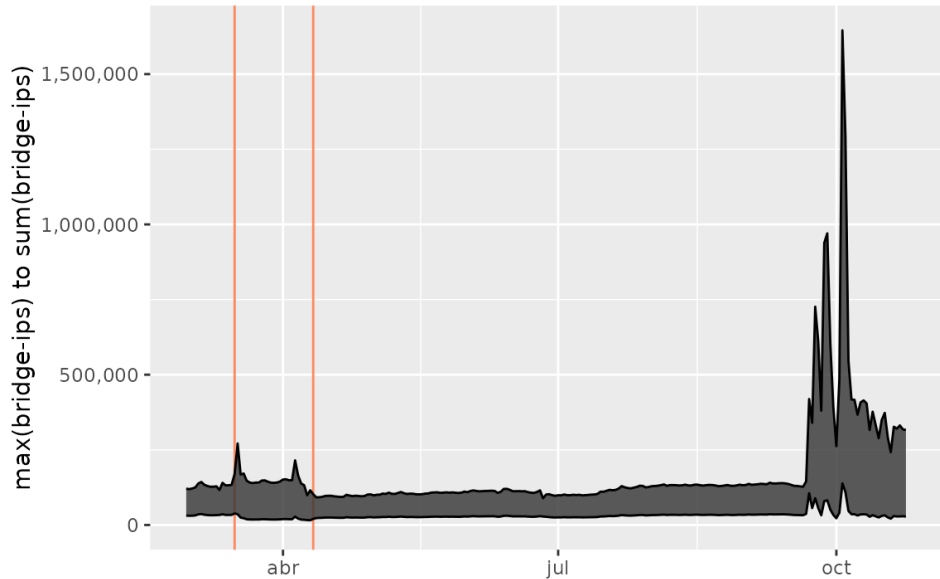
Snowflake unique daily users estimate (Russia only)

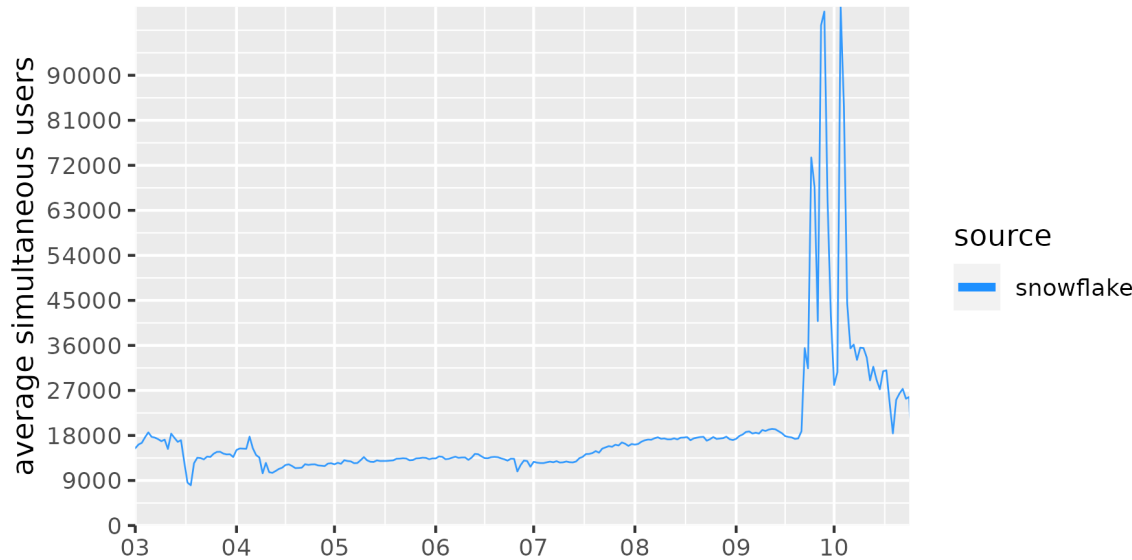


Snowflake unique daily users estimate (Russia only)



Snowflake unique daily users estimate







باران !! توییت پین !!

@radfembaran

...

ایرانی‌های خارج نشین، می‌خواین کمک کنین؟
Snowflake رو نصب کنید، مثل یه پل
میمونه. من الان هیچکدوم از VPN هام کار
نمیکنن، Orbot رو نصب کردم و کانکت شدم
و به لطف بچه‌های خارج که Snowflake
نصب کردن توییتر، تلگرام و واتسپ رو باز
کردم. #مهسا_امینی #Oplran
لینکش:

snowflake.torproject.org

Translate Tweet

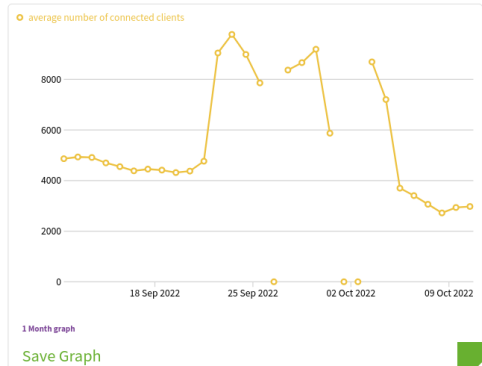
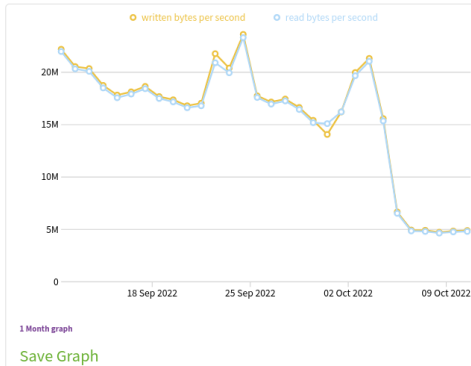


Google Play Ranking: The Top Free Overall in Iran

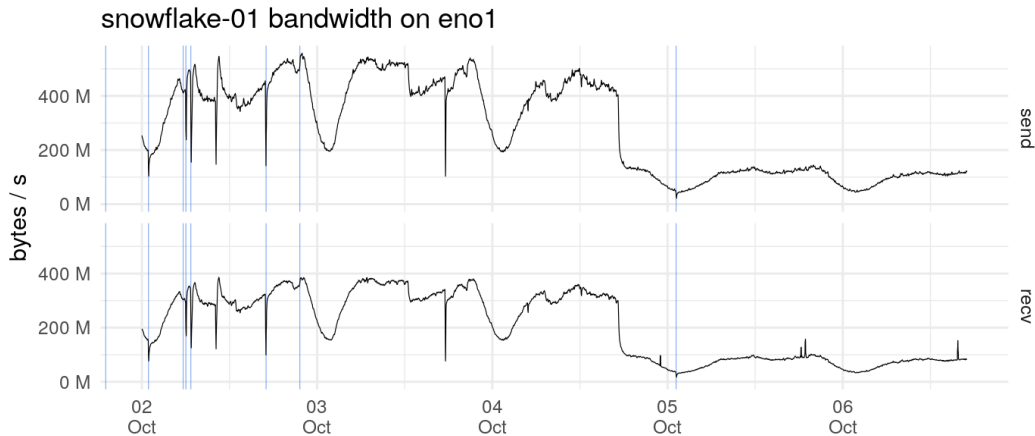
Track the rankings of your Android apps for free with AppBrain. The rankings are refreshed daily from Google Play.

