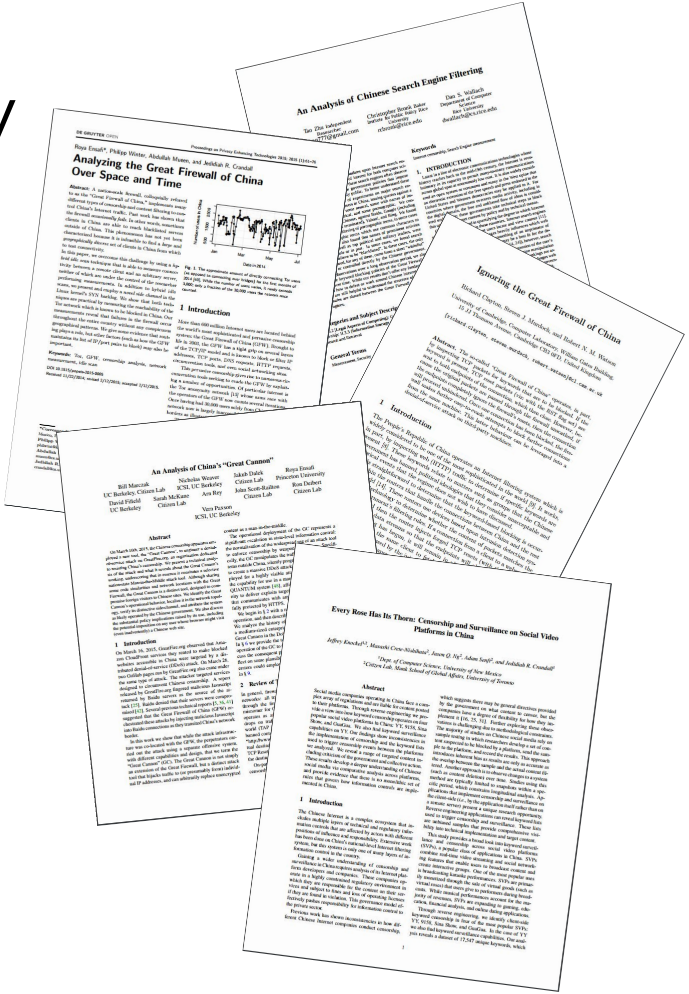
How the **Great Firewall** discovers **hidden circumvention servers**

Roya Ensafi
David Fifield
Philipp Winter
Nick Weaver
Nick Feamster
Vern Paxson

Much already known about GFW

- Numerous research papers and blog posts
  - Open access library: censorbib.nymity.ch
- We know…
  - What is blocked
  - How it is blocked
  - Where the GFW is, topologically
- Unfortunately, most studies are one-off
  - Continuous measurements challenging



An Analysis of Chinese Search Engine Filtering

Tao Zhu, Christopher Bronk, Dan S. Wallach

Analyzing the Great Firewall of China Over Space and Time

Roya Ensafi*, Philipp Winter, Abdullah Mueen, and Jedidiah R. Crandall

Ignoring the Great Firewall of China

Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson

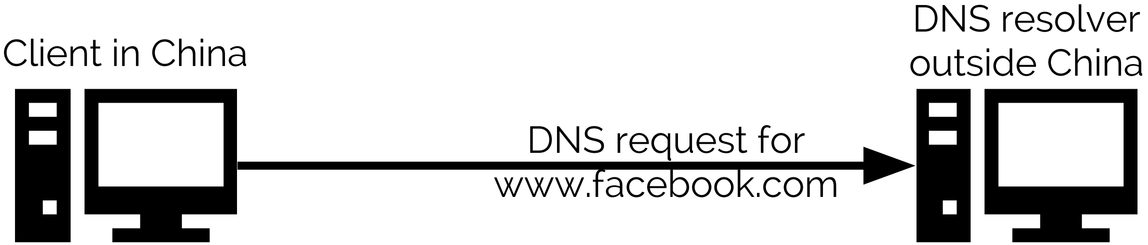
An Analysis of China's "Great Cannon"

Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, Vern Paxson

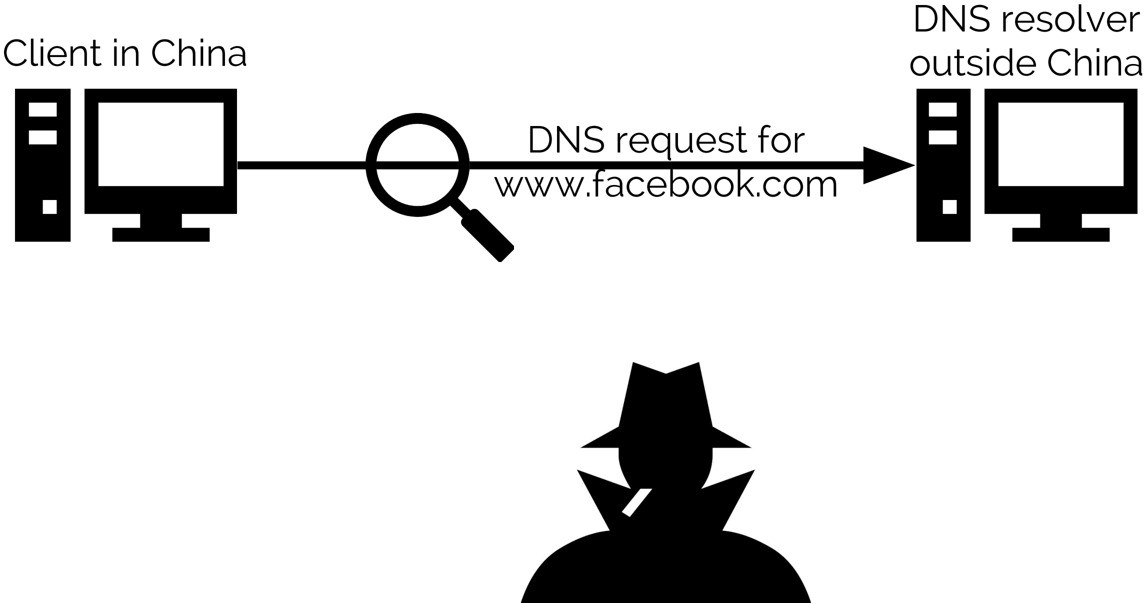
Every Rose Has Its Thorn: Censorship and Surveillance on Social Video Platforms in China

Jeffrey Knockel, Masashi Crete-Nishihata, Jason Q. Ng, Adam Senft, and Jedidiah R. Crandall

Many domains are blocked



Many domains are blocked

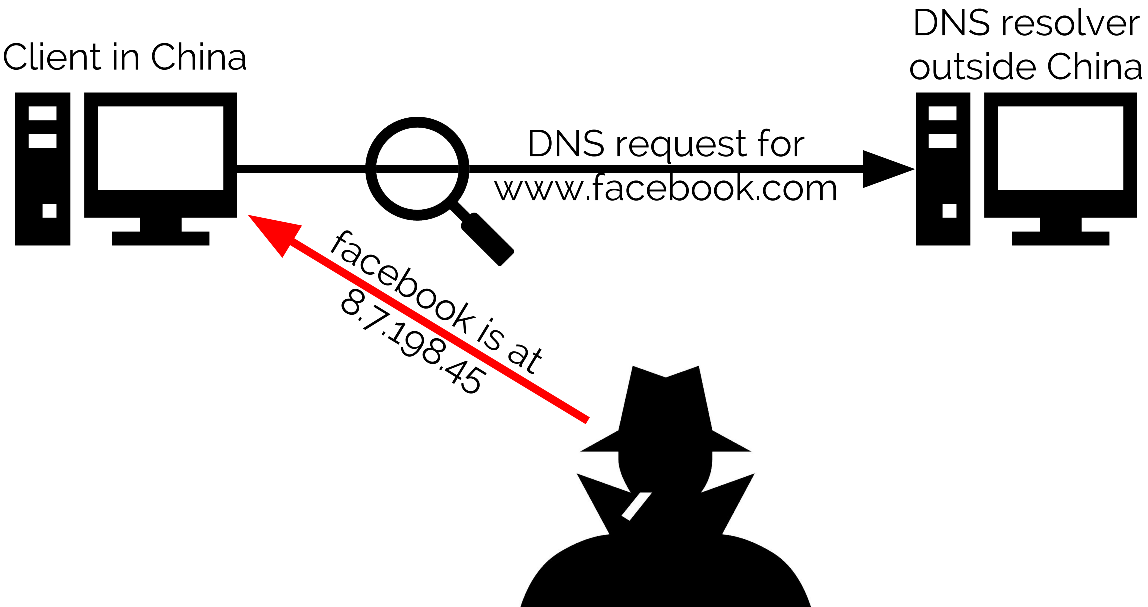


Client in China

DNS resolver outside China

DNS request for www.facebook.com

Many domains are blocked



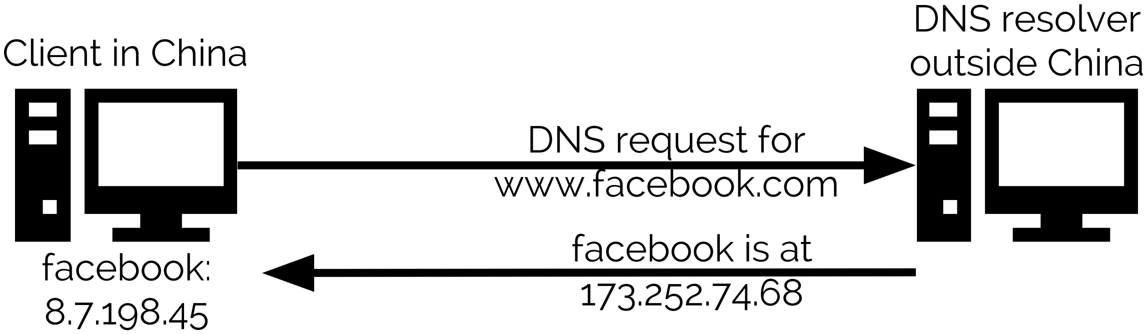
Client in China

DNS resolver outside China

DNS request for www.facebook.com

facebook is at 8.7.198.45

Many domains are blocked



Client in China

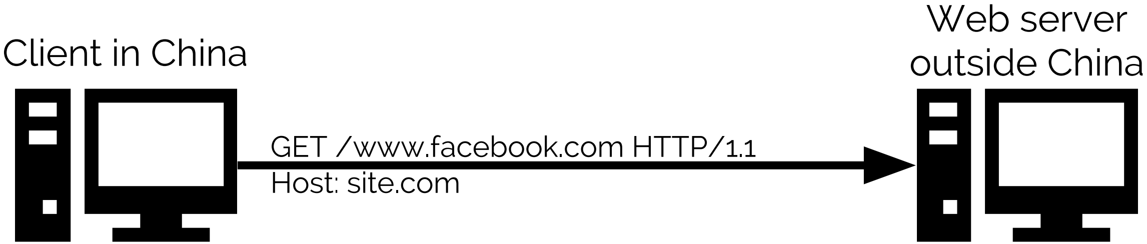
DNS resolver outside China

DNS request for www.facebook.com

facebook is at 173.252.74.68

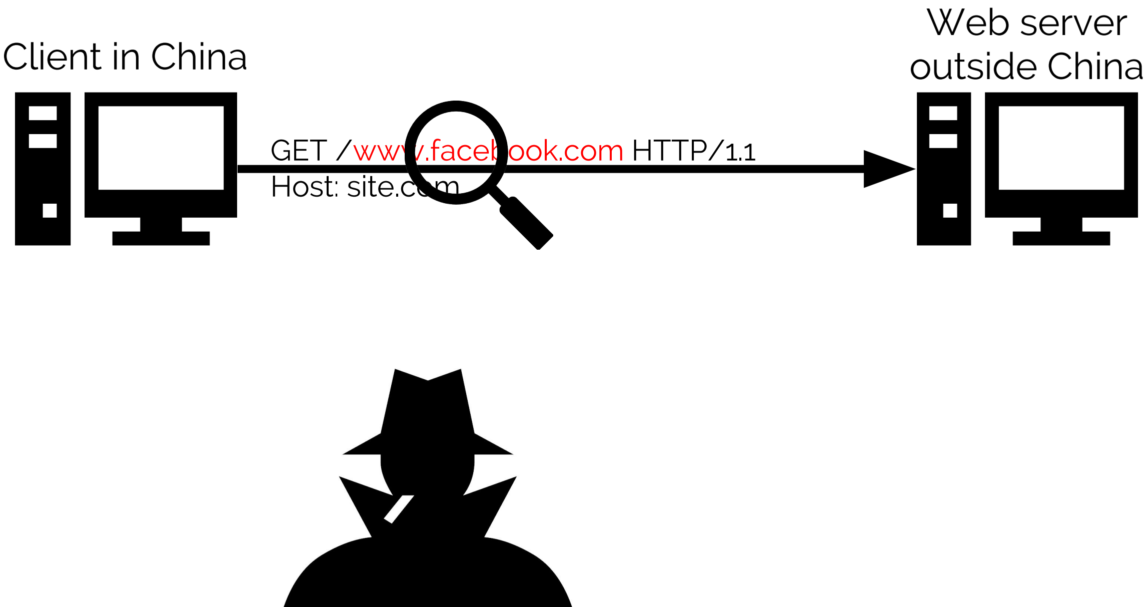
facebook: 8.7.198.45

Many keywords are blocked



```
GET /www.facebook.com HTTP/1.1
Host: site.com
```

