Come As You Are:
Helping Unmodified Clients Bypass Censorship with Server-Side Evasion

Kevin Bock

George Hughey, Louis-Henri Merino, Tania Arya, Daniel Liscinsky, Regina Pogosian, Dave Levin
In-network censorship by nation-states
In-network censorship by nation-states
In-network censorship by nation-states
In-network censorship by nation-states
In-network censorship by nation-states
In-network censorship by nation-states
In-network censorship by nation-states
In-network censorship by nation-states

Deep packet inspection

Client → Deep packet inspection → Server
In-network censorship by nation-states

Deep packet inspection
In-network censorship by nation-states
In-network censorship by nation-states

Client
Spoofed tear-down packets
Server
In-network censorship by nation-states

Spoofed tear-down packets
In-network censorship by nation-states

Spoofed tear-down packets
In-network censorship by nation-states

Spoofed tear-down packets
In-network censorship by nation-states

Spoofed tear-down packets

Client

The server terminated

Server

The client terminated
In-network censorship by nation-states

Client

Spoofed tear-down packets

Server

The client terminated

Injecting tear-down packets

The server terminated
In-network censorship by nation-states

Spoofed tear-down packets

Injecting tear-down packets
Requires *per-flow state*
Censors necessarily *take shortcuts*
In-network censorship by nation-states

Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts
In-network censorship by nation-states

Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts
In-network censorship by nation-states

Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts
In-network censorship by nation-states

Client  \rightarrow  \text{Censor}  \rightarrow  \text{Server}

TTL=1

Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts
In-network censorship by nation-states

Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts
In-network censorship by nation-states

Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts
In-network censorship by nation-states

Injecting tear-down packets
Requires *per-flow state*
Censors necessarily *take shortcuts*
Evasion can take advantage of these shortcuts
Evasion has always involved the client
Evasion has always involved the client
Evasion has always involved the client

Installing software can pose risks to the user
Evasion has always involved the client

Installing software can pose risks to the user

Cannot help users who do not know they are censored
Ideally, servers could help
Server-side evasion

Censoring regime

Client

Server

Software
Server-side evasion

Potentially broadens reachability without *any* client-side deployment

Censoring regime

Clients

Server
Software
Server-side evasion “shouldn’t” work
Server-side evasion “shouldn’t” work
Server-side evasion “shouldn’t” work

All a server does before client is censored

Censored keyword
This paper

Server-side evasion is possible

- For every country and protocol we tested
  - Artifact-evaluated, open-source tool

New insights into how censors work

- GFW's resynchronization state
  - “Multibox Theory”
This paper

Server-side evasion is possible

- For every country and protocol we tested
- Artifact-evaluated, open-source tool

New insights into how censors work

- GFW's resynchronization state
- “Multibox Theory”
Geneva runs strictly at one side

Manipulates packets to and from the client
Manipulates packets to and from the client

- Duplicate
- Tamper
- Fragment
- Drop
Geneva
Genetic Evasion

Manipulates packets to and from the client

- Duplicate
- Tamper
- Fragment
- Drop

Alter or corrupt any TCP/IP header field

No semantic understanding of what the fields mean

Bock et al. CCS’19
Geneva
Genetic Evasion

Manipulates packets to and from the client

- Duplicate
- Tamper
- Fragment
- Drop

Fragment (IP) or Segment (TCP)