CSV		Top Free		Iran		Overall	
Rank		App	Category	Rating	Installs		
1	=	 Tor Browser by The Tor Project	Communication	★★★★★ 4.4	10 M+		
2	▲1	 Orbot: Tor for Android by The Tor Project	Communication	★★★★★ 4.2	10 M+		
3	▼1	 HulaVPN - Fast Secure VPN by Hula Link	Tools	★★★★★ 4.2	1 M+		
4	=	 VPN Fast - Secure VPN Proxy by Phone Master Lab	Tools	★★★★★ 4.7	10 M+		
5	=	 Turbo VPN - Secure VPN Proxy by Innovative Connecting	Tools	★★★★★ 4.7	100 M+		
6	=	 Ultrasurf - Fast Unlimited VPN by Ultrareach	Tools	★★★★★ 4.8	10 M+		

Snowflake activity on 1 of 12 bridges



Snowflake activity reported by back-end server



S



Open Sudden reduction in snowflake-01 bridge bandwidth, 2022-10-04 17:15



D

David Fifield @dcf · 3 weeks ago

Author

Owner



I think I found the cause. The snowflake-server process was running out of file descriptors. I am not sure why that should have caused such a drastic reduction in throughput, but the log messages around the time of the drop are clear:

```
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 handleConn: failed to connect to ORPort: dial tcp [scrubbed]->[scrubbed]:
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 handleConn: failed to connect to ORPort: error reading TOR_PT_AUTH_COOKIE_
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 handleConn: failed to connect to ORPort: error reading TOR_PT_AUTH_COOKIE_
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 http: Accept error: accept tcp [scrubbed]: accept4: too many open files; r
2022/10/04 17:13:59 handleConn: failed to connect to ORPort: error reading TOR_PT_AUTH_COOKIE_
2022/10/04 17:13:59 handleConn: failed to connect to ORPort: error reading TOR_PT_AUTH_COOKIE_
2022/10/04 17:14:01 http: TLS handshake error from [scrubbed]: EOF
2022/10/04 17:14:03 http: TLS handshake error from [scrubbed]: read tcp [scrubbed]->[scrubbed]
```



1



No mil...



None



None



Tor

8

3

99+

S

Open

Sudden reduction in snowflake-01 bridge bandwidth, 2022-10-04 17:15

country	2022-10-04 17:01:14	2022-10-05 17:01:14	increase/decrease
ir	52408	15680	-70%
??	12232	8208	-33%
us	10560	4480	-58%
ru	5232	5056	-3%
cn	656	600	-9%
mu	648	384	-41%
tn	472	200	-58%
de	408	624	+53%
ma	256	112	-56%
gb	208	272	+31%
eg	200	168	-16%

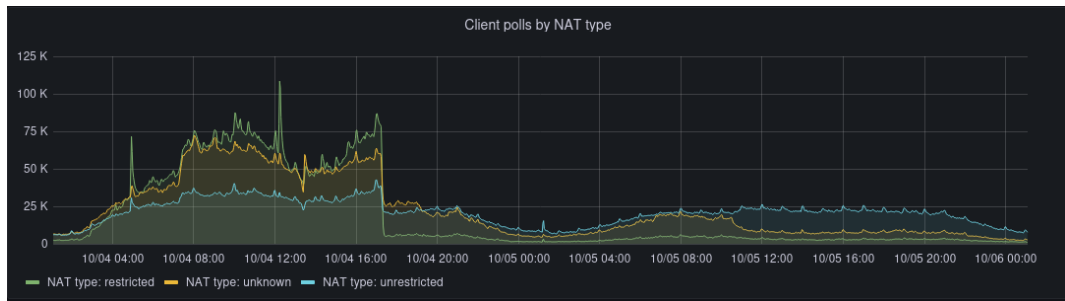
<<

1

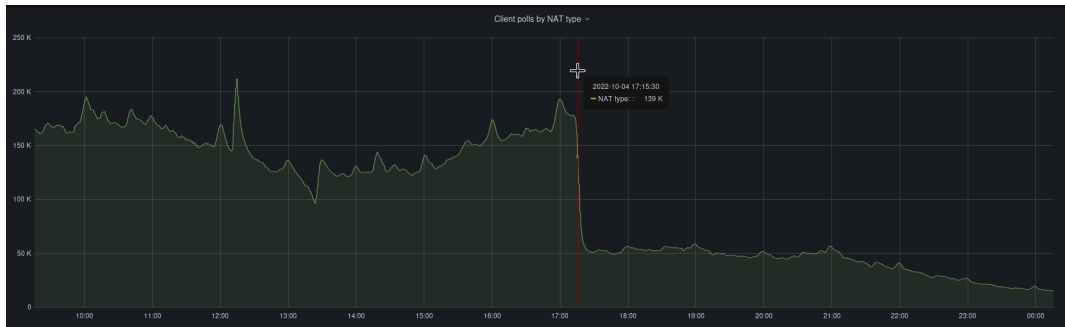
No mil...

None

None



number of clients contacting the Snowflake broker



S

Open

Sudden reduction in snowflake-01 bridge bandwidth, 2022-10-04 17:15

D

D

David Fifield @dcf · 1 week ago

Author

Owner

😊

💬

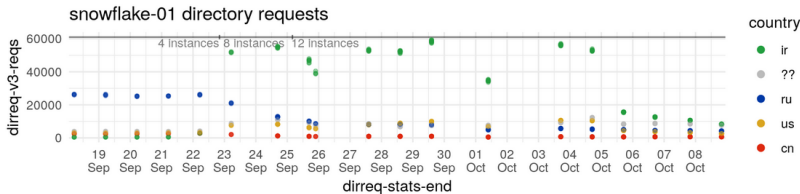
⋮

However the metrics *do* show that the cause is not a country-wide block of broker rendezvous.

I have a new working hypothesis: the sudden decline *is* caused by a partial block of the broker in Iran. As for why countries other than Iran are seemingly affected, my best guess is that they are geolocation errors: IP addresses in Iran wrongly being attributed to other countries (chiefly `us` and `??`).

Below are the top 20 countries according to `dirreq-v3-reqs` on 2022-10-04, and how much they changed 24 hours later.

Here's that information with more context.



📎 [snowflake-01-dirreqs-20221014.zip](#)

«

✓

👤

📄

1

🕒

No mil...

📅

None

🕒

None

👁

🔒

»



Unexplained drop in Snowflake client polls and bandwidth, testers wanted #131

wkrp opened this issue 21 days ago · 68 comments



SaSyda commented 21 days ago



ok, the thing is i've been using tor in the first days of outage, but for certain reasons, been using other solution for the days since. As i saw ur post last night, decided to help out and started things up, first thing i realised, i could connect from my pc but not my mac laptop, but on a second thought i'm using tor browser on windows and orbot on mac. so i downloaded tor browser on mac os and wow! it's connecting with superspeeds, so, my conclusion, it's not actually a problem of ur systems, but the fact that people are mainly using orbot, and whatever thing they've done to stop us, is related to that app and the streams its using and going through. actually i'm more confident about my assumption, cuz i've suggested orbot to many people as it runs on mobile devices and clearly most users in iran use a phone to get to websites like instagram or certain messengers. And computer celebs over twitter and other places, been suggesting that app as well.

