Geneva: Evolving Censorship Evasion Strategies

Kevin Bock   George Hughey

Louis-Henri Merino, Tania Arya, Daniel Liscinsky, Regina Pogosian
Xiao Qiang, Dave Levin

University of Maryland   Berkeley School of Information
Why study censorship evasion?

We have Tor!

We have VPNs!

We have secure HTTPS!
Why study censorship evasion?

We have Tor

We have VPNs

We have secure HTTPS
Why study censorship evasion?

We have Tor

We have VPNs

We have secure HTTPS
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
In-network censorship by nation-states

Client ➔ Server

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
In-network censorship by nation-states

Spoofed tear-down packets

The server terminated

The client terminated
In-network censorship by nation-states

Spoofed tear-down packets

Requires \textit{per-flow state}

The client terminated

The server terminated

Requires \textit{per-flow state}
In-network censorship by nation-states

Spoofed tear-down packets

The client terminated

Client

The server terminated

Server

Requires *per-flow state*

Censors necessarily *take shortcuts*
In-network censorship by nation-states

Requires *per-flow state*

Censors necessarily *take shortcuts*

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

Client

TTL=2

Server

Requires *per-flow state*

Censors necessarily *take shortcuts*

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

Client -> [TTL=1] Censor -> [TTL=1] Server

Requires *per-flow state*

Censors necessarily *take shortcuts*

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

Client → Server

TTL=1

Requires *per-flow state*

Censors necessarily *take shortcuts*

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

Requires *per-flow state*

Censors necessarily *take shortcuts*

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

Requires *per-flow state*

Censors necessarily *take shortcuts*

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

The client terminated

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts
Censorship evasion research

Hypothesize → Measure → Evade

- Hypothesize
- Measure
- Evade
Censorship evasion research

1. Understand how censors operate

Hypothesize -> Measure -> Evade
Censorship evasion research

1. *Understand* how censors operate
2. *Apply insight* to create evasion strategies
Censorship evasion research

1. Understand how censors operate
2. Apply insight to create evasion strategies

Largely manual efforts give censors the advantage
Censorship evasion research

1. Understand how censors operate

2. Apply insight to create evasion strategies

Largely manual efforts give censors the advantage

Our work gives evasion the advantage
AI-assisted censorship evasion research

Evade ➞ Hypothesize ➞ Measure

- Evade
- Hypothesize
- Measure
AI-assisted censorship evasion research

1. Use AI to automatically learn new evasion strategies
AI-assisted censorship evasion research

1. Use AI to automatically learn new evasion strategies

2. Use the strategies the AI finds to understand how the censor works
1. Use AI to **automatically learn** new evasion strategies

2. Use the strategies the AI finds to **understand** how the censor works
Geneva
Genetic Evasion

Building Blocks
A
T
C
G

Composition
DNA

Mutation
DNA → DNA

Fitness
Evolution
Geneva runs strictly at the client.

Manipulates packets to and from the client.
Geneva
Genetic Evasion

Building Blocks

Manipulates packets to and from the client
Geneva
Genetic Evasion

Building Blocks

Manipulates packets to and from the client

- Bit manipulation
  Versatile but inefficient
Geneva
Genetic Evasion

Building Blocks

Manipulates packets to and from the client

- Bit manipulation: Versatile but inefficient
- Known strategies: Efficient but limited
Geneva
Genetic Evasion

Building Blocks

Manipulates packets to and from the client

- Duplicate
- Tamper
- Fragment
- Drop
Geneva
Genetic Evasion

Building Blocks

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
Geneva
Genetic Evasion

Building Blocks

Manipulates packets to and from the client

- Duplicate
- Tamper
- Fragment (IP) or Segment (TCP)
- Drop

Alter or corrupt any TCP/IP header field

No semantic understanding of what the fields mean
Geneva
Genetic Evasion

Building Blocks
Actions manipulate individual packets
- Duplicate
- Tamper
- Fragment
- Drop

Composition

Mutation

Fitness

Geneva
Genetic Evasion

Building Blocks
Actions manipulate individual packets

- Duplicate
- Tamper
- Fragment
- Drop

Composition

Mutation

Fitness

Actions manipulate individual packets
Geneva
Genetic Evasion

Composition

out:tcp.flags=A

Duplicate

Tamper
tcp.flags = R

Tamper
ip.ttl = 2
Running a Strategy

Composition

Client

Duplicate

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

Server
Running a Strategy

Composition

Client → Duplicate (out:tcp.flags=A)

Duplicate → Tamper tcp.flags = R
Duplicate → Tamper ip.ttl = 2

Server
Running a Strategy

Composition

Client

Server

Duplicate

Tamper tcp.flags = R

Tamper ip.ttl = 2

out:tcp.flags=A
Running a Strategy

Composition

Client

Duplicate

Tamper
tcp.flags = R

Tamper
ip.ttl = 2

Server
Running a Strategy

Composition

Client -> Duplicate

Tamper
   tcp.flags = R

Tamper
   ip.ttl = 2

Server
Running a Strategy

Composition

Client <<< Duplicate <<< Tamper
tcp.flags = R
ip.ttl = 2

TTL=8

TTL=2

Server

TTL=8

TTL=2
Running a Strategy

Composition

Client
- Duplicate
  - Tamper (tcp.flags = R)
  - Tamper (ip.ttl = 2)