Alter or corrupt any TCP/IP header field

No semantic understanding of what the fields mean

Bock et al. CCS'19
Geneva
Genetic Evasion

Manipulates packets to and from the client

out : tcp.flags = A

Duplicate

Tamper
tcp.flags = R

Tamper
ip.ttl = 2
Genetic Evasion

Manipulates packets to and from the client

out:tcp.flags=A

Duplicate

Tamper

tcp.flags = R

Tamper

ip.ttl = 2

Bock et al. CCS'19
Genetic Evasion

Manipulates packets to and from the client

- out:tcp.flags = A
- Duplicate
- Tamper tcp.flags = R
- Tamper ip.ttl = 2

Match exact
Action in-order

Bock et al. CCS'19
Running a Strategy

Client

Duplicate

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

Server
Running a Strategy

Client

Duplicate

out: tcp.flags = A

Tamper

tcp.flags = R

Tamper

ip.ttl = 2

Server

Bock et al. CCS'19
Running a Strategy

Client

Server

Duplicate

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

out: tcp.flags = A
Running a Strategy

Client

Duplicate

out: tcp.flags = A

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

Server

Bock et al. CCS'19
Running a Strategy

Client

Server

TTL=8

TTL=2

Duplicate

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

out:tcp.flags=A

Bock et al. CCS'19
Running a Strategy

Client

Duplicate

timout:tcp.flags=A

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

TTL=2

Server

Bock et al. CCS'19
Running a Strategy

Client

Duplicate

out: tcp.flags = A

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

Server
This paper: Server-side Geneva

Modified Geneva to run server-side

Deployed against real world censors
Results against real censors

Diversity of protocols

HTTP  HTTPS  DNS  FTP  SMTP
Results against real censors

Diversity of protocols

HTTP  HTTPS  DNS  FTP  SMTP

Forbidden keywords & domains
Results against real censors

Diversity of protocols

HTTP  HTTPS  DNS  FTP  SMTP

Forbidden keywords & domains

xiazai@upup.info
Results against real censors

Diversity of censors

- Injects TCP RSTs
  - China

- Injects & blackholes
  - Iran

- Injects & blackholes
  - Kazakhstan

- Injects a block page
  - India

Diversity of protocols

- HTTP
- HTTPS
- DNS
- FTP
- SMTP

* Indicates a specific protocol censorship.
Server-side evasion “shouldn’t” work

All a server does before client is censored
A successful server-side evasion strategy
A successful server-side evasion strategy

TCP simultaneous open
A successful server-side evasion strategy

TCP simultaneous open

Client sends a SYN/ACK

Client

Server
A successful server-side evasion strategy

Censor de-synchronizes

TCP simultaneous open

Client sends a SYN/ACK
A successful server-side evasion strategy

Success rates:
- DNS: 89%
- FTP: 36%
- HTTP: 54%
- HTTPS: 55%
- SMTP: 70%
Server-side evasion strategies

China
8 strategies

Iran/India
1 strategy

Kazakhstan
3 strategies
Server-side evasion results

**NULL TCP Flags**

- Client
  - SYN
  - $\emptyset$ (no flags)
  - SYN/ACK
  - ACK
  - PSH/ACK (query)
  - ACK
  - PSH/ACK (response)

- Server

Success rates

HTTP 100%
Server-side evasion results

NULL TCP Flags

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN</td>
<td></td>
</tr>
<tr>
<td>Ø (no flags)</td>
<td></td>
</tr>
<tr>
<td>SYN/ACK</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td></td>
</tr>
<tr>
<td>PSH/ACK (query)</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td></td>
</tr>
<tr>
<td>PSH/ACK (response)</td>
<td></td>
</tr>
</tbody>
</table>

Success rates:

HTTP 100%
Server-side evasion results

NULL TCP Flags

Client

SYN

Ø

(no flags)

SYN/ACK

ACK

PSH/ACK (query)

ACK

PSH/ACK (response)

Server

Server sends a packet with no TCP flags set

Success rates

HTTP 100%
Server-side evasion results

NULL TCP Flags

Client
- SYN
- SYN/ACK
- ACK
- PSH/ACK (query)
- ACK
- PSH/ACK (response)

Server
- Server sends a packet with no TCP flags set
- Censor can’t handle unexpected flags

Success rates
- HTTP 100%
Server-side evasion results

Double benign-GETs

- Client
  - SYN
  - SYN/ACK (benign GET)
  - SYN/ACK (benign GET)
  - ACK
  - ACK
  - ACK
  - PSH/ACK (query)
  - ACK
  - PSH/ACK (response)

- Server
Server-side evasion results

Double benign-GETs

Server sends uncensored GETs inside two SYN/ACKs
Server-side evasion results

Double benign-GETs

Censor confuses connection direction

Server sends uncensored GETs inside two SYN/ACKs
Server-side evasion results

Double benign-GETs

Censor confuses connection direction

Server sends uncensored GETs inside two SYN/ACKs

Success rates
HTTP 100%
Server-side evasion results

Double benign-GETs

Client

Server

SYN

SYN/ACK
(benign GET)

SYN/ACK
(benign GET)

ACK

ACK

PSH/ACK
(query)

ACK

PSH/ACK
(response)

Success rates

HTTP 100%
Server-side evasion strategies

- China: 8 strategies
- Iran/India: 1 strategy
- Kazakhstan: 3 strategies
Server-side evasion strategies

China
8 strategies

Iran/India
1 strategy

Kazakhstan
3 strategies

None of these require *any* client-side deployment
Come as you are

Windows XP
Windows 7
Windows 8.1
Windows 10
Server 2003
Server 2008
Server 2013
Server 2018

OS X 10.14
OS X 10.15

Android 10

iOS 13.3

Centos 6
Centos 7

Ubuntu 12.04
Ubuntu 14.04
Ubuntu 16.04
Ubuntu 18.04
This paper

Server-side evasion is possible

For every country and protocol we tested

Artifact-evaluated, open-source tool

New insights into how censors work

GFW's resynchronization state

"Multibox Theory"
This paper

Server-side evasion is possible

- For every country and protocol we tested
- Artifact-evaluated, open-source tool