And i assume, as I can connect to the free internet, my log is no use to u, but still, tell me if u need me to send it as i could successfully go through ur tutorial.

and in my next experiments, i'll be using cellular connection, as it has been way heavier censored and the so called "national internet" which is an actual intranet with measures to limit connection to foreign servers and computers, is mainly implemented on OTG internet connections.

Reachability testing from inside Iran

	Works?
Tor Browser Linux	yes
Tor Browser Android	yes
Orbot	yes
Orbot	no

Tor

≡

🔍

+

▼

📺 8

🔗 3 ▼

📧 99+

🔍 ▼

👤 ▼

S

Open Sudden reduction in snowflake-01 bridge bandwidth, 2022-10-04 17:15

📁

📄

📄

🔗

🚀

🛡️

🕒

🏠

📺

📊

📄

✂️

»

D

David Fifield @dcf · 1 week ago

Author

Owner

😊

⋮

Thank you, that's very helpful. The fingerprint of your orbot.pcap is indeed identical to [my one](#). In `tlsfingerprint.io` terms, it is `adfe55afa6f23950`.

This fingerprint `adfe55afa6f23950` [differs only minorly](#) from `750e3f0f585283bd`, which was observed in <https://github.com/net4people/bbs/issues/139#issuecomment-1280057679> to be produced by a Go program running on Raspberry Pi.

The critical thing in native Go crypto/tls fingerprints [since go1.17](#) is that the order of ciphersuites depends on whether the platform has support for accelerated AES-GCM. See how there are two versions of everything: `cipherSuitesPreferenceOrder` and `cipherSuitesPreferenceOrderNoAES`; `defaultCipherSuitesTLS13` and `defaultCipherSuitesTLS13NoAES`. This explains the two different ciphersuite orderings you observed. The choice of which to use [happens at runtime](#):

```
hasGCMAsmAMD64 = cpu.X86.HasAES && cpu.X86.HasPCLMULQDQ
hasGCMAsmARM64 = cpu.ARM64.HasAES && cpu.ARM64.HasPMULL
hasGCMAsmS390X = cpu.S390X.HasAES && cpu.S390X.HasAESCBC && cpu.S390X.HasAESCTR &&
    (cpu.S390X.HasGHASH || cpu.S390X.HasAESGCM)
hasAESGCMHardwareSupport = runtime.GOARCH == "amd64" && hasGCMAsmAMD64 ||
    runtime.GOARCH == "arm64" && hasGCMAsmARM64 ||
    runtime.GOARCH == "s390x" && hasGCMAsmS390X
```

«

📧

👤

📄 1

🕒

No mil...

📅

None

🕒

None

👁️

🔒

Reachability testing from inside Iran

	Works?	
Tor Browser Linux	yes	⇐ go 1.17, AES-GCM
Tor Browser Android	yes	⇐ go 1.18, AES-GCM
Tor Browser Android	yes	⇐ go 1.18, no AES-GCM
Orbot	yes	⇐ go 1.17, AES-GCM
Orbot	no	⇐ go 1.17, no AES-GCM



build failing godoc reference

uTLS is a fork of "crypto/tls", which provides ClientHello fingerprinting resistance, low-level access to handshake, fake session tickets and some other features. Handshake is still performed by "crypto/tls", this library merely changes ClientHello part of it and provides low-level access.

Golang 1.11+ is required.

If you have any questions, bug reports or contributions, you are welcome to publish those on GitHub. If you want to do so in private, you can contact one of developers personally via sergey.frolov@colorado.edu

Documentation below may not keep up with all the changes and new features at all times, so you are encouraged to use [godoc](#).

Features



Shutdowns, intensified blocking in Iran since 2022-09-21 #125

wkrp opened this issue on Sep 21 · 45 comments



n8fr8 commented 7 days ago



Orbot for Android 16.6.3-BETA-2-tor.0.4.7.10 with utls enabled, now available here:
<https://github.com/guardianproject/orbot/releases/tag/16.6.3-BETA-2-tor.0.4.7.10>

(arm64 direct APK: <https://github.com/guardianproject/orbot/releases/download/16.6.3-BETA-2-tor.0.4.7.10/Orbot-16.6.3-BETA-2-tor.0.4.7.10-fullperm-arm64-v8a-release.apk>)

This uses "utls-imitate=hellochrome_auto" - we will add the other options and ability to customize/select in the next update.

This release also has the ability to get Snowflake logs directly from the log window (enable Prefs->DEBUG log, tap on status messages to open log window, tap on snowflake icon to show snowflake log, then share!)



2



2



mehdifirefox commented 7 days ago





Unexplained drop in Snowflake client polls and bandwidth, testers wanted #131

wkrp opened this issue 21 days ago · 68 comments



SaSyda commented 8 days ago



@wkrp hello man and sorry for my long absence, i dont think u would want to know where i've been these day. Got back here to see your new theory and man u are right, just tested tor browser on android with ur custom snowflake, and it works, not perfectly or fast, but does connect and brings certain restricted websites. actually i'm not that of a professional in that app. but i was able to test it and really, well done. I'll be back with more details soon. keep rocking.



1



 **quartermarsh** commented 8 days ago • edited



@hexrot It may recover after a few hours of idle. Wait and see. A restart of the proxy can help (assuming you're running standalone). The snowflake team is working on it.

<https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transport/snowflake>



Unexplained drop in Snowflake client polls and bandwidth, testers wanted #131

wkrp opened this issue 21 days ago · 68 comments



iRhonin commented 7 days ago



There is now a release available of Orbot that enables uTLS for Snowflake (from [#125 \(comment\)](#)).

You can download APKs here: <https://github.com/guardianproject/orbot/releases/tag/16.6.3-BETA-2-tor.0.4.7.10>

This release makes it possible to see the snowflake-client log. If there's a failure to connect, it will help us figure out what is going wrong. Enable **Settings** → **Debug Log**, then go back to the main screen and **Start**. Tap on a status message to show the tor log, then tap the **snowflake** snowflake icon to view the snowflake-client log.

I can confirm this works in Iran.



2



2



free-the-internet commented 7 days ago



Closed

Reopen issue

⋮

uTLS for broker negotiation

The connection from the snowflake client to the broker server currently uses go's `net/http.DefaultTransport`. That connection is optionally domain fronted, but the TLS handshake is easily fingerprintable as a go handshake, which might trigger additional scrutiny in regimes where that kind of TLS inspection is possible and actionable.

[This paper](#), though a bit out of date now (2019) references meek and even snowflake:

Snowflake is under active development, and its authors were aware of potential TLS fingerprintability issues. Indeed, we find that Snowflake (built from git master branch on April 17, 2018) generates a fingerprint that is close to, but not exactly the same as the default Golang TLS fingerprint. In particular, it diverges by including the NPN and ALPN extensions, and offers a different set of signature algorithms. As a result, this fingerprint is seen in fewer than 0.0008% of connections, making it susceptible to blocking.

The author of that paper, Sergey Frolov, maintains <https://tlsfingerprint.io/> which is a list of the most popularly seen TLS fingerprints, and created <https://github.com/refraction-networking/utls> which is a library designed for creating TLS connections with various commonly witnessed TLS fingerprints.