Many keywords are blocked

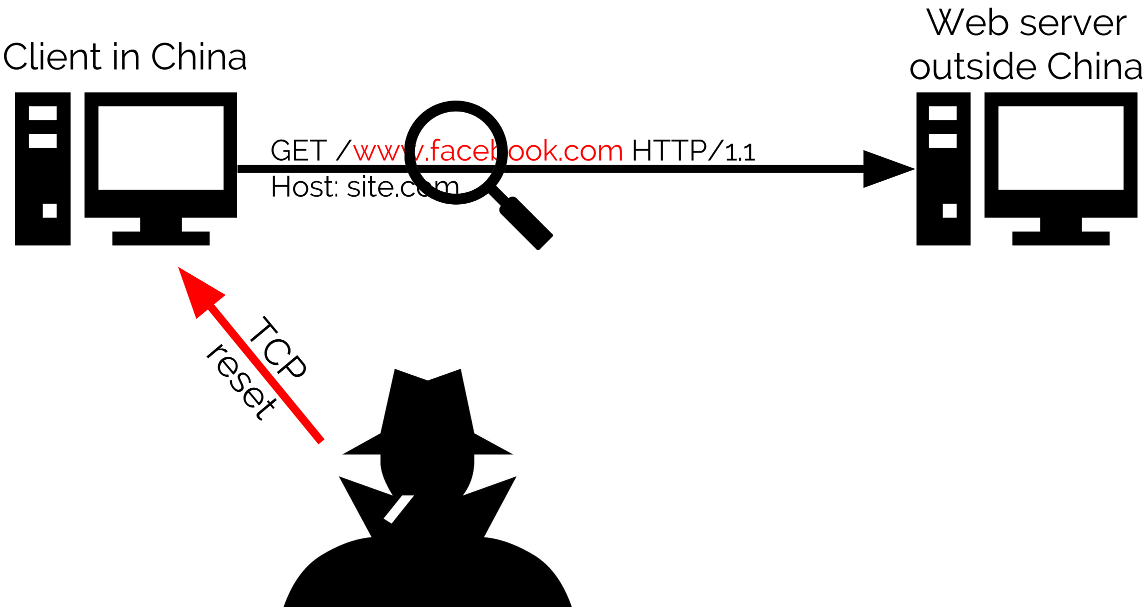


Client in China

Web server
outside China

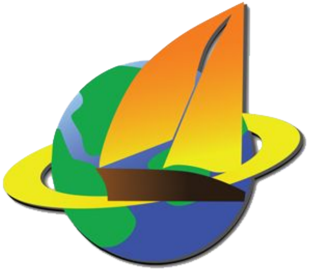
```
GET /www.facebook.com HTTP/1.1
Host: site.com
```

Many keywords are blocked

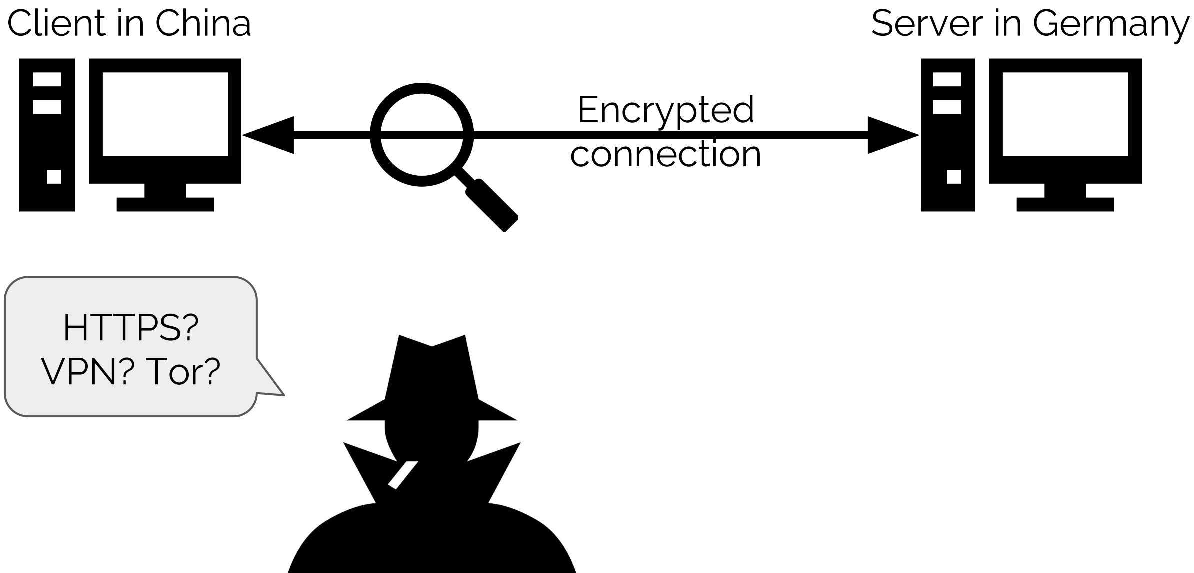


GoAgent

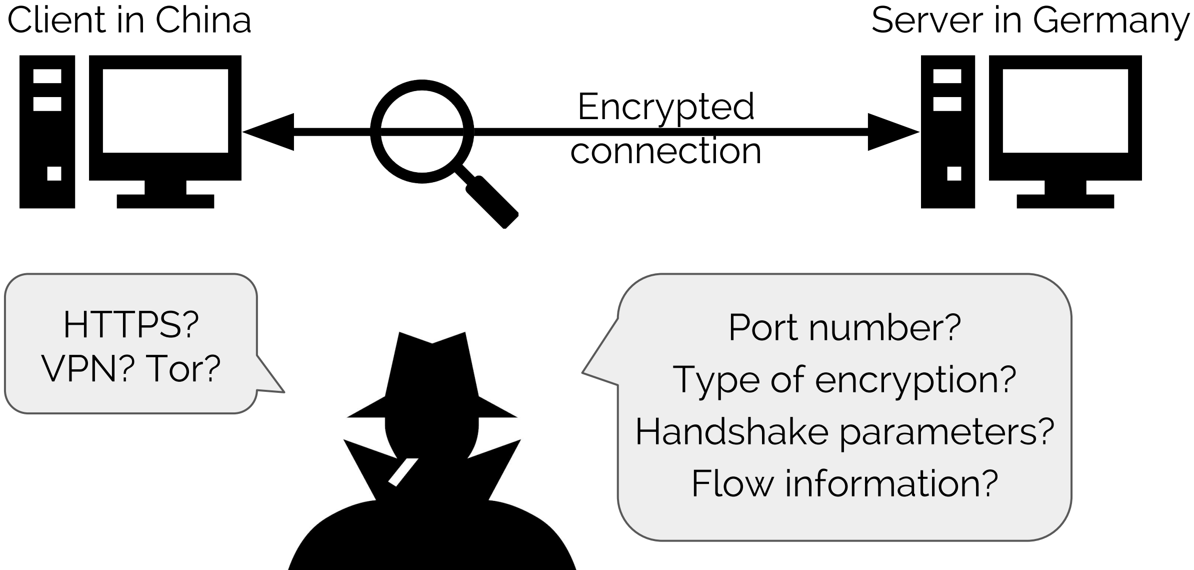
PSIPHON



Encryption reduces blocking accuracy



Encryption reduces blocking accuracy



Client in China

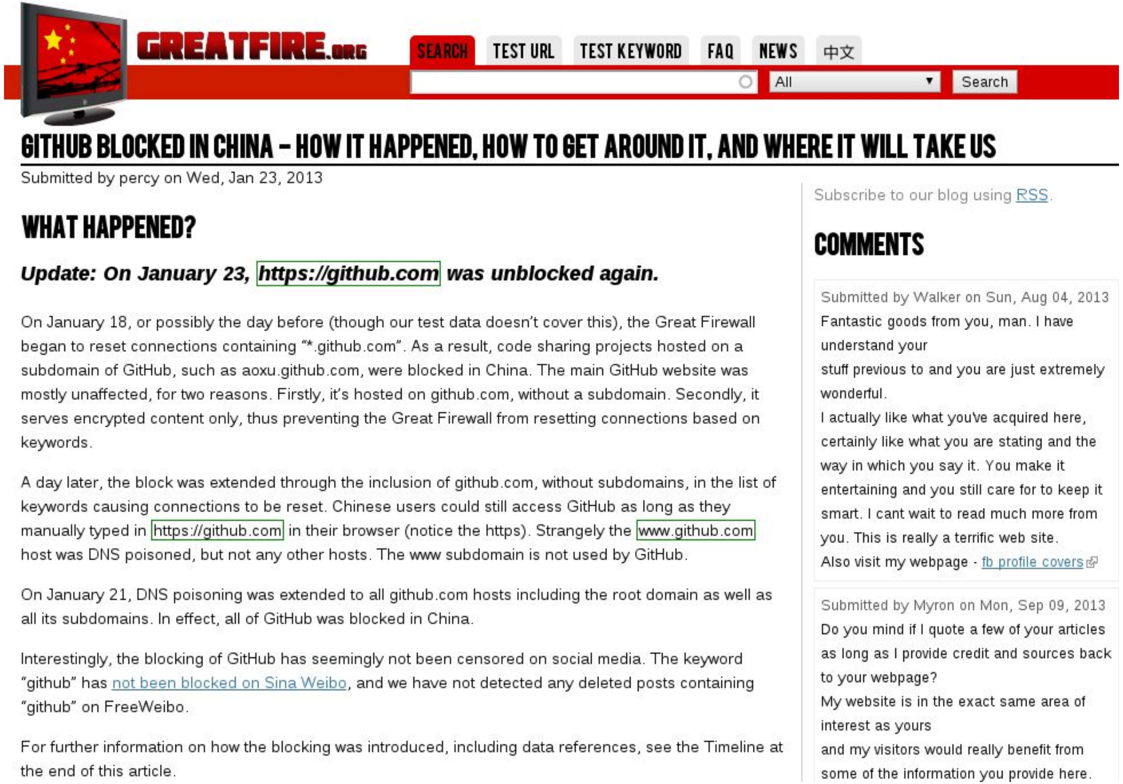
Server in Germany

Encrypted connection

HTTPS?
VPN? Tor?

Port number?
Type of encryption?
Handshake parameters?
Flow information?

Censors often test how far they can go



GITHUB BLOCKED IN CHINA – HOW IT HAPPENED, HOW TO GET AROUND IT, AND WHERE IT WILL TAKE US

Submitted by percy on Wed, Jan 23, 2013

WHAT HAPPENED?

Update: On January 23, https://github.com was unblocked again.

On January 18, or possibly the day before (though our test data doesn't cover this), the Great Firewall began to reset connections containing "*.github.com". As a result, code sharing projects hosted on a subdomain of GitHub, such as aoxu.github.com, were blocked in China. The main GitHub website was mostly unaffected, for two reasons. Firstly, it's hosted on github.com, without a subdomain. Secondly, it serves encrypted content only, thus preventing the Great Firewall from resetting connections based on keywords.

A day later, the block was extended through the inclusion of github.com, without subdomains, in the list of keywords causing connections to be reset. Chinese users could still access GitHub as long as they manually typed in https://github.com in their browser (notice the https). Strangely the www.github.com host was DNS poisoned, but not any other hosts. The www subdomain is not used by GitHub.

On January 21, DNS poisoning was extended to all github.com hosts including the root domain as well as all its subdomains. In effect, all of GitHub was blocked in China.

Interestingly, the blocking of GitHub has seemingly not been censored on social media. The keyword "github" has not been blocked on Sina Weibo, and we have not detected any deleted posts containing "github" on FreeWeibo.

For further information on how the blocking was introduced, including data references, see the Timeline at the end of this article.