New insights into how censors work

- GFW’s resynchronization state
- “Multibox Theory”
Censoring middleboxes tolerant to packet loss
If middleboxes misses a packet

Censor can resynchronize its state

Censoring middleboxes tolerant to packet loss
Resynchronization State

Simultaneous-open-based desynchronization

Payload from server triggers resynchronization
Resynchronization State

Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client

Payload from server triggers resynchronization

Client
- SYN
- SYN
- SYN
- SYN/ACK
- ACK
- ACK
- PSH/ACK (query)
- ACK
- PSH/ACK (response)

Server
- SYN
- SYN
- SYN
- SYN/ACK
- ACK
- ACK
- PSH/ACK (query)
- ACK
- PSH/ACK (response)
Resynchronization State

Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client

...but does not properly increment ISN

Payload from server triggers resynchronization
Resynchronization State

Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client

...but does not properly increment ISN

Off-by-1 bug in the Great Firewall

Payload from server triggers resynchronization
Resynchronization State

Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client
...but does not properly increment ISN

Payload from server triggers resynchronization

Off-by-1 bug in the Great Firewall

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>89%</td>
</tr>
<tr>
<td>FTP</td>
<td>36%</td>
</tr>
<tr>
<td>HTTP</td>
<td>54%</td>
</tr>
<tr>
<td>HTTPS</td>
<td>55%</td>
</tr>
<tr>
<td>SMTP</td>
<td>70%</td>
</tr>
</tbody>
</table>
Resynchronization State

GFW resynchronizes differently depending on protocol

GFW resynchronizes on the next:

- **FTP**: Client packet if SYN+ACK has a bad ack number
- **All but HTTPS**: Client packet if server sends a RST
- **All protocols**: ACK packet if server sends non-SYN+ACK with a payload
## New Model for Chinese Censorship

All of the server-side strategies operate strictly during the TCP 3-way handshake.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>DNS</th>
<th>FTP</th>
<th>HTTP</th>
<th>HTTPS</th>
<th>SMTP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>China</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>2%</td>
<td>3%</td>
<td>3%</td>
<td>3%</td>
<td>26%</td>
</tr>
<tr>
<td>Sim. Open, Injected RST</td>
<td>89%</td>
<td>52%</td>
<td>54%</td>
<td>14%</td>
<td>70%</td>
</tr>
<tr>
<td>Sim. Open, Injected Load</td>
<td>83%</td>
<td>36%</td>
<td>54%</td>
<td>55%</td>
<td>59%</td>
</tr>
<tr>
<td>Corrupt ACK, Sim. Open</td>
<td>26%</td>
<td>65%</td>
<td>4%</td>
<td>4%</td>
<td>23%</td>
</tr>
<tr>
<td>Corrupt ACK Alone</td>
<td>7%</td>
<td>33%</td>
<td>5%</td>
<td>5%</td>
<td>22%</td>
</tr>
<tr>
<td>Corrupt ACK, Injected Load</td>
<td>15%</td>
<td>97%</td>
<td>4%</td>
<td>3%</td>
<td>25%</td>
</tr>
<tr>
<td>Injected Load, Induced RST</td>
<td>82%</td>
<td>55%</td>
<td>52%</td>
<td>54%</td>
<td>55%</td>
</tr>
<tr>
<td>Injected RST, Induced RST</td>
<td>83%</td>
<td>85%</td>
<td>54%</td>
<td>4%</td>
<td>66%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td>3%</td>
<td>47%</td>
<td>2%</td>
<td>3%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>India</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>100%</td>
<td>100%</td>
<td>2%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td></td>
<td>100%</td>
<td></td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td><strong>Iran</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>100%</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td></td>
<td>100%</td>
<td></td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td><strong>Kazakhstan</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>100%</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td></td>
<td>100%</td>
<td></td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Triple Load</td>
<td></td>
<td></td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double GET</td>
<td></td>
<td></td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null Flags</td>
<td></td>
<td></td>
<td>100%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### New Model for Chinese Censorship

All of the server-side strategies operate strictly during the TCP 3-way handshake. So why are different applications affected differently in China?