There's a fork of that library, <https://gitlab.com/yawning/utls> which seems to be used in obfs4's `meeklite` implementation, and it looks like `@dcf` implemented a version of that in <https://gitweb.torproject.org/pluggable-transports/meek.git/tree/meek-client/utls.go> which actually implements a `RoundTripper`. It seems as though actually using that library could be a relatively painless way to adopt utls for the broker negotiation.

```
snowflake 192.0.2.3:80
2B280B23E1107BB62ABFC40DDCC8824814F80A72
fingerprint=2B280B23E1107BB62ABFC40DDCC8824814F80A72
url=https://snowflake-
broker.torproject.net.global.prod.fastly.net/
front=cdn.sstatic.net
ice=stun:stun.l.google.com:19302,stun:stun.altar.com.pl:3478,
stun:stun.antisip.com:3478,stun:stun.bluesip.net:3478,stun:st
un.dus.net:3478,stun:stun.epygi.com:3478,stun:stun.sonetel.co
m:3478,stun:stun.sonetel.net:3478,stun:stun.stunprotocol.org:
3478,stun:stun.uls.co.za:3478,stun:stun.voipgate.com:3478,stu
n:stun.voys.nl:3478 utls-imitate=hellorandomizedalpn
```

Outline

- (1) Russia follow-ups since March
- (2) the new Iran chapter
- (3) obfs4 flaws and fixes**
- (4) dynamic bridge reachability experiment
- (5) building blocks toward better bridge distribution

Incorrect (non canonical) representative output for `ScalarBaseMult()` #27

New issue



LoupVallant opened this issue on Feb 24, 2020 · 3 comments



LoupVallant commented on Feb 24, 2020

...

Hi,

I learned of this bug was found by [@tankf33der](#). Contrary to the original paper, the Elligator 2 representative of a public key is not always canonical. That is, this library does not make sure the output stay between 0 and $(p-1)/2$ (that is, $2^{254}-10$).

The mapping is such that for each point on the curve we can map, two representative (`r1` and `r2`) map to that point on the curve, and those points are such that `r1+r2=(2255-19)` . Which means one of them is in $[0, 2^{254}-11]$, and the other is in $[2^{254}-10, 2^{255}-18]$. We are supposed to chose the smaller of the two. If we don't, there is no guarantee the distribution of canonical and non-canonical output will be even. And if it's not, we'll introduce exactly the kind of bias Elligator was meant to eliminate, and basically ruin everything.

The problem can be reproduced with this code:

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

obfs distinguishers

First distinguisher, bit 254 should be randomized because we should be picking the lower of the two representations, but we pick either.

Second distinguisher, <details embargoed until Nov 15>.

Two-ish total bits of distinguisher so not the end of the world.

Partly fixed in obfs4proxy 0.0.12, 2021-12-31. Fixed better in obfs4proxy 0.0.14, 2022-09-04.

obfs distinguishers

We now have a scanner to assess whether a remote obfs4 bridge has the bugs.

Many users not yet updated!

Still working on getting a new obfs4proxy deb to bridge volunteers. We'll do an awareness campaign once the deb is more available.

One lesson is the fancy provably secure modern crypto gadget maybe wasn't better than, say, simple encrypt+xor.

obfs still one of the best options

Analysis continues of GFW entropy-based flow blocking for a few destinations.

Current conclusion is that the collateral damage remains too high to roll out the blocking more widely.

Eric Wustrow has been exploring how much damage these rules would do on his university network.

Consider that GFW could block Digital Ocean entirely and nobody would argue.

Outline

- (1) Russia follow-ups since March
- (2) the new Iran chapter
- (3) obfs4 flaws and fixes
- (4) dynamic bridge reachability experiment**
- (5) building blocks toward better bridge distribution

[From last time] Centralized probe log collection

- <https://gitlab.torproject.org/tpo/anti-censorship/team/-/issues/54>
- First vantage points in China, Turkey, Russia, Ukraine
- Emphasis on more frequent testing because partial (slow) bootstrap results are more common than we first expected
- Upcoming step: link these probe results to the dynamic bridge tools, and then the automation really starts to come together

Relay Search

sr2



sr2

Show 10 entries

Nickname [†]	Advertised Bandwidth	Uptime	Country	IPv4	IPv6	Flags	Add. Flags	ORPort	DirPort	Type
 Sr2SilverWipe	17.86 MiB/s	49d 3h		IPv4	IPv6	   	 		0	Bridge
 Sr2SilverUmbrella	15.1 MiB/s	49d 3h		IPv4	IPv6	   	 		0	Bridge
 Sr2SilverSupply	14.21 MiB/s	49d 3h		IPv4	IPv6	   	 		0	Bridge
 Sr2SilverScreener	12.74 MiB/s	9d 23h		IPv4	IPv6	   	 		0	Bridge
 Sr2IronDaffodil	10.72 MiB/s	66d 20m		IPv4	-	   			0	Bridge
 Sr2IronDinosaur	10.66 MiB/s	66d 20m		IPv4	-	   			0	Bridge
 Sr2IronTelegraphy	10.53 MiB/s	49d 3h		IPv4	-	   			0	Bridge
 Sr2IronBaguette	10.34 MiB/s	42d 5h		IPv4	-	   			0	Bridge
 Sr2IronPound	9.14 MiB/s	42d 5h		IPv4	-	   			0	Bridge
 Sr2PlasticDrone	9.03 MiB/s	1d 16h		IPv4	IPv6	   	 		0	Bridge
Total	384.6 MiB/s									

Showing 1 to 10 of 66 entries



B

main

bridgestatus /

+

Find file

Web IDE

Download

Clone

update

ShellBot-BridgeAccessibilityAutomation authored 7 hours ago

63909375

README

Add LICENSE

Add CHANGELOG

Add CONTRIBUTING

Set up CI/CD

Name	Last commit	Last update
README.md	Initial commit	5 months ago
recentResult_cn	update	7 hours ago
recentResult_ie_test	update	1 month ago
recentResult_russia	update	3 days ago
recentResult_turkey	update	1 month ago

README.md

Tor

≡

🔍

+

▼

📄 8

🔗 3 ▼

📧 99+

❓ ▼

👤 ▼

B

🏠

📄

📄

🔗

🚀

🛡️

🔄

📁

🏠

🖨️

📊

📄

✂️

»