Subscribe to our blog using RSS.

COMMENTS

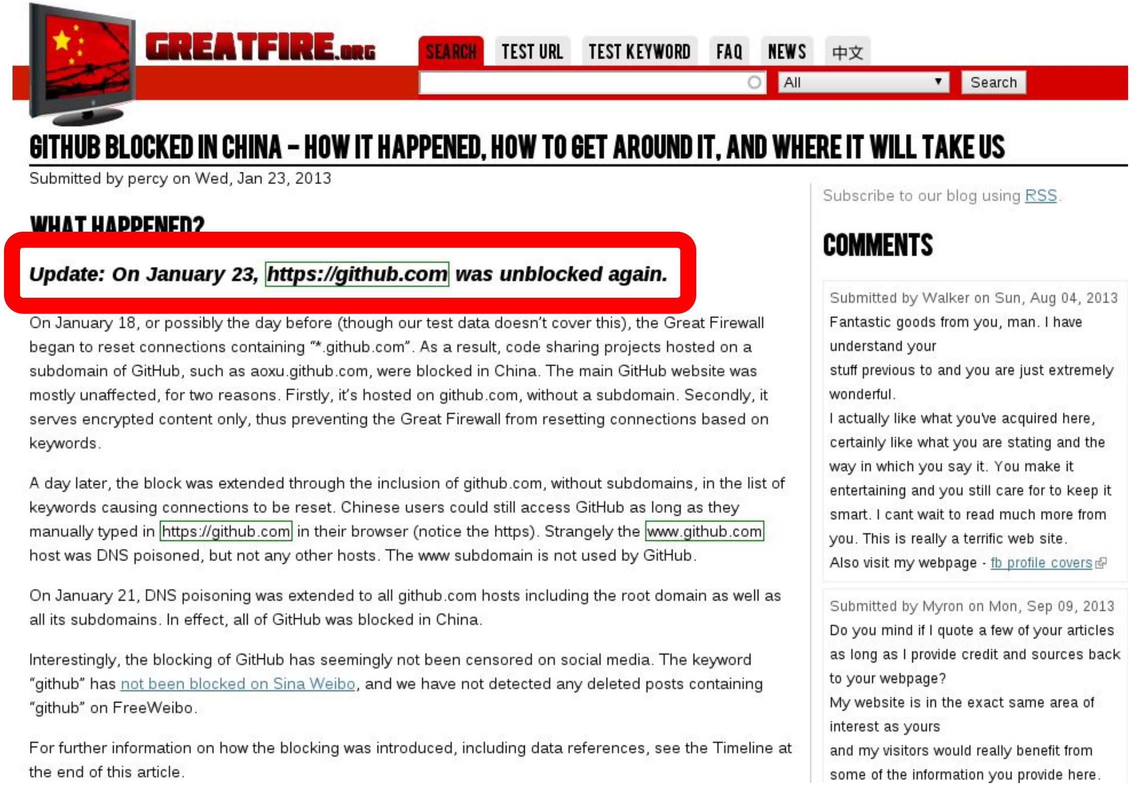
Submitted by Walker on Sun, Aug 04, 2013

Fantastic goods from you, man. I have understand your
stuff previous to and you are just extremely wonderful.
I actually like what you've acquired here, certainly like what you are stating and the way in which you say it. You make it entertaining and you still care for to keep it smart. I cant wait to read much more from you. This is really a terrific web site.
Also visit my webpage - fb profile covers

Submitted by Myron on Mon, Sep 09, 2013

Do you mind if I quote a few of your articles as long as I provide credit and sources back to your webpage?
My website is in the exact same area of interest as yours
and my visitors would really benefit from some of the information you provide here.

Censors often test how far they can go



GITHUB BLOCKED IN CHINA – HOW IT HAPPENED, HOW TO GET AROUND IT, AND WHERE IT WILL TAKE US

Submitted by percy on Wed, Jan 23, 2013

WHAT HAPPENED?

Update: On January 23, https://github.com was unblocked again.

On January 18, or possibly the day before (though our test data doesn't cover this), the Great Firewall began to reset connections containing "*.github.com". As a result, code sharing projects hosted on a subdomain of GitHub, such as aoxu.github.com, were blocked in China. The main GitHub website was mostly unaffected, for two reasons. Firstly, it's hosted on github.com, without a subdomain. Secondly, it serves encrypted content only, thus preventing the Great Firewall from resetting connections based on keywords.

A day later, the block was extended through the inclusion of github.com, without subdomains, in the list of keywords causing connections to be reset. Chinese users could still access GitHub as long as they manually typed in https://github.com in their browser (notice the https). Strangely the www.github.com host was DNS poisoned, but not any other hosts. The www subdomain is not used by GitHub.

On January 21, DNS poisoning was extended to all github.com hosts including the root domain as well as all its subdomains. In effect, all of GitHub was blocked in China.

Interestingly, the blocking of GitHub has seemingly not been censored on social media. The keyword "github" has not been blocked on Sina Weibo, and we have not detected any deleted posts containing "github" on FreeWeibo.

For further information on how the blocking was introduced, including data references, see the Timeline at the end of this article.

Subscribe to our blog using RSS.

COMMENTS

Submitted by Walker on Sun, Aug 04, 2013

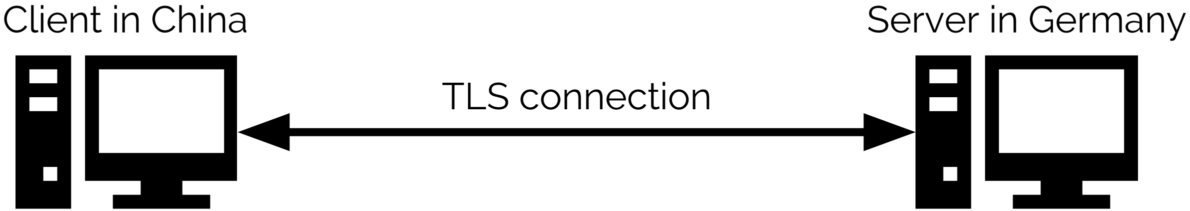
Fantastic goods from you, man. I have understand your stuff previous to and you are just extremely wonderful.
I actually like what you've acquired here, certainly like what you are stating and the way in which you say it. You make it entertaining and you still care for to keep it smart. I cant wait to read much more from you. This is really a terrific web site.
Also visit my webpage - fb profile covers

Submitted by Myron on Mon, Sep 09, 2013

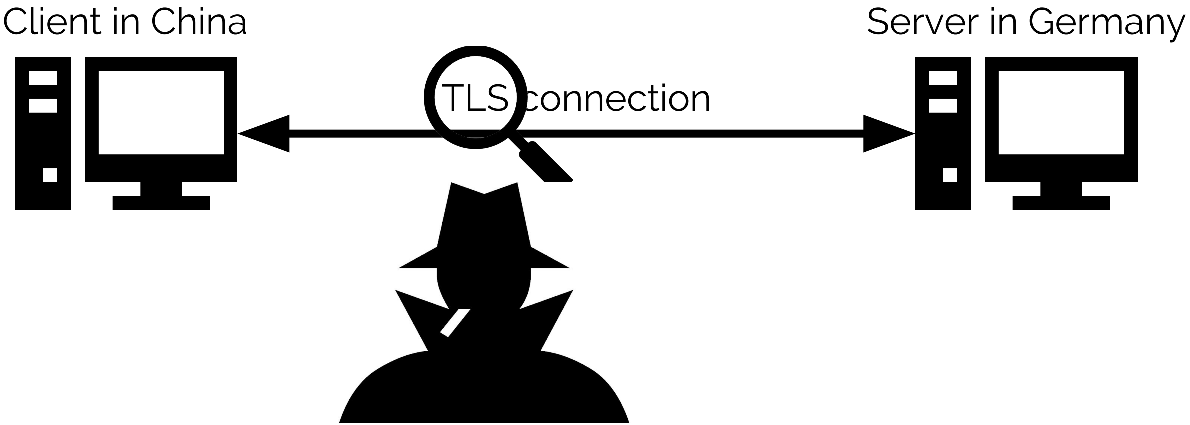
Do you mind if I quote a few of your articles as long as I provide credit and sources back to your webpage?
My website is in the exact same area of interest as yours
and my visitors would really benefit from some of the information you provide here.