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Success Rates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DNS</td>
</tr>
<tr>
<td><strong>China</strong></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>2%</td>
</tr>
<tr>
<td>Sim. Open, Injected RST</td>
<td>89%</td>
</tr>
<tr>
<td>Sim. Open, Injected Load</td>
<td>83%</td>
</tr>
<tr>
<td>Corrupt ACK, Sim. Open</td>
<td>26%</td>
</tr>
<tr>
<td>Corrupt ACK Alone</td>
<td>7%</td>
</tr>
<tr>
<td>Corrupt ACK, Injected Load</td>
<td>15%</td>
</tr>
<tr>
<td>Injected Load, Induced RST</td>
<td>82%</td>
</tr>
<tr>
<td>Injected RST, Induced RST</td>
<td>83%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td>3%</td>
</tr>
<tr>
<td><strong>India</strong></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>100%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td>-</td>
</tr>
<tr>
<td><strong>Iran</strong></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>100%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td>-</td>
</tr>
<tr>
<td><strong>Kazakhstan</strong></td>
<td></td>
</tr>
<tr>
<td>No evasion</td>
<td>100%</td>
</tr>
<tr>
<td>TCP Window Reduction</td>
<td>-</td>
</tr>
<tr>
<td>Triple Load</td>
<td>-</td>
</tr>
<tr>
<td>Double GET</td>
<td>-</td>
</tr>
<tr>
<td>Null Flags</td>
<td>-</td>
</tr>
</tbody>
</table>
New Model for Chinese Censorship

DNS  HTTP  FTP
TCP
IP
New Model for Chinese Censorship

Sane

<table>
<thead>
<tr>
<th>DNS</th>
<th>HTTP</th>
<th>FTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

 Apparently what’s happening

<table>
<thead>
<tr>
<th>DNS</th>
<th>HTTP</th>
<th>FTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
New Model for Chinese Censorship

Sane

DNS HTTP FTP
TCP
IP

Apparently what’s happening

DNS HTTP FTP
TCP TCP TCP
IP IP IP
New Model for Chinese Censorship

Results suggest GFW is running multiple censoring middleboxes in parallel
Multi-box theory

Client → GFW → Server
Multi-box theory

GFW

DNS HTTP

FTP HTTPS

Client

Server
How does the censor know which one to apply to a connection?
Multi-box theory

GFW

DNS  HTTP
FTP  HTTPS

Client ———————— Server

Not port number
Censors effectively on any port
Multi-box theory

GFW

DNS  HTTP

FTP  HTTPS

Client

Not port number

Censors effectively on any port

Server
Multi-box theory

GFW

DNS HTTP

FTP HTTPS

Not port number
Censors effectively on any port
Multi-box theory

Applies protocol fingerprinting
Multi-box theory

GFW

DNS  HTTP

FTP  HTTPS

Applies protocol fingerprinting
Multi-box theory

GFW

DNS  HTTP

FTP  HTTPS

Applies protocol fingerprinting
Multi-box theory

GFW

Applies protocol fingerprinting
Multi-box theory

GFW

Client

Not mine

DNS

HTTP

Mine! Forbidden

FTP

HTTPS

Server

Not mine

Applies protocol fingerprinting
Where are these middleboxes?

Used TTL-limited probes

Client → DNS HTTP FTP HTTPS → Server
Where are these middleboxes?

Used TTL-limited probes

Co-located at the network level
Responsive to new censorship events

February 2020: Iran launched a new system: a protocol filter
Responsive to new censorship events

February 2020: Iran launched a new system: a protocol filter

Censors connections that do not match protocol fingerprints
Responsive to new censorship events

**February 2020:** Iran launched a new system: a **protocol filter**

Censors connections that do not match **protocol fingerprints**

Those that do match are then subjected to standard censorship
Responsive to new censorship events

February 2020: Iran launched a new system: a protocol filter

Censors connections that do not match protocol fingerprints

Those that do match are then subjected to standard censorship

Geneva discovered 4 strategies to evade Iran’s filter
Responsive to new censorship events

July 29th 2020: China begins censoring the use of Encrypted SNI
Responsive to new censorship events

July 29th 2020: China begins censoring the use of Encrypted SNI

Geneva discovered 6 strategies to evade ESNI censorship
Real world deployment

Assist in **bootstrapping connections**

Harden existing evasion protocols

Client

Server

Real world deployment

[Image of a network diagram with a client, a web browser, a police officer, and a server, indicating connections and security measures.]
Middleboxes create new possibilities

The good

They make server-side evasion possible!
Middleboxes create new possibilities

The good
They make server-side evasion possible!

The ugly
They have exploitable bugs and assumptions
Middleboxes create new possibilities

The good
They make server-side evasion possible!

The very bad
Middleboxes can be weaponized

The ugly
They have exploitable bugs and assumptions

TCP-based reflected amplification

USENIX Security '21
Middleboxes create new possibilities

The good
They make server-side evasion possible!

The very bad
Middleboxes can be weaponized

The ugly
They have exploitable bugs and assumptions

TCP-based reflected amplification

Automated tools like Geneva are important in understanding what middleboxes enable
Server-side Evasion

Geneva

Genetic Evasion

Server-side evasion is possible

New insights into censors

Code is open source

Real world deployment

Geneva code and website [geneva.cs.umd.edu]