The Tor Project > ... > Connectivity Measurement > BridgeStatus > Repository

main ▼

bridgestatus / recentResult_russia

Find file

Blame

History

Permalink

👤

update

ShellBot-BridgeAccessibilityAutomation authored 3 days ago

baleafa7

📄

📄 recentResult_russia

📄 17.81 KiB

Open in Web IDE ▼

Replace

Delete

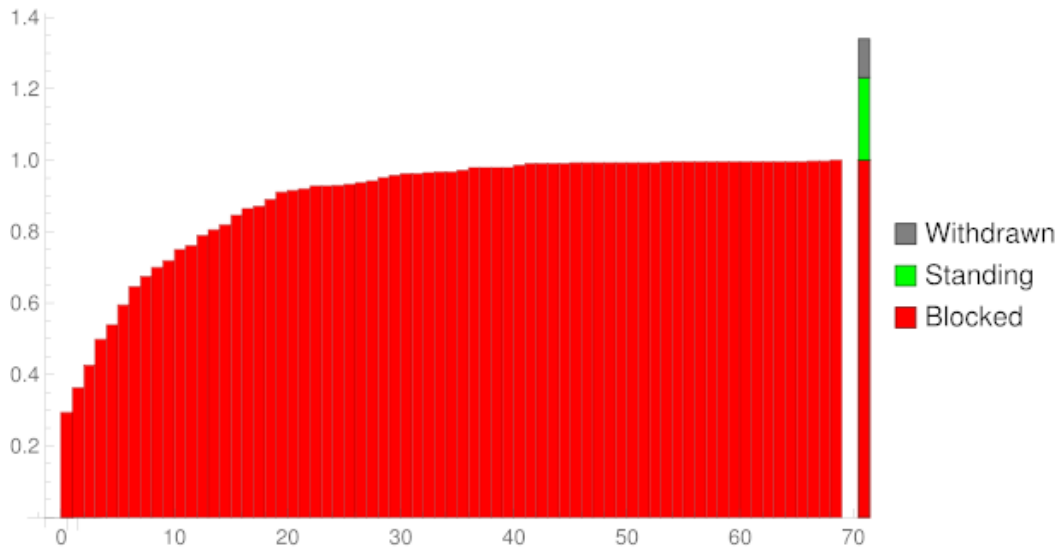
📄

📄

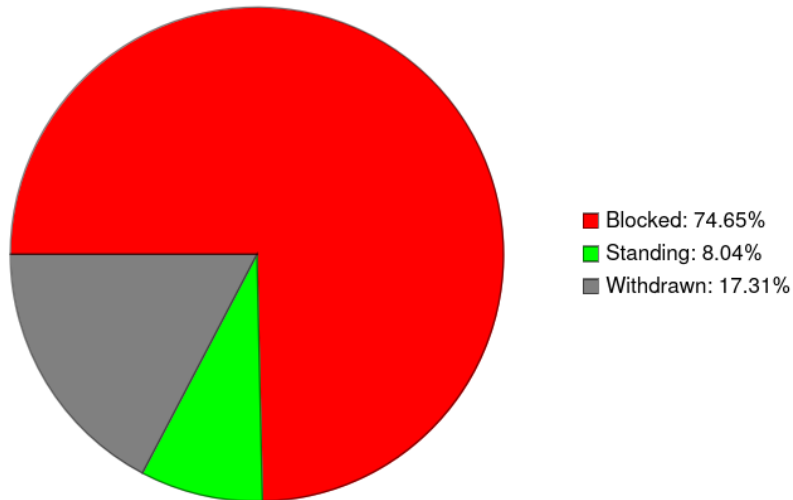
📄

1	047AB264EA0962A080ABAA8E789DAE3D980340A8	100	2022-10-26T00:38:37Z	1666744717	f2e
2	09CC45EED639F2FD58C14BF6D99A726B2F5098C8	100	2022-10-26T00:38:37Z	1666744717	Sec
3	14A4212B5DCC10F7C0F7124EB8FFB326F3B1C387	100	2022-10-26T00:38:37Z	1666744717	5e7
4	163FA2C15B2FC5FBBB4F17971F884EA81658E721	100	2022-10-26T00:38:37Z	1666744717	436
5	1ABE3437D71324F269998DF7FDFFF440D9425DAF	100	2022-10-26T00:38:37Z	1666744717	882
6	1F9F3A67CCDCEE2063B3E2BEEB136FE10B08F088	100	2022-10-26T00:38:37Z	1666744717	5b0
7	22E2530788E68B680CDCE861B65EC77389681BF3	77	2022-10-26T00:38:37Z	1666744717	e9a
8	28E82C0D7A0E0E956CECF604CDE1450852F0C1D4	100	2022-10-26T00:38:37Z	1666744717	025
9	2A44BB7DD2356E99833A528686AB799EDBEC7E50	100	2022-10-26T00:38:37Z	1666744717	b5b
10	2B920A09B6E0B4BF0924310753EA141C0B610388	100	2022-10-26T00:38:37Z	1666744717	ab9
11	2CE8DEB3C10368637CE17452CF70D94AEB8857AD	100	2022-10-26T00:38:37Z	1666744717	baa
12	2D650C89A12848DB316E8F9D603094EFE6916AB5	100	2022-10-26T00:38:37Z	1666744717	b2b
13	3228809791A855063CFFF831A5E1E7429390A434	100	2022-10-26T00:38:37Z	1666744717	2e3
14	32D7D81186E7C97176C5266BD01213A76CBE81B4	100	2022-10-26T00:38:37Z	1666744717	878

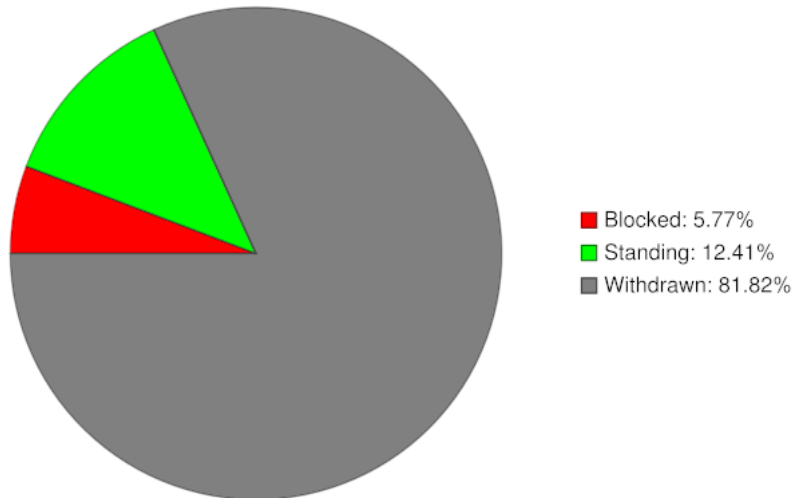
CDF of time-until-block for China bridges



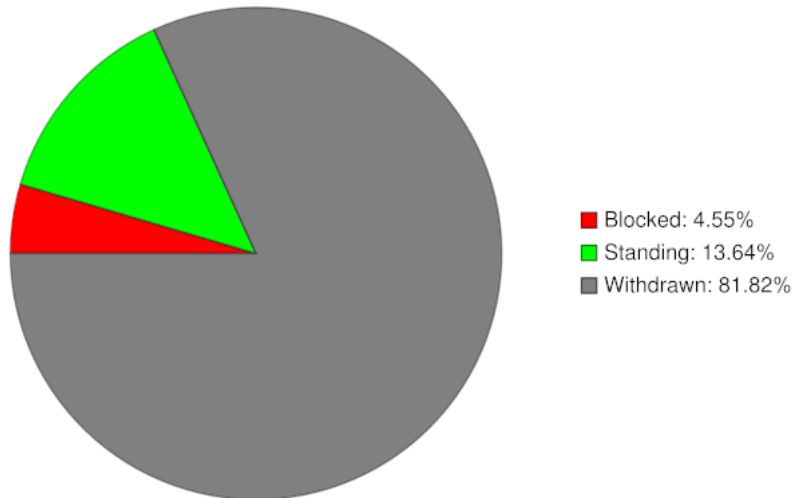
End state for each bridge address in China