Active Probing

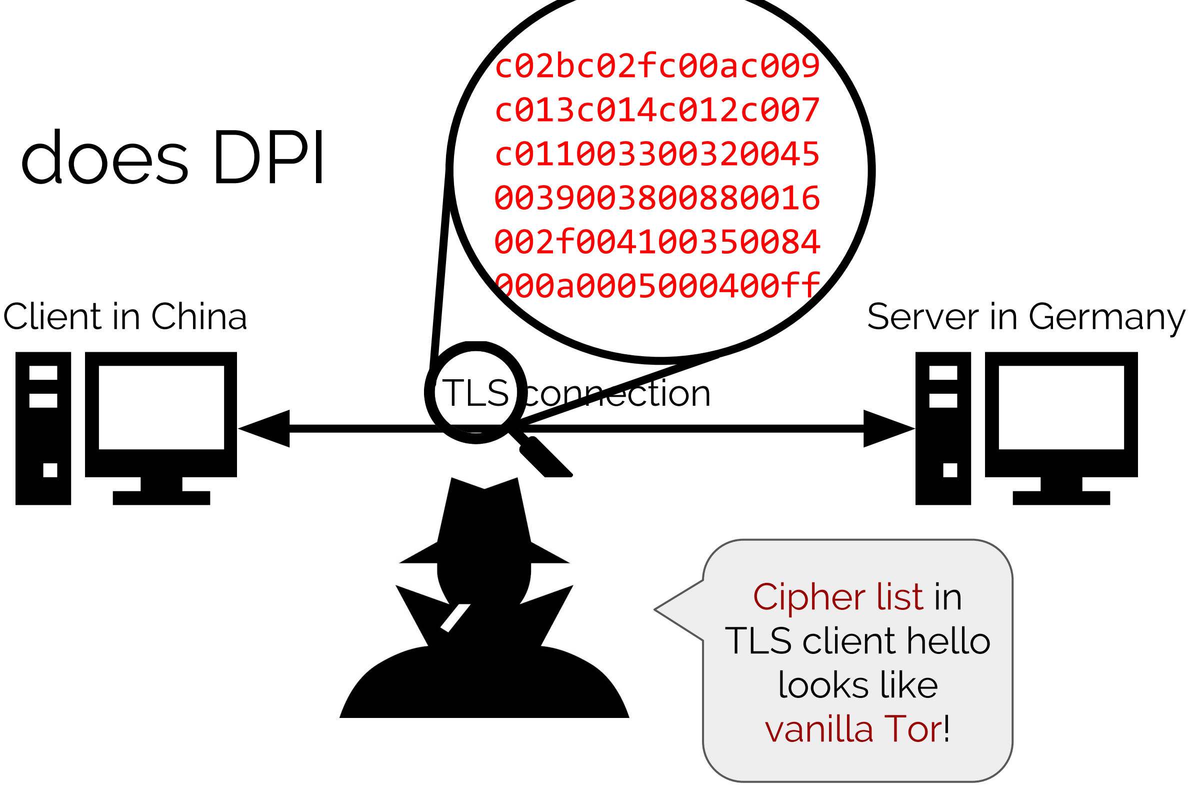
Assume an encrypted tunnel



1. GFW does DPI



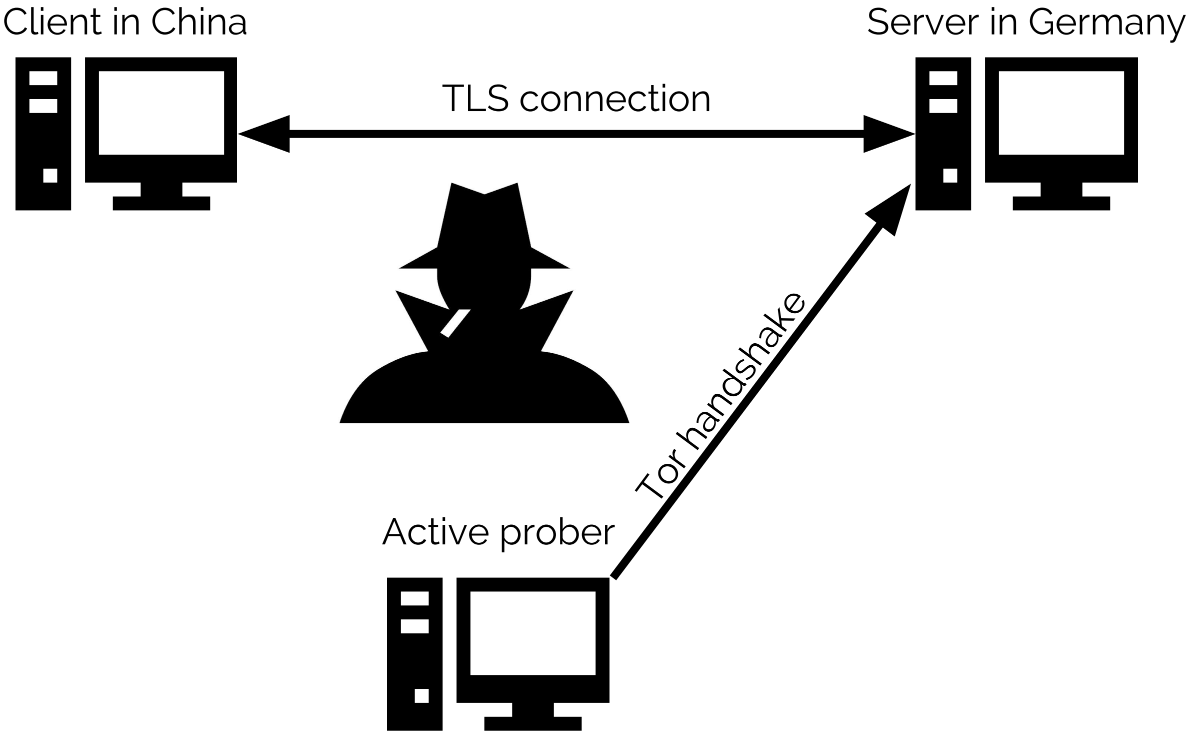
1. GFW does DPI



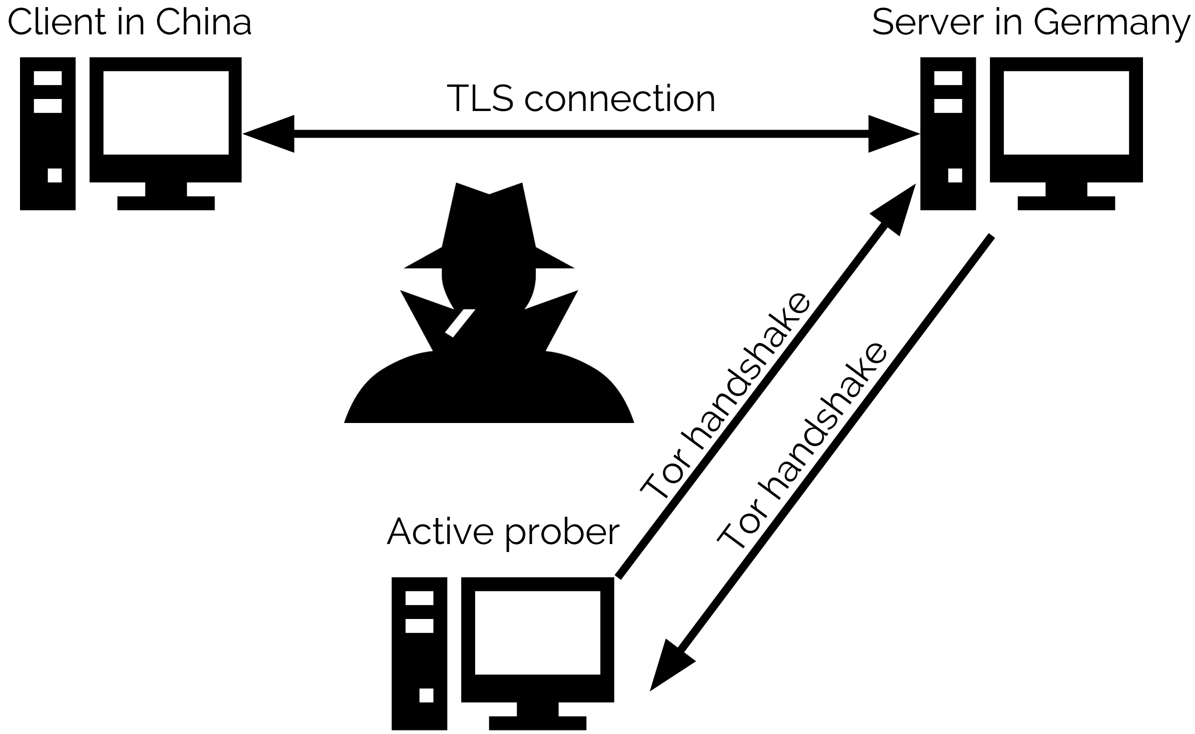
```
c02bc02fc00ac009
c013c014c012c007
c011003300320045
0039003800880016
002f004100350084
000a0005000400ff
```

Cipher list in TLS client hello looks like vanilla Tor!