End state for each bridge address in Russia

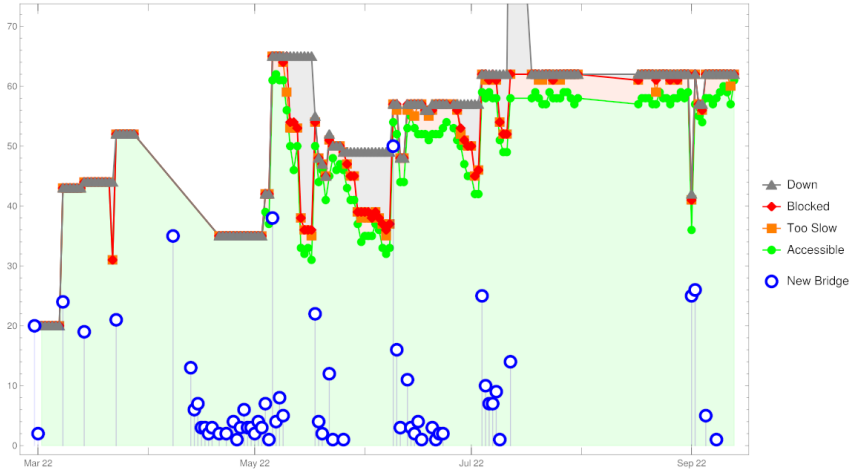


End state for each bridge address in Turkey

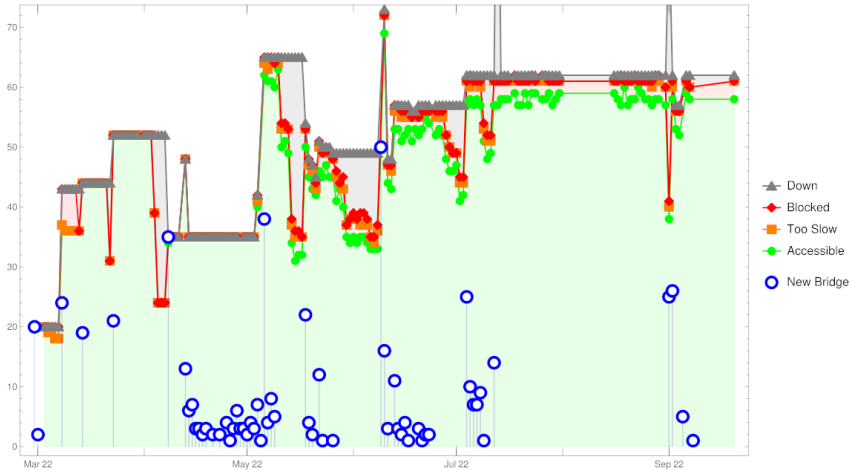


```
Terminal - screen
File Edit View Terminal Tabs Help
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,100,1652122548,turkey,2022-05-09T18:55:
48Z,,NULL,2022-05-09,S96,Accessible,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,100,1652123404,vinnica,2022-05-09T19:10
:04Z,,NULL,2022-05-09,S96,Accessible,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,100,1652133465,russia,2022-05-09T21:57:
45Z,,NULL,2022-05-09,S96,Accessible,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,100,1652136118,cn,2022-05-09T22:41:58Z,
,NULL,2022-05-09,S96,Accessible,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,2,1652208028,turkey,2022-05-10T18:40:28
Z,,NULL,2022-05-10,S96,Blocked,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,100,1652212010,vinnica,2022-05-10T19:46
:50Z,,NULL,2022-05-10,S96,Accessible,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,100,1652219607,russia,2022-05-10T21:53:
27Z,,NULL,2022-05-10,S96,Accessible,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,100,1652224275,cn,2022-05-10T23:11:15Z,
,NULL,2022-05-10,S96,Accessible,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,2,1652294867,turkey,2022-05-11T18:47:47
Z,,NULL,2022-05-11,S96,Down,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,2,1652296057,vinnica,2022-05-11T19:07:3
7Z,,NULL,2022-05-11,S96,Down,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,2,1652305160,russia,2022-05-11T21:39:20
Z,,NULL,2022-05-11,S96,Down,1651864255
BBCBBB3A8E87AFDA919FBD465BE13C15E22781B7,2,1652309311,cn,2022-05-11T22:48:31Z,,N
:
█
```

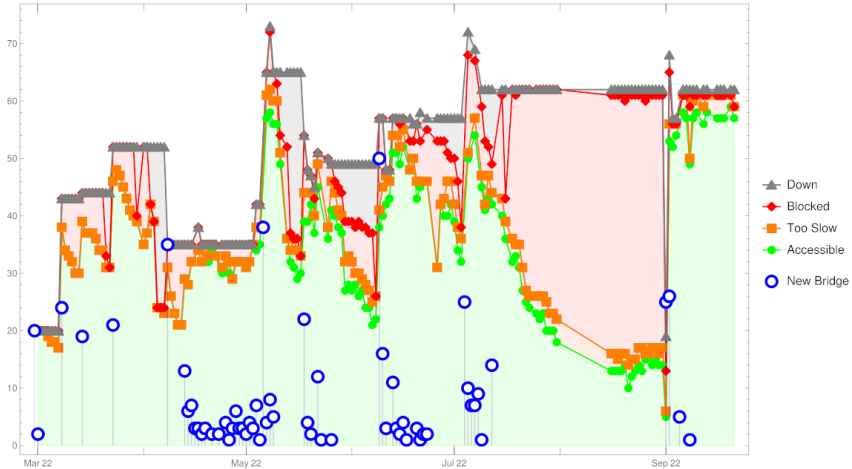
Status of Turkey bridges over time



Status of Russia bridges over time



Status of China bridges over time



Issue created 4 days ago by **Roger Dingleline**

Close issue

Keep irl's dynamic bridges around for a few days after rotation

Right now as I understand it, the moment we get a "blocked!" result out of bridgestatus in any country, we tell irl's dynamic bridge farm that it's time to rotate that bridge.

This is good from an overall reachability perspective, because any waiting, if we're right and the bridge got blocked, is bad for users.

But also, by spinning the old bridge down right then, we lose out on a lot of potential understanding:

- How often are these 'blocked' events false positives? That is, how often do we rotate away from a bridge when actually it was just a brief connectivity issue or a slow bootstrap? These are cases that we are labeling "blocked" in our data yet we don't know how much we're overcounting the blocked cases.
- By rotating the bridge immediately, we make it hard to figure out what happened in other countries. We have what look like a bunch of false positives in Turkey and Russia in the [tpo/anti-censorship/team#92](#) analysis, where we get one connection failure from Turkey but it works from other countries, and then the next day the bridge is down, because we intentionally took it down after that first 'blocked' case.

This approach of taking the bridge down after the first failure prevents us from using a more nuanced definition of blocked such as "down three days in a row in this country while still reachable from some other countries during that time".

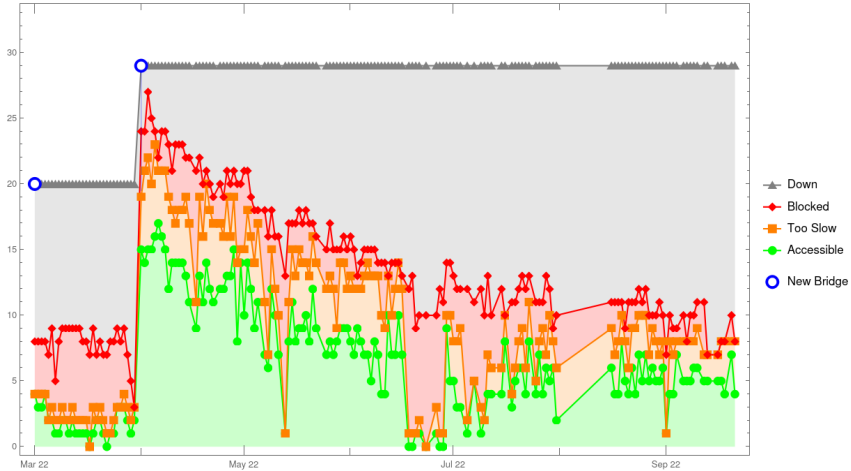
So my proposed fix is to change irl's dynamic bridge framework to schedule a spin-down of the old bridge for N (e.g. N=3) days in the

No mil...