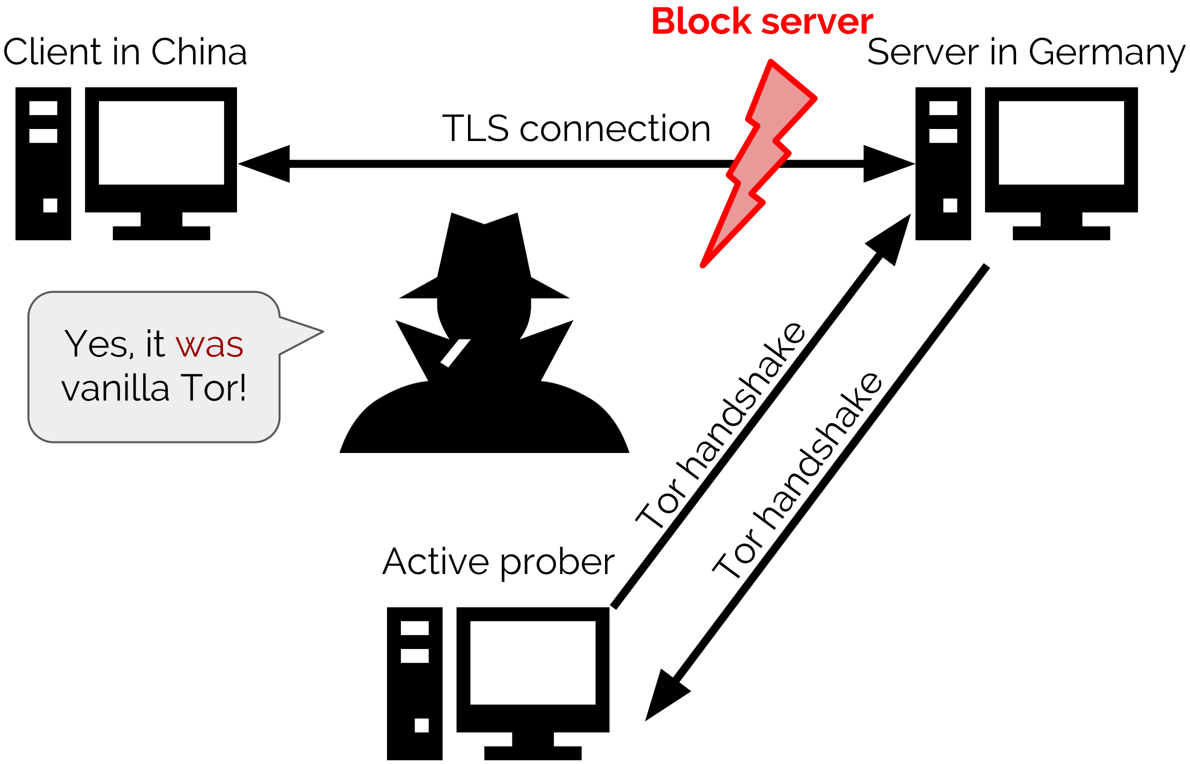
2. GFW launches active probe



2. GFW launches active probe

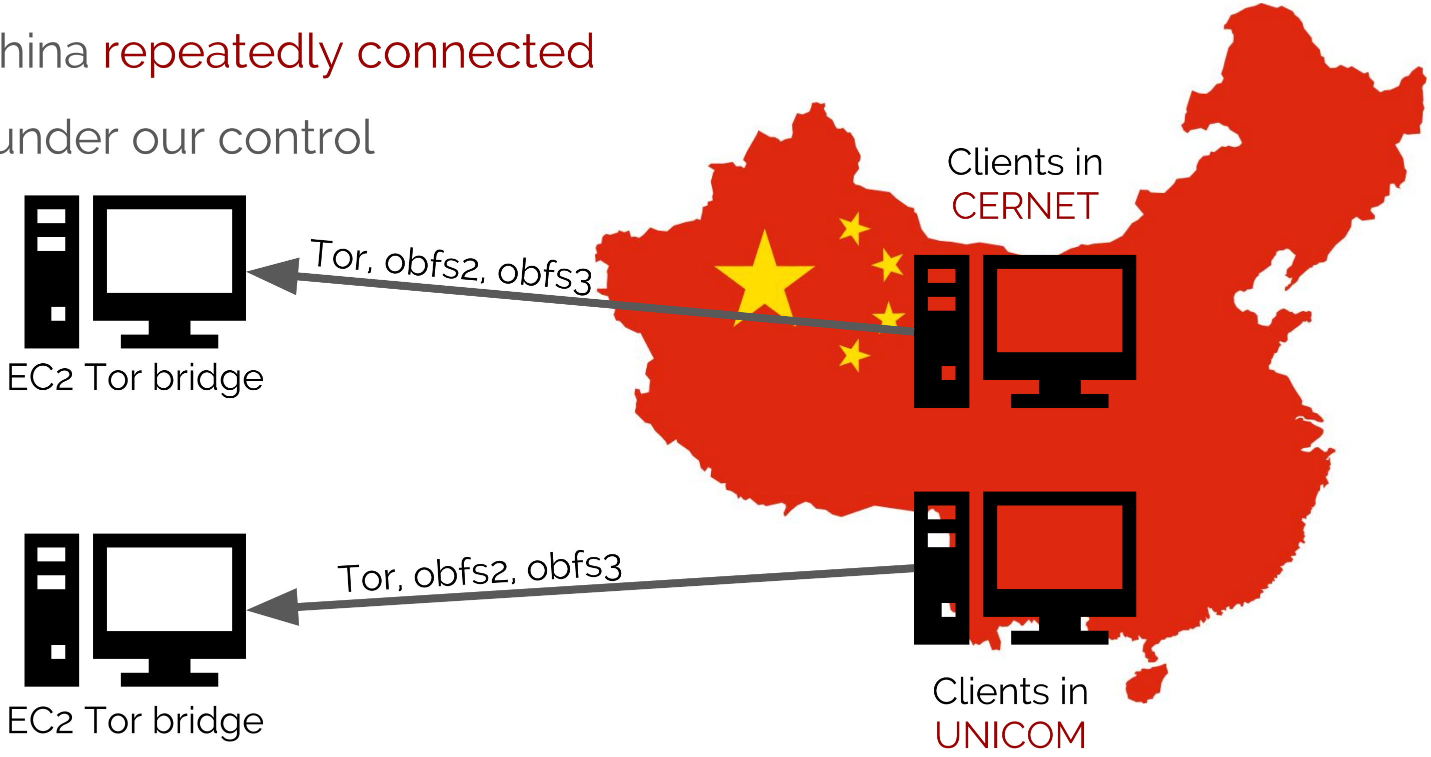


3. GFW blocks server



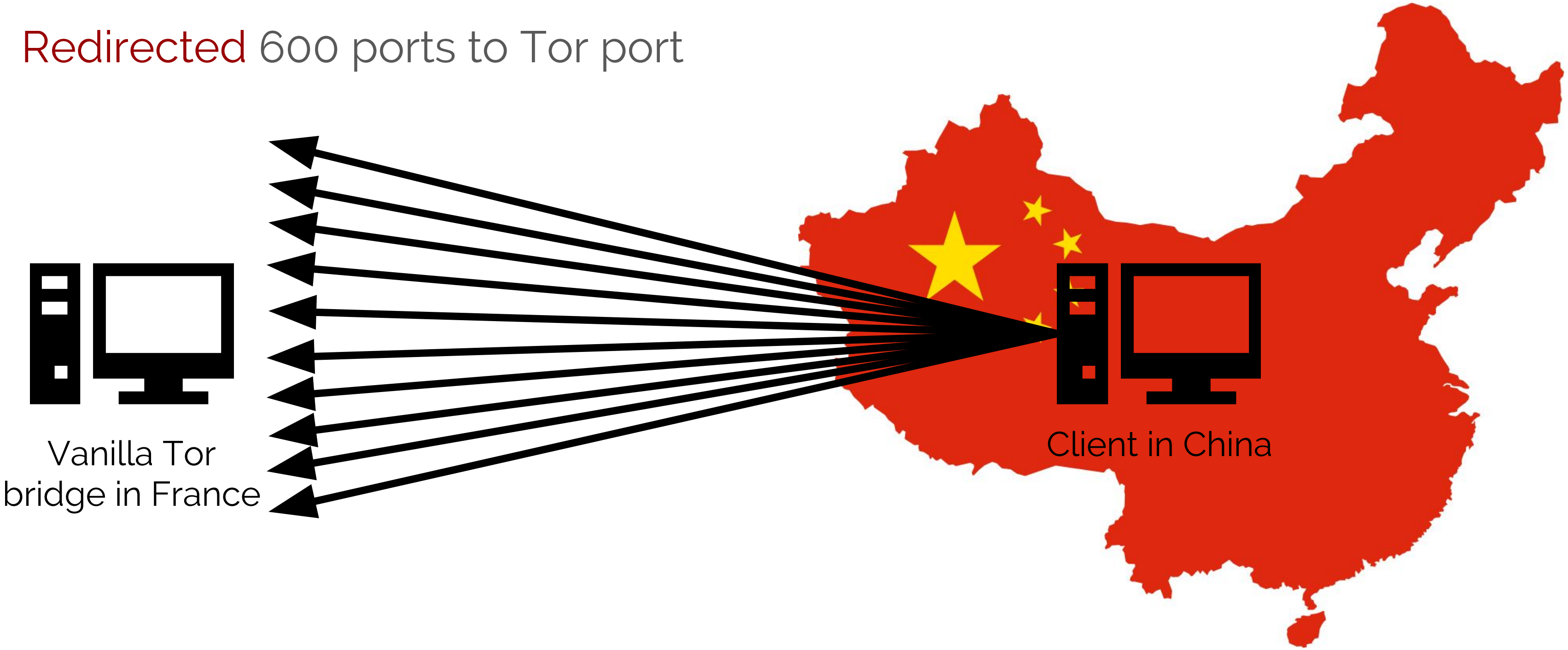
Our "Shadow" dataset

- Clients in China repeatedly connected to bridges under our control



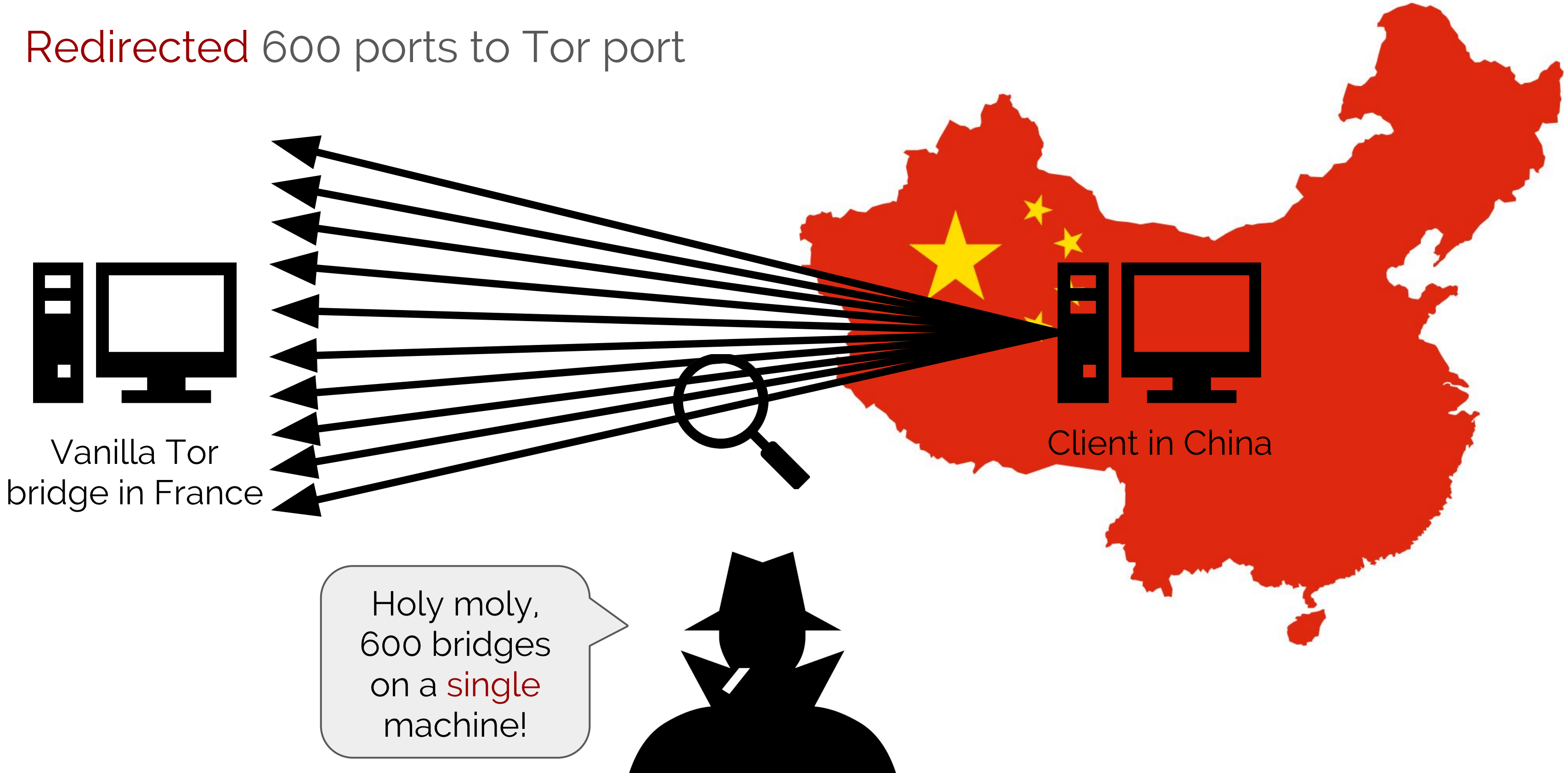
Our "Sybil" dataset

- Redirected 600 ports to Tor port



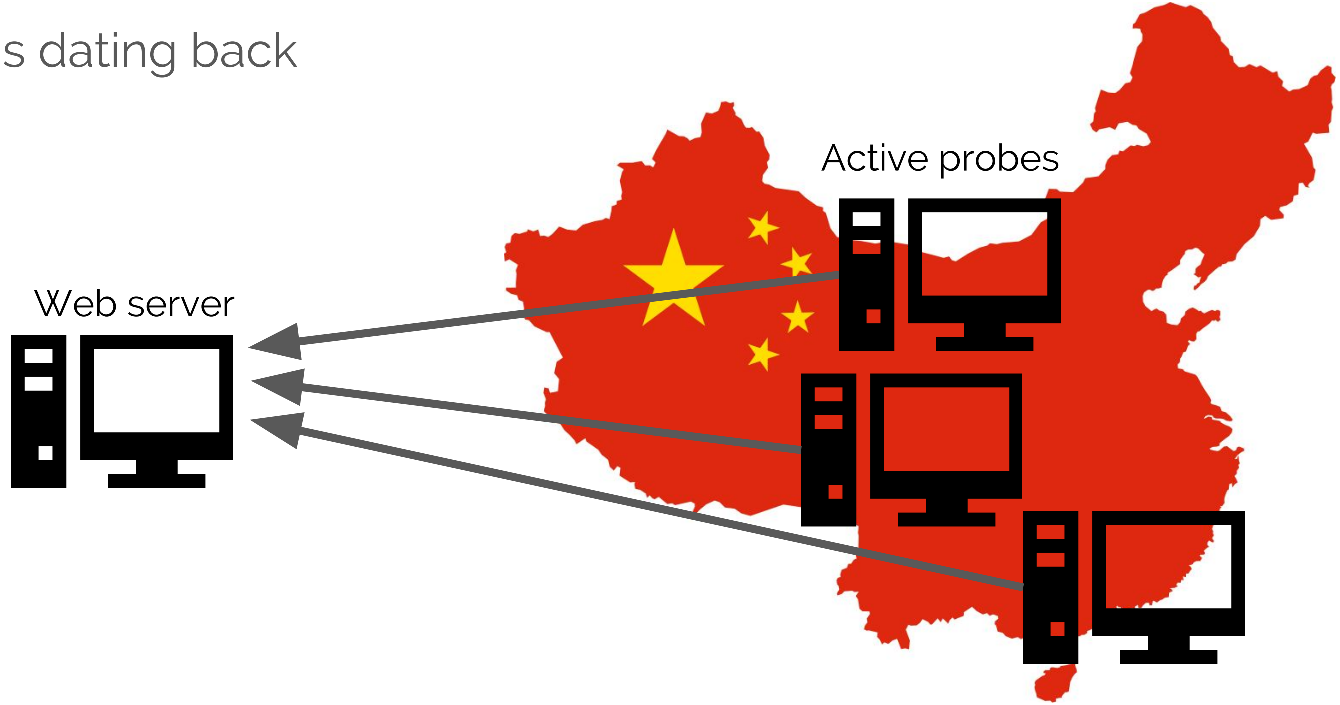
Our "Sybil" dataset

- Redirected 600 ports to Tor port


Vanilla Tor bridge in France
Client in China
Holy moly, 600 bridges on a single machine!

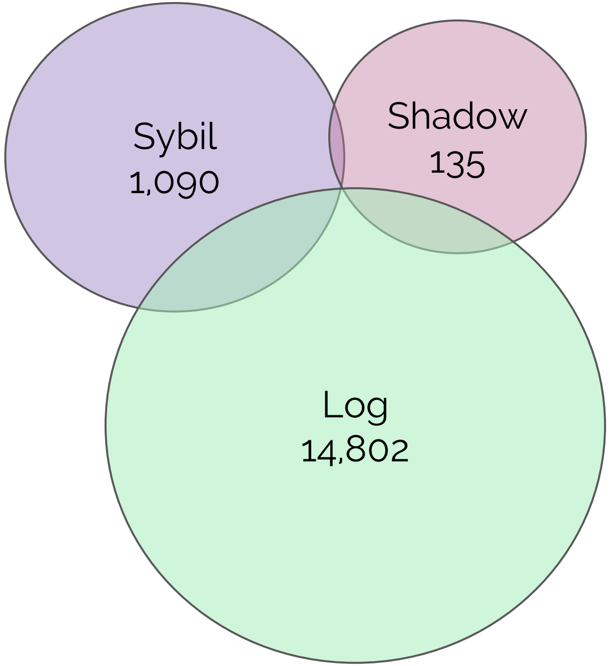
Our "Log" dataset

- Web server logs dating back to Jan 2010



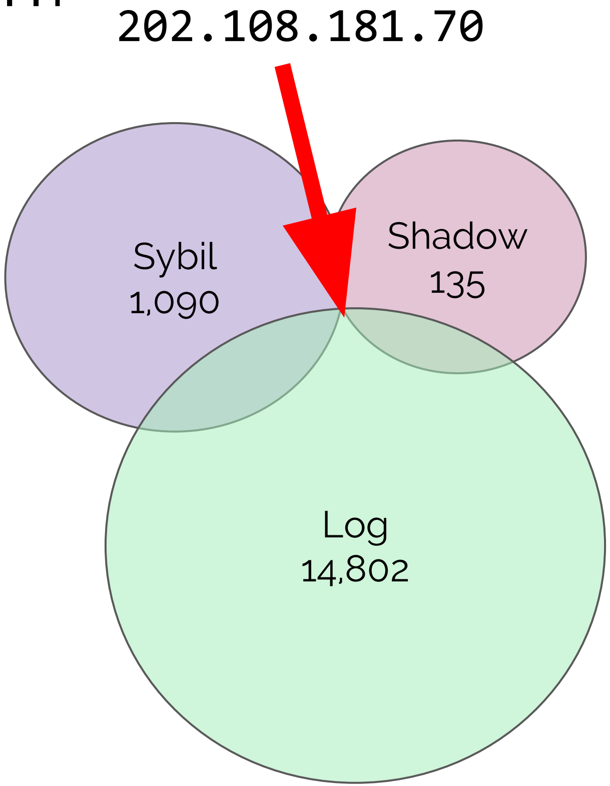
Where are the probes coming from?

- Collected 16,083 unique prober IP addresses
- 95% of addresses seen only once
- Reverse DNS suggests ISP pools
  - adsl-pool.sx.cn
  - kd.ny.adsl
  - online.tj.cn
- Majority of probes come from three autonomous systems
  - ASN 4837, 4134, and 17622



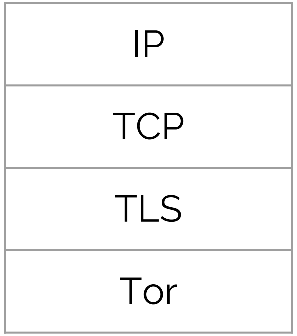
Where are the probes coming from?

- Collected 16,083 unique prober IP addresses
- 95% of addresses seen only once
- Reverse DNS suggests ISP pools
  - adsl-pool.sx.cn
  - kd.ny.adsl
  - online.tj.cn
- Majority of probes come from three autonomous systems
  - ASN 4837, 4134, and 17622



Are probes hijacking IP addresses?

- While probe is active, no other communication with probe possible
    - Traceroutes time out several hops before destination
    - Port scans say all ports are filtered
- What do probes have in common?
    - IP TTL
    - IP ID
    - TCP ISN
    - TCP TSval
    - TLS client hello
    - Pcaps online: nymity.ch/active-probing/



IP

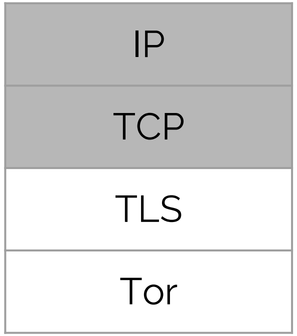
TCP

TLS

Tor

What do probes have in common?

- All probes…
  - Have narrow IP TTL distribution
  - Use source ports in entire 16-bit port range
  - Exhibit patterns in TCP TSval
- Does not seem like off-the-shelf networking stack
- User space TCP stack?



IP

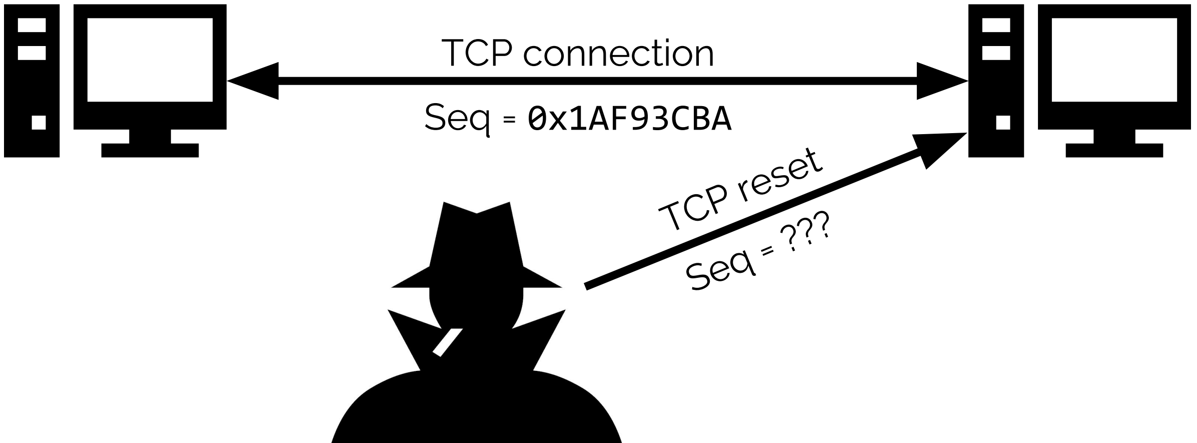
TCP

TLS

Tor

TCP's initial sequence numbers

- TCP uses 32-bit initial sequence numbers (ISNs)
- Protects against off-path attackers
- Attacker must guess correct ISN range to inject segments
- Every SYN segment should have random ISN



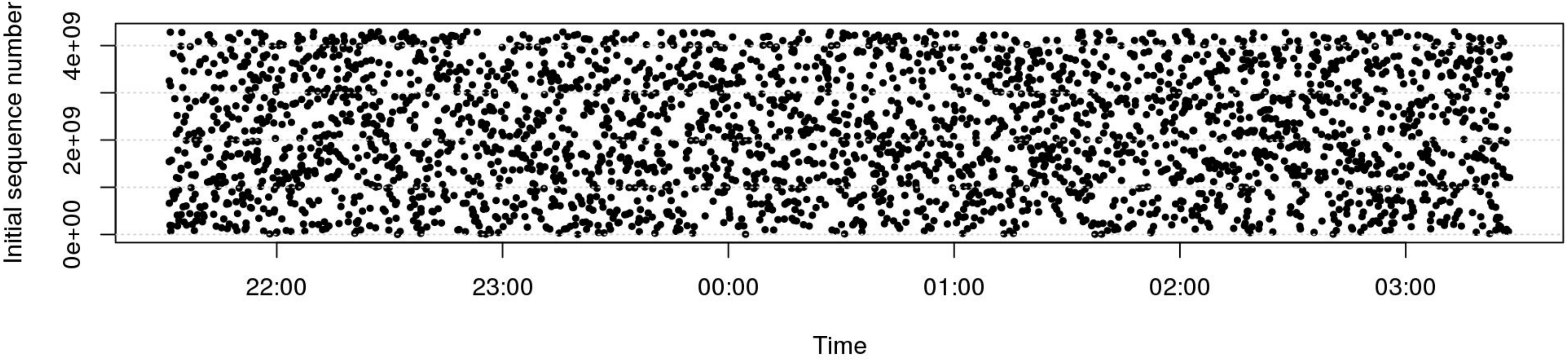
IP

TCP

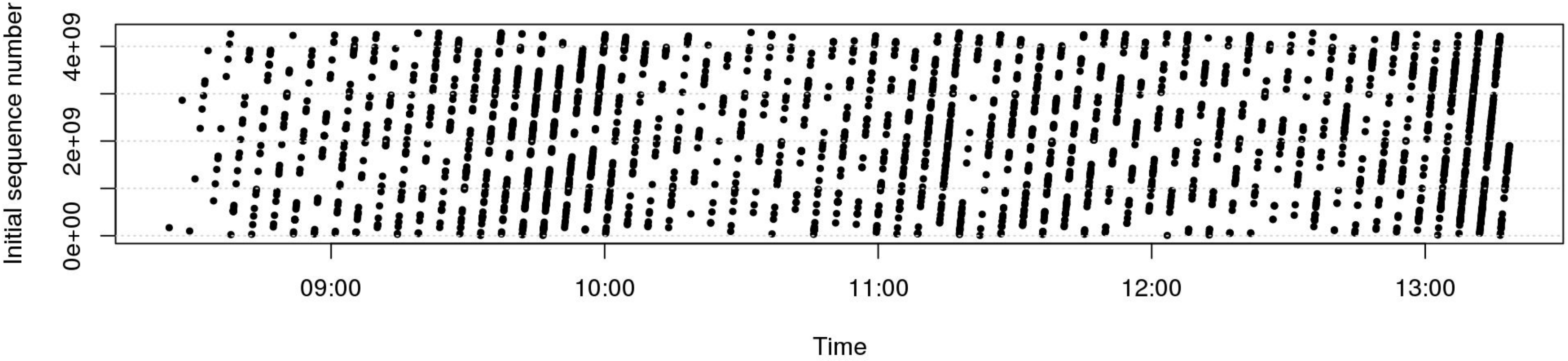
TLS

Tor

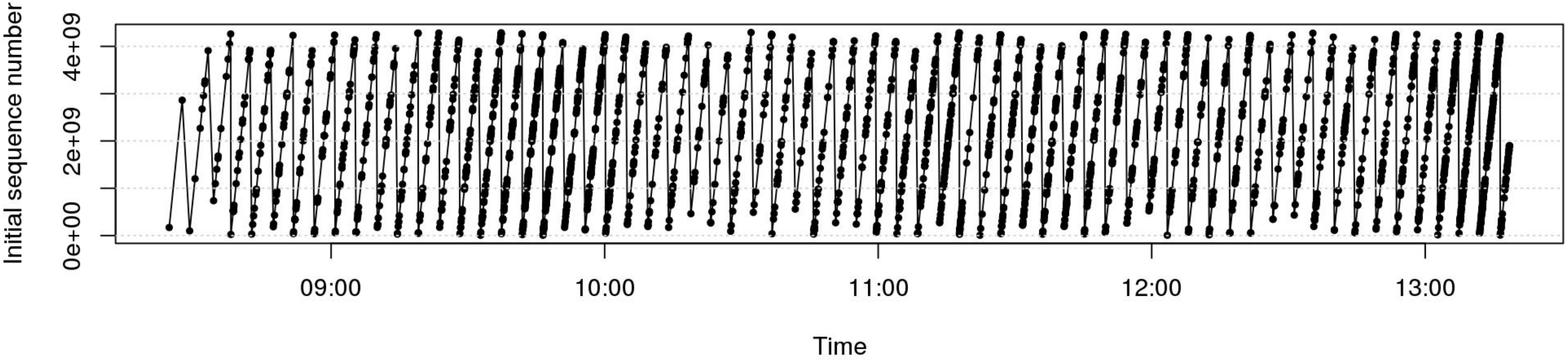
What we expected to see



What we did see

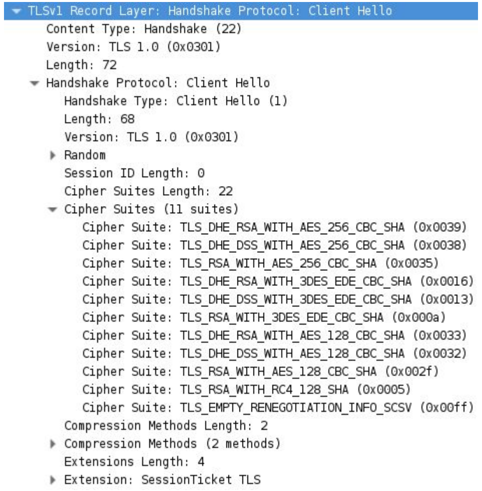


What we did see

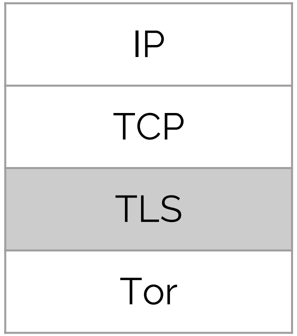


TLS fingerprint

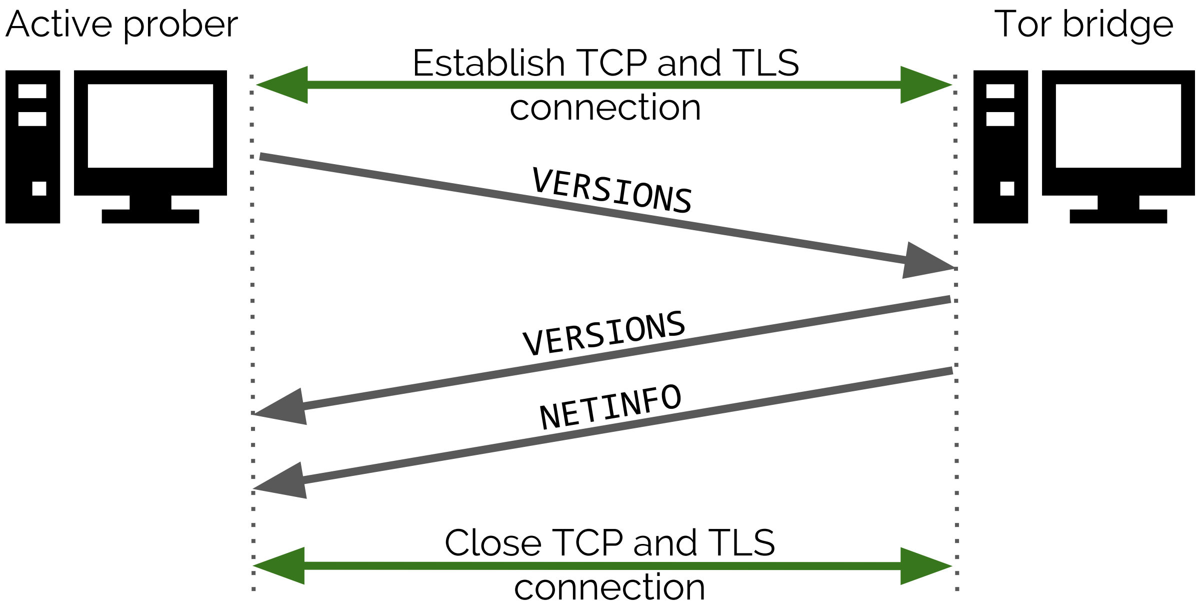
- Probes all share uncommon TLS client hello
- Not running original Tor client
  - No randomly-generated SNI
  - Unique (?) cipher suite
- Measured on a busy Tor guard relay:
  - Observed 236,101 client hellos over 24 hours
  - Only 67 (0.02%) had identical setup
  - Recorded only client hellos, no IP addresses



```
▾ TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 72
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 68
      Version: TLS 1.0 (0x0301)
    ▸ Random
      Session ID Length: 0
      Cipher Suites Length: 22
    ▾ Cipher Suites (11 suites)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
        Cipher Suite: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x0038)
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
        Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x0016)
        Cipher Suite: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x0013)
        Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
        Cipher Suite: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
        Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
        Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
        Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
      Compression Methods Length: 2
    ▸ Compression Methods (2 methods)
      Extensions Length: 4
    ▸ Extension: SessionTicket TLS
```