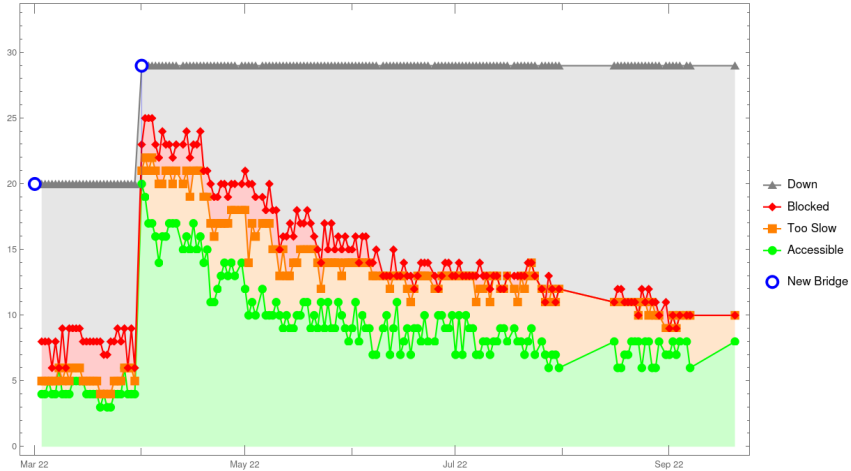
None

None

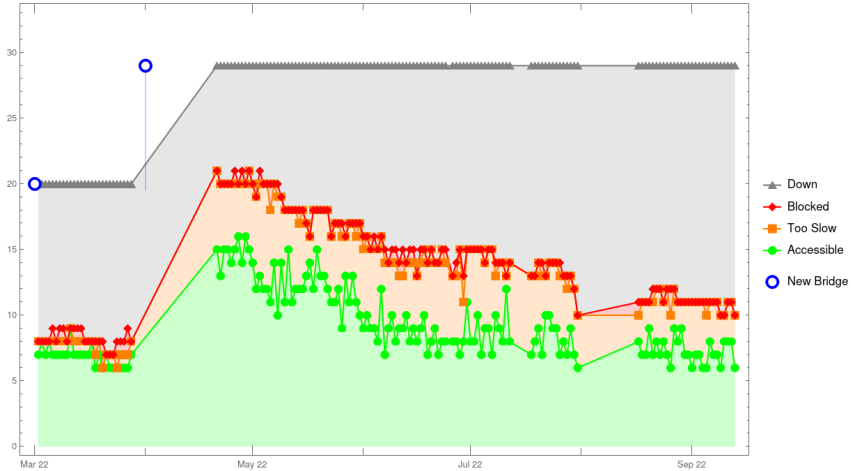
Community bridge reachability, China



Community bridge reachability, Russia



Community bridge reachability, Turkey



Analysis by distribution strategy

44 dynamic bridges given out both via moat (in-browser captcha) and via telegram+new. 10 given out only via telegram+new.

Tentative results:

- moat mostly blocked in China
- telegram+new mostly working in China
- both classes working in Russia and Turkey

Follow-up question: what about user load on each class of bridge?

Outline

- (1) Russia follow-ups since March
- (2) obfs4 flaws and fixes
- (3) the new Iran chapter
- (4) dynamic bridge reachability experiment
- (5) building blocks toward better bridge distribution**

Building block 1

#1: a DPI-resistant point-to-point channel.

That's obfs4, but it can also be v2ray's vmess or any similar project.

Building block 2

#2: a Sybil-resistant sign-up mechanism.

moat/https/gmail buckets are a good start, but Telegram distributor (with twist) is a better start.

Need to identify resources where (a) individual users probably have a few but it's tough for the censor to have many, and (b) it's easy to verify in an automated way.

Think “Facebook accounts” or “friends on Twitter”

Building block 3

#3: a way to establish ground-truth about a suspected-blocked bridge.

Building block 4

#4: a large pool of diverse bridge addresses.

(1) the thousands of volunteers who run their own bridges.

(2) automated framework for easily spinning up and down bridge images on popular cloud providers.

(3) Conjure pools?

Building block 5

#5: a robust signaling channel.

Need some bidirectional channel that will reliably work, i.e. that censors won't want to block even when they know the details of how it works.

It's fine if it's low-bandwidth and not-great-latency.

Domain fronting through Fastly; proxy through Google AMP cache; tunnel through DoH or email.

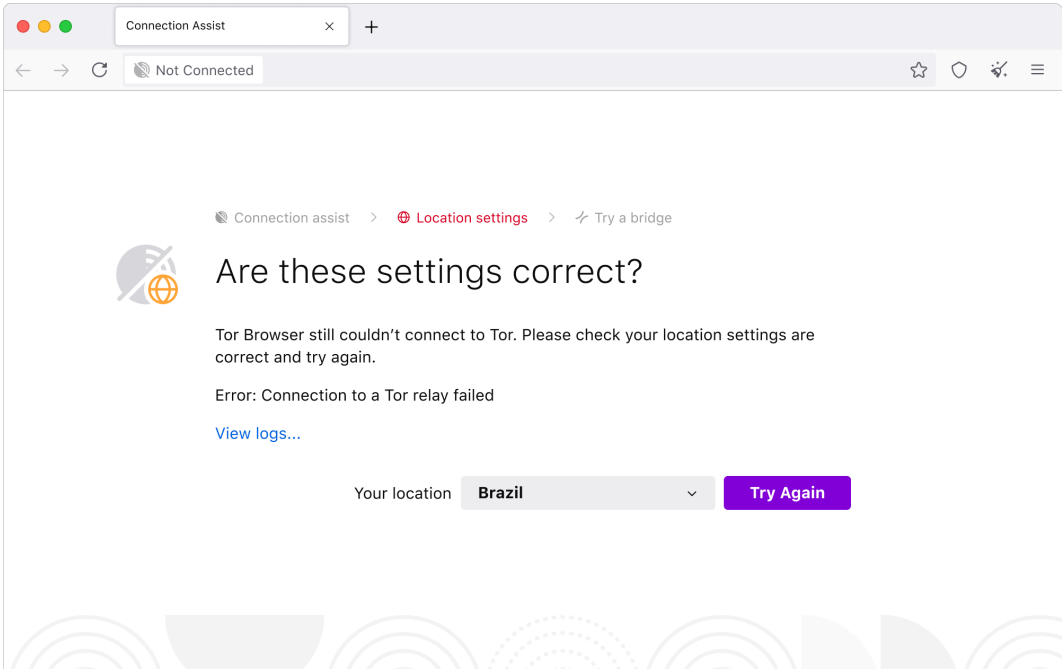
Milestone 1: Tor Browser automation

Tor Browser uses the signaling channel to learn the recommendations for your country.

It fetches a json file with an ordered list of which pluggable transports and which bridge distribution strategies are expected to work best in each country.