Tor probing



Active prober

Tor bridge

Establish TCP and TLS connection

VERSIONS

VERSIONS

NETINFO

Close TCP and TLS connection

IP

TCP

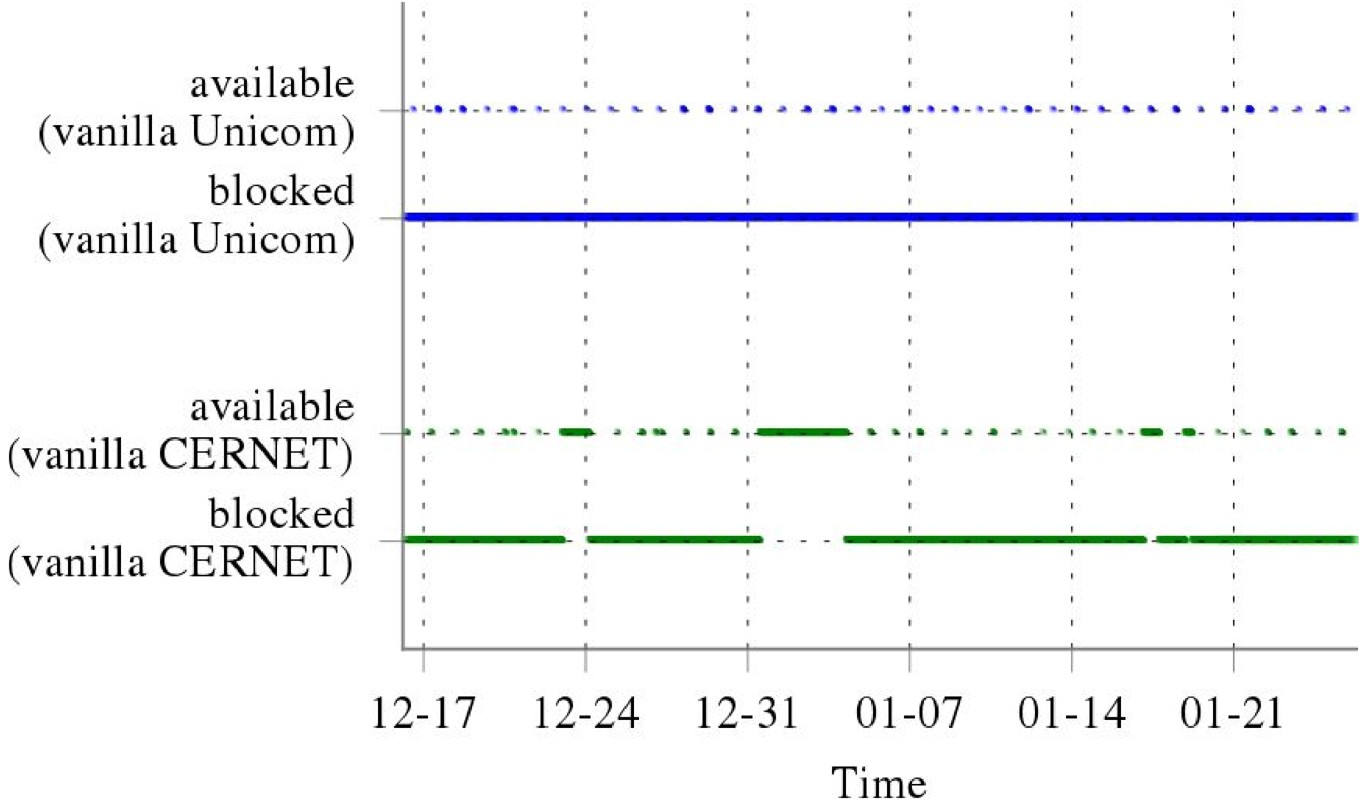
TLS

Tor

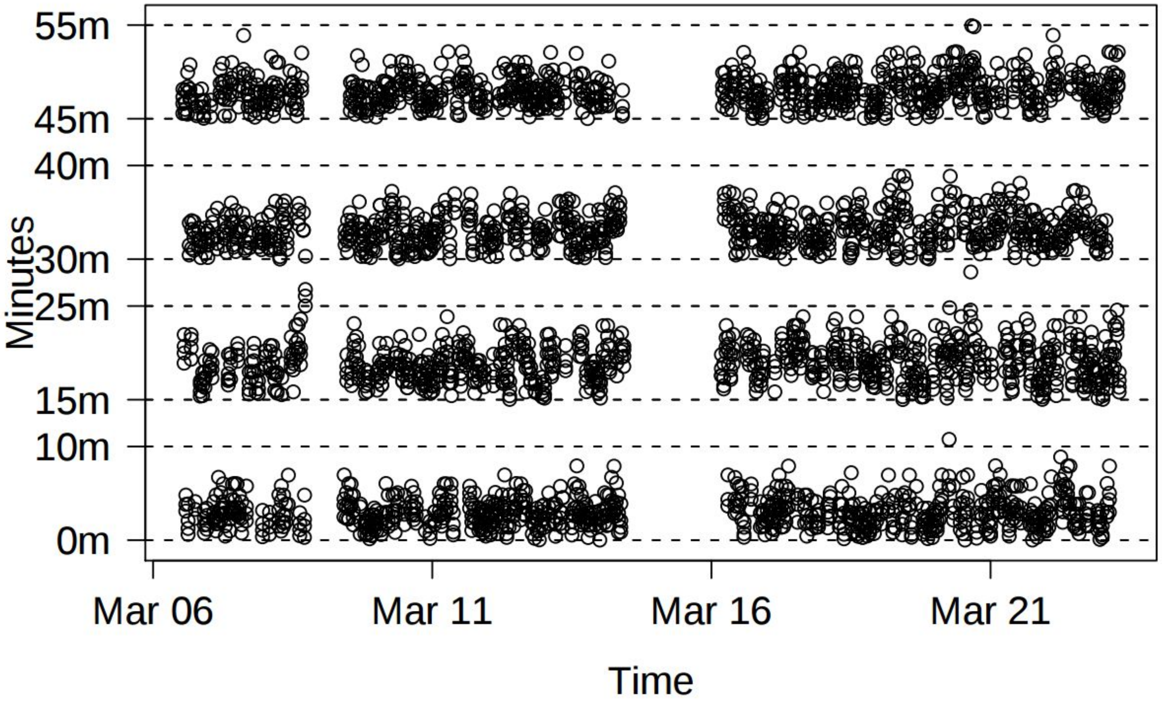
Physical infrastructure

- State leakage shows that probes are controlled by centralised entity
- Not clear how central entity controls probes
- Proxy network?
  - Geographically distributed set of proxy machines
- Off-path device in ISP's data centre?
  - Machines connected to switch mirror ports

Blocking is reliable, but fails predictably

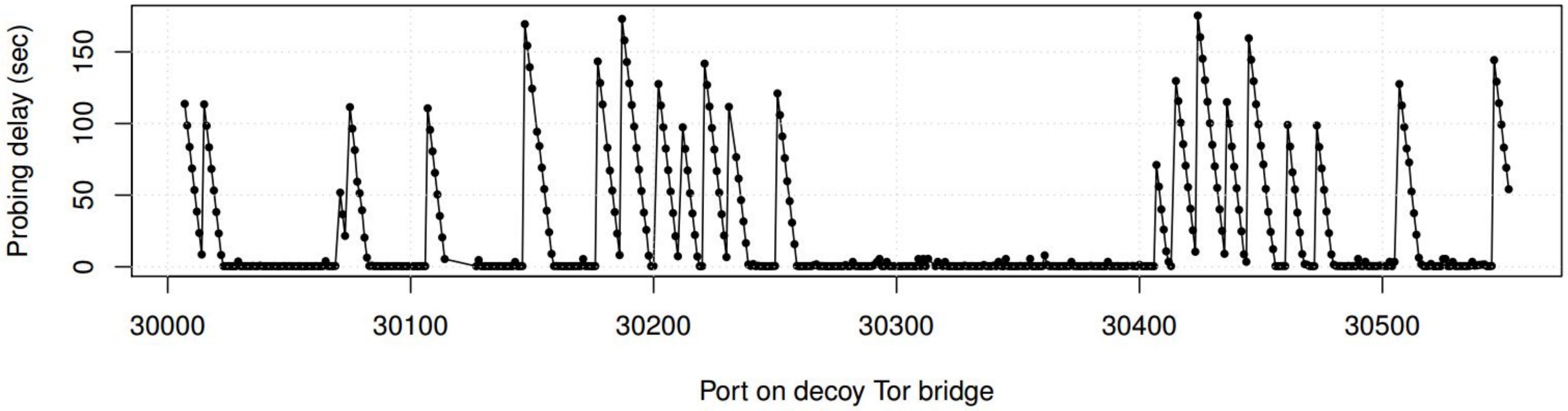


In 2012, probes were batch-processed



Today, probes are invoked in real-time

- Median arrival time of only 500 ms
- Odd, linearly-decreasing outliers



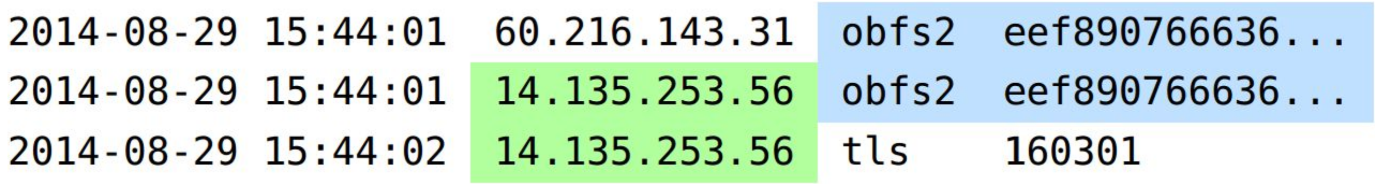
Blocked protocols

Protocols that are probed or blocked

- SSH
  - In 2011, not anymore?
- VPN
  - OpenVPN occasionally
  - SoftEther
- Tor
  - Vanilla Tor
  - obfs2 and obfs3
- AppSpot
  - To find GoAgent?
- TLS
- Anything else?