Then Tor Browser walks the user through trying them.

```
{
  "by": {
    "settings": [
      {"bridges": {"type": "obfs4", "source": "builtin"}},
      {"bridges": {"type": "vanilla", "source": "bridgedb"}},
      {"bridges": {"type": "obfs4", "source": "bridgedb"}},
      {"bridges": {"type": "snowflake", "source": "builtin"}}
    ]
  },
  "cn": {
    "settings": [
      {"bridges": {"type": "snowflake", "source": "builtin"}}
    ]
  },
  "tm": {
    "settings": [
      {"bridges": {"type": "obfs4", "source": "bridgedb"}},
      {"bridges": {"type": "snowflake", "source": "builtin"}}
    ]
  },
  "ru": {
    "settings": [
      {"bridges": {"type": "snowflake", "source": "builtin"}},
      {"bridges": {"type": "obfs4", "source": "bridgedb"}}
    ]
  }
}
```



[Home](#)[Search](#)[Privacy & Security](#)[Connection](#)[Extensions & Themes](#)[Tor Browser Support](#)

Connection

Tor Browser routes your traffic over the Tor Network, ran by thousands of volunteers around the world. [Learn more](#)

**Internet:** Online[Test Again](#)**Tor Network:** Potentially Blocked

Quickstart

Quickstart connects Tor Browser to the Tor Network automatically when launched, based on your last used connection settings. [Learn more](#)

☐ Always connect automatically

Bridges

Bridges help you access the Tor Network in places where Tor is blocked. Depending on where you are, one bridge may work better than another. [Learn more](#)

Your location

Automatic

[Choose a Bridge For Me...](#)

Add a New Bridge

Choose from one of Tor Browser's built-in bridges

[Select a Built-In Bridge...](#)

Request a bridge from torproject.org

[Request a Bridge...](#)

Enter a bridge address you already know

[Add a Bridge Manually...](#)

Advanced

Configure how Tor Browser connects to the internet

[Settings...](#)

View the Tor logs

[View Logs...](#)

Milestone 2: Bridge “subscription” model

With Tor Browser automation, next step use the signaling channel to ask for a replacement bridge.

The subscription model is a key building block for future plans:

- auto replace bridges that go down
- fast flux bridges that get blocked
- reputation-based bridge distribution schemes like Salmon

For first two, client sends hash-of-bridge-fingerprint to prove knowledge of bridge. For Salmon/Lox, it's a cryptographic identity.

Need to complete the feedback loop

With the “vantage point”, “subscription model”, “dynamic cloud bridges”, and “signaling channel” building blocks, the next step is to complete the feedback loop: we need to auto notice that bridges have been blocked, and auto replace the bridge for the user.

At the client side: When exactly should Tor Browser launch a bridge refresh attempt? How to distinguish from simple network failure.

At the service side: use heuristics for prioritizing which bridges to test from our sensitive vantage points, so we only risk a scan when it's probably blocked.

Get really good at knowing when bridges go offline

Scanning for ‘down’ is a different race than scanning for ‘blocked’, but getting really good at the former can help the latter.

Data sources:

- Self-reported usage data from bridge extrainfo descriptors
- Client-initiated bridge replacement requests to rdsys
- Better bridgestrap designs to quickly detect bridges going offline



Long term project: feedback loop to dynamically adjust bridge pool sizes (inspired by proximax)



Now that we have bridgestrap working, and we are ramping up our in-country deployments for assessing which bridges are getting blocked, we're back in position to reach another long-wanted milestone for the bridge distribution system. The proximax-inspired bridge distribution "meta-strategy" is simple to explain: keep track of which bridge distribution strategies are effective, and automatically send new bridges toward the strategies that are succeeding.

As described in the "five ways to test bridge reachability" blog post:

"We can define the *efficiency* of a bridge address in terms of how many people use it and how long before it gets blocked. So a bridge that gets blocked very quickly scores a low efficiency, a bridge that doesn't get blocked but doesn't see much use scores a medium efficiency, and a popular bridge that doesn't get blocked scores high. We can characterize the efficiency of a distribution channel as a function of the efficiency of the bridges it distributes. The key insight is that we then adapt how many new bridges we give to each distribution channel based on its efficiency. So channels that are working well automatically get more addresses to give out, and channels that aren't working well automatically end up with fewer addresses."

Many more details in that blog post (<https://blog.torproject.org/research-problem-five-ways-test-bridge-reachability/>) and in the original research paper (<https://www.freehaven.net/anonbib/#proximax11>).

In terms of roadmap, here are possible steps.

Phase one, getting the building blocks in place:

- ☐ Pick a utility function for how to assess a given bridge. Start with something simple and then iterate. For example, "current



1



No mil...



None



None





Long term project: feedback loop to dynamically adjust bridge pool sizes (inspired by proximax)



- ☐ Pick a utility function for how to assess a given bridge. Start with something simple and then iterate. For example, "current number of users times number of days we've been giving it out and it's been unblocked." A later more complex function might be "for each country where it has users, add up the (# of users in that country times the number of days it's been reachable from that country)." A third iteration might be to assign a per-country weight multiplier based on how much blocking the country is experiencing, how much we want to prioritize that country, etc. Another idea is to look at bandwidth use (more is better) in addition to simple user count.
- ☐ Work with the metrics team to make decisions about the data architecture. Calculating bridge scores involves looking at historical info about bridges, and bridge usage, and bridge reachability. Maybe this is all better done inside onionoo -- and then the rest of the metrics pipeline like relay-search is in a better position to use it too.
- ☐ Pick a composite utility function for how to assess a given distribution strategy given scores for each bridge in its pool. Again start simple, for example "add up the utility scores for each bridge in that pool."
- ☐ Take a step back: Examine the scores we have, and see if it matches our intuition. Retune the utility functions. Put up some ongoing graphs on the metrics pages. The per-strategy summary scores are already useful at this point, to let us and others see which strategies are being effective.

Phase two, automating the feedback loop:

- ☐ Research: based on the above experience, invent utility functions which are both effective and more robust to gaming. In particular, consider how to handle attacks where colluding bridges report inflated user counts, to make a given distribution strategy look better than it really is. And make sure our utility functions converge, rather than e.g. letting one strategy's score



1



No mil...



None



None



Milestone 4: Trust-based distribution (1/2)

(designs like Salmon, rBridge, Hyphae, Lox)

With Tor Browser doing background connections to maintain your bridges, now rather than just replacing bridges when they die, instead track whether the bridges you gave somebody got blocked.

Get rid of users whose bridges tend to get blocked, and reward users whose bridges last a long time.

Milestone 4: Trust-based distribution (2/2)

Need to pick the right parameters so good users get enough bridges yet bad people can't find too many.

Salmon lets trusted users invite friends to higher trust levels; if a user misbehaves blame their sponsor too. But...privacy-preserving social graph management? Lox?

This is where the arms race needs to go: focus on *exploiting asymmetries* against well-funded censors.

Race follow-up roadmap

- (1) Understand where we are censored and react to it
 - design and implement subscription model
 - use it to auto-replace broken bridges for Tor Browser users
- (2) Strengthen Snowflake
 - instrument each layer to better understand usage
 - redundancy of components, look at protocol-level attacks
- (3) Make our tooling more agile
 - redesign the bridge assignment / snowflake assignment code
 - staging (sandbox) environments and continuous integration