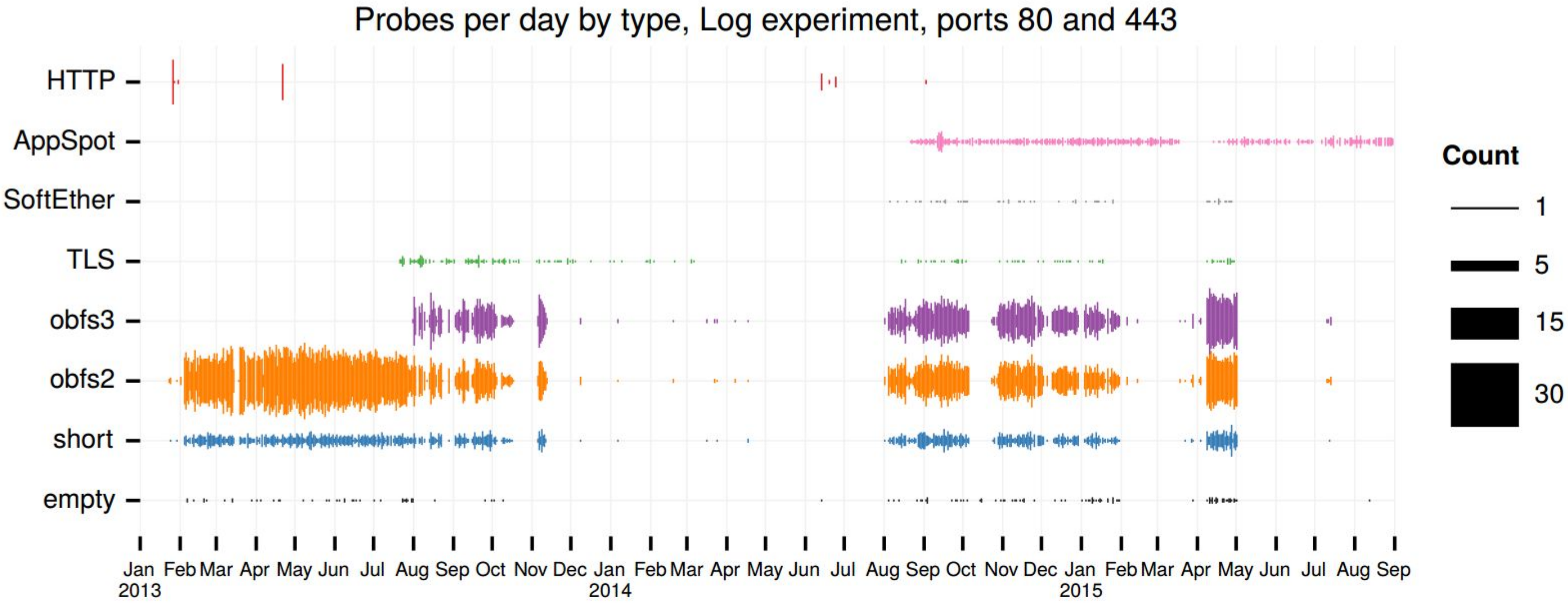
Oddities in obfs2 and obfs3 probing

- Tor probes don’t use reference implementations
  - obfs3 padding sent in one segment instead of two
- Probes sometimes send duplicate payload
  - State leakage?



```
2014-08-29 15:44:01  60.216.143.31  obfs2  eef890766636...
2014-08-29 15:44:01  14.135.253.56  obfs2  eef890766636...
2014-08-29 15:44:02  14.135.253.56  tls    160301
```

Probe type and frequency since 2013



Find your own probes

- SoftEther: `POST /vpnsvc/connect.cgi`
- AppSpot: `GET /twitter.com`
- `tcpdump ‘host 202.108.181.70’`
- More instructions on nymity.ch/active-probing

Trolling the GFW

Block list exhaustion

```
for ip_addr in “$ip_addrs”; do
  for port in $(seq 1 65535); do
    timeout 5 tor --usebridges 1 --bridge “$ip_addr:$port”
  done
done
```

One /24 network can add 16 million blocklist entries

File descriptor exhaustion

- Processes have OS-enforced file descriptor limit
  - Often 1,024, but configurable
  - Every new, open socket brings us closer to limit
- What's the limit for active probes?
- Attract many probes and don't ACK data, don't close socket
- Will GFW be unable to scan new bridges?

Make GFW block arbitrary addresses

- See VPN Gate's "innocent IP mixing"
  - See censorbib.nymity.ch/#Nobori2014a
- For a while, GFW blindly fetched and blocked IP addresses
- Add critical IP addresses to server list
  - Windows update servers
  - DNS root servers
  - Google infrastructure
- GFW operators soon started verifying addresses

VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls

Operational Systems Track

Daiyuu Nobori and Yasushi Shinjo
Department of Computer Science, University of Tsukuba, Japan

Abstract

VPN Gate is a public VPN relay service designed to achieve blocking resistance to censorship firewalls such as the Great Firewall (GFW) of China. To achieve such resistance, we organize many volunteers to provide a VPN relay service, with many changing IP addresses. To block VPN Gate with their firewalls, censorship authorities must find the IP addresses of all the volunteers. To prevent this, we adopted two techniques to improve blocking resistance. The first technique is to mix a number of innocent IP addresses into the relay server list provided to the public. The second technique is collaborative spy detection. The volunteer servers work together to create a list of spies, meaning the computers used by censorship authorities to probe the volunteer servers. Using this list, each volunteer server ignores packets from spies. We launched VPN Gate on March 8, 2013. By the end of August it had about 3,000 daily volunteers using 6,300 unique IP addresses to facilitate 464,000 VPN connections from users worldwide, including 45,000 connections and 9,000 unique IP addresses from China. At the time VPN Gate maintained about 70% of volunteer VPN servers as unblocked by the GFW.

1. Introduction

Some countries in the world have censorship firewalls operated by their governments to prohibit access to servers in foreign countries. For instance, the Great Firewall (GFW) of China blocks access to Twitter, Facebook, and YouTube. Internet users in countries subject to censorship often use overseas public relay servers to bypass censorship firewalls. Public proxies, VPN servers, and Tor nodes [7] are popular examples of such relay servers. Usually, the IP addresses of relay servers are publically available for user convenience. A censorship authority can easily block these relays, however, by adding the IP addresses to its firewall blocking list. Moreover, the Chinese authority, in particular, scans for unlisted Tor nodes and blocks them automatically [19]. Tor relays currently have no blocking resistance [12] against such scanning activities.

In this research, we have built a public VPN relay server system with blocking resistance to censorship firewalls such as the GFW. We call this system VPN Gate. To achieve blocking resistance, VPN Gate uses frequently changing IP addresses that are provided by volunteers. The central list server, called the *VPN Gate List Server*, manages a list of the IP addresses of all active VPN servers. We call this list the *Server List*. A user can get only part of the Server List and connect his/her PC to an active VPN server in the list. The user can then communicate with blocked Internet servers through the active VPN server. It is hard for a censorship authority to block all the active VPN servers in VPN Gate.

It is important for anti-censorship systems to achieve blocking resistance. We adopted two techniques for blocking resistance: innocent IP mixing and collaborative spy detection. In innocent IP mixing, we include a number of IP addresses, which are unrelated to VPN Gate, in the Server List. For instance, we include vitally important servers (e.g., Windows Update servers). This technique forces a censorship authority to remove innocent IP addresses from the Server List before adding addresses to the firewall blocking list. The second technique, collaborative spy detection, seeks probing activities from censorship authority's computers, called *spies*. In this technique all the volunteer VPN servers work together to create a source IP address list of spies, called the *Spy List*, and they ignore probing packets from spies. This technique makes the authority unable to distinguish between the IP addresses of active VPN servers and innocent IP addresses or those of inactive VPN servers.

The VPN Gate system consists of instances of the VPN Gate Server software, an optional application, the VPN Gate Client software, and a central List Server. Volunteers can easily install and execute VPN Gate Server. For instance, volunteers don't have to configure Network Address Translation (NAT) boxes to open TCP/UDP ports. Users can connect to VPN Gate Server with a Secure Sockets Layer (SSL)-VPN protocol by using VPN Gate Client. Users can also connect to a VPN server with the L2TP/IPsec, OpenVPN, and MS-SSTP protocols by using the built-in, OS-provided VPN clients on PCs and smartphones. As for the third piece of the system, our research group runs the VPN Gate List Server which accepts registration from volunteer servers, generates the Server List, and distributes it to users.

USENIX Association

11th USENIX Symposium on Networked Systems Design and Implementation 229

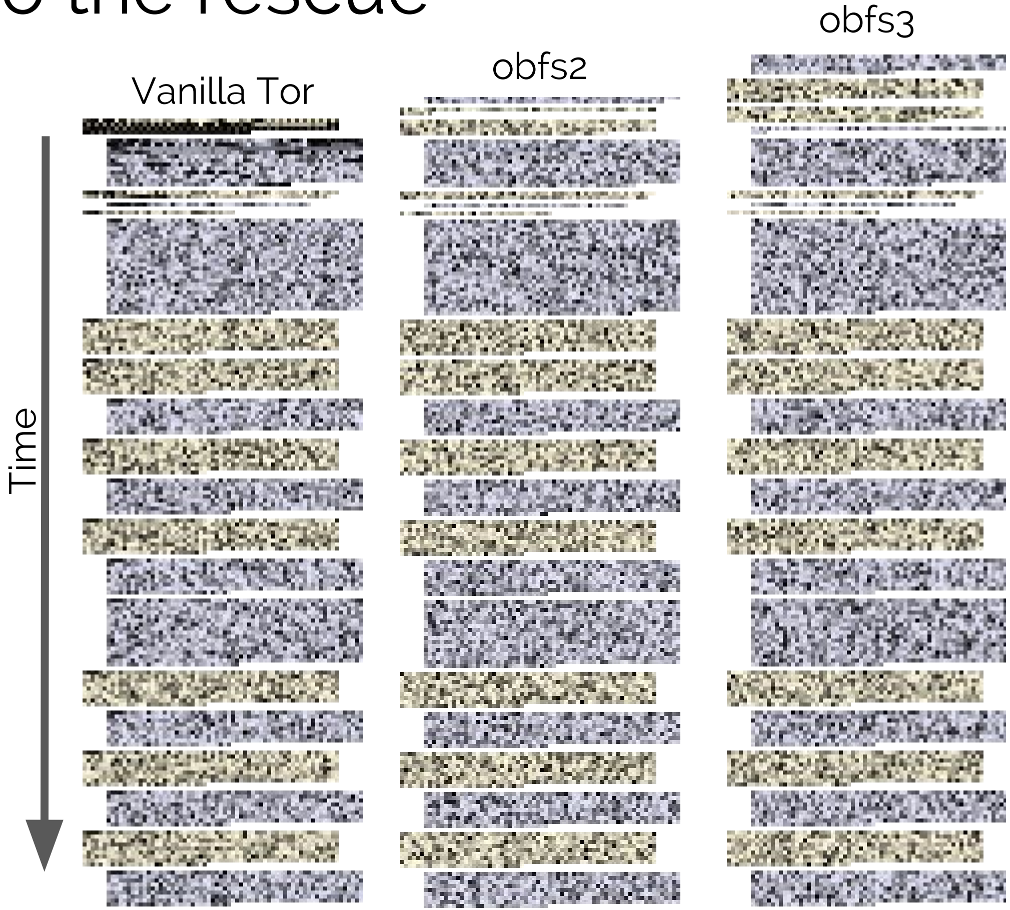
Circumvention

Problems in the GFW's DPI engine

- DPI engine must reassemble stream before pattern matching
- TCP stream often not reassembled
  - Server-side manipulation of TCP window size can "hide" signature
  - Exploited in brdgrd: gitweb.torproject.org/brdgrd.git/
- Ambiguities in TCP/IP parsing
  - See censorbib.nymity.ch/#Khattak2013a
- TCP/IP-based circumvention difficult to deploy
  - "Hey, how about you run this kernel module for me?"

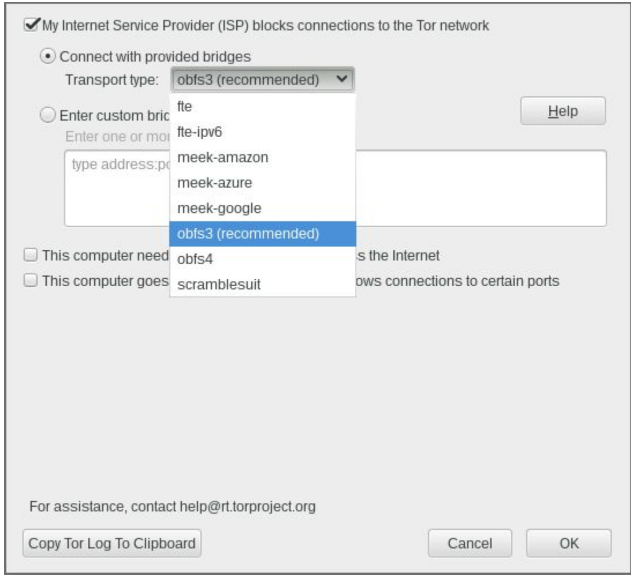
Pluggable transports to the rescue

- SOCKS interface on client
- Turn Tor into something else
  - Payload
  - Flow
- Several APIs
  - Python
  - Go
  - C



Pluggable transports that work in China

- **ScrambleSuit**
  - Flow shape polymorphic
  - Clients must prove knowledge of shared secret
- **obfs4**
  - Extends ScrambleSuit
  - Uses Elligator elliptic curve key agreement
- **meek**
  - Tunnels traffic over CDNs (Amazon, Azure, Google)
- **FTE**
  - Shapes ciphertext based on regular expressions
- More is in the making!
  - WebRTC-based transport



☑ My Internet Service Provider (ISP) blocks connections to the Tor network

● Connect with provided bridges

Transport type: obfs3 (recommended)

○ Enter custom bric

Enter one or mor

type address:pc

□ This computer need s the Internet

□ This computer goes ows connections to certain ports

For assistance, contact help@rt.torproject.org

Q&A

Roya Ensafi — rensafi@cs.princeton.edu — @_royaen_

David Fifield — david@bamsoftware.com

Philipp Winter — phw@nymity.ch — @__phw

Code, data, and paper: nymity.ch/active-probing/