TTL=2

Server
Geneva
Genetic Evasion

Building Blocks
Actions manipulate individual packets
- Duplicate
- Tamper
- Fragment
- Drop

Composition
Actions compose to form trees
- out: tcp.flags = A
- Duplicate
- Tamper tcp.flags = R
- Tamper ip.ttl = 2

Mutation

Fitness
Geneva
Genetic Evasion

Building Blocks
Actions manipulate individual packets
- Duplicate
- Tamper
- Fragment
- Drop

Composition
Actions compose to form trees
- out: tcp.flags = A
  - Duplicate
    - Tamper tcp.flags = R
    - Tamper ip.ttl = 2

Mutation

Fitness
Geneva
Genetic Evasion

**Building Blocks**
Actions manipulate individual packets
- Duplicate
- Tamper
- Fragment
- Drop

**Composition**
Actions compose to form trees
- out:tcp.flags = A
- Duplicate
- Tamper tcp.flags = R
- Tamper ip.ttl = 2

**Mutation**
Randomly alter types, values, and trees

**Fitness**
Geneva
Genetic Evasion

Building Blocks
Actions manipulate individual packets
- Duplicate
- Tamper
- Fragment
- Drop

Composition
Actions compose to form trees
- out: tcp.flags=A
- Duplicate
- Tamper tcp.flags = R
- Tamper ip.ttl = 2

Mutation
Randomly alter types, values, and trees

Fitness
Fitness

Which *individuals* should survive to the next *generation*?
Fitness

Which **individuals** should survive to the next **generation**?
Which *individuals* should survive to the next *generation*?
Which **individuals** should survive to the next **generation**?

- Not triggering on any packets
- Breaking the TCP connection
- Successfully obtaining forbidden content
- Conciseness
Geneva
Genetic Evasion

Building Blocks
Actions manipulate individual packets
- Duplicate
- Tamper
- Fragment
- Drop

Composition
Actions compose to form trees
- out:tcp.flags = A
- Duplicate
- Tamper tcp.flags = R
- Tamper ip.ttl = 2

Mutation
Randomly alter types, values, and trees

Fitness
Goal: Fewest actions needed to succeed
- No trigger
- Break TCP
- Successful
- Concise
Geneva’s results

Real censor experiments
China  India  Kazakhstan
Geneva’s results
Real censor experiments

China

India

Kazakhstan
Geneva’s results
Real censor experiments

Injects TCP RSTs

China
India
Kazakhstan

FTP
DNS
SMTP
HTTP
HTTPS
Geneva’s results
Real censor experiments

FTP
DNS
SMTP
HTTP
HTTPS

Injects TCP RSTs

Injects a block page

China
India
Kazakhstan
Geneva’s results
Real censor experiments

FTP
DNS
SMTP
HTTP
HTTPS
Injects TCP RSTs
China

HTTP
Injects a block page
India

HTTP
Injects & blackholes
Kazakhstan
Geneva’s results
Real censor experiments

China
India
Kazakhstan
Geneva's results
Real censor experiments

6 Species
13 Sub-species
36 Variants

China
India
Kazakhstan
Geneva’s results
Real censor experiments

6 Species
13 Sub-species
36 Variants

The underlying bug
How Geneva exploits it
Functionally distinct

China
India
Kazakhstan
Geneva’s results
Real censor experiments

6 Species ——— The underlying bug

13 Sub-species ——— How Geneva exploits it

36 Variants ——— Functionally distinct

31

6

13

China

India

Kazakhstan

Species
Sub-species
Variants
Trick the censor into thinking the client is the server

Turnaround species

tcp.flags = S

Duplicate

Tamper
tcp.flags = SA
Turnaround species

Trick the censor into thinking the client is the server

Segmentation species

Segment the request
Turnaround species

out:tcp.flags=S

Duplicate

Tamper
tcp.flags = SA

Segmentation species

out:tcp.flags=PA

Fragment
tcp:8:inorder

Fragment
tcp:4:inorder

GET /?search=ultrasurf

Trick the censor into thinking
the client is the server

Segment the request
Trick the censor into thinking the client is the server

Segmentation species

Segment the request
Trick the censor into thinking the client is the server

Segmentation species

- out:tcp.flags=PA
- Fragment
  - tcp:8:inorder
- Fragment
  - tcp:4:inorder
- GET /?se=ultrasurf
  - 8
- arch
  - 4
- Remainder

Segment the request, but *not the keyword*
Turnaround species

out:tcp.flags=S

Duplicate

Tamper
tcp.flags = SA

Segmentation species

out:tcp.flags=PA

Fragment
tcp:8:inorder

Fragment
tcp:4:inorder

GET /?se=ultrasurf

 Trick the censor into thinking the client is the server

 Segment the request, but not the keyword
Turnaround species

out:tcp.flags=S

Duplicate

Tamper
tcp.flags = SA

Segmentation species

out:tcp.flags=PA

Fragment
tcp:8:inorder

Fragment
tcp:4:inorder

GET /?se=ultrasurf

Segment the request, but not the keyword

Trick the censor into thinking the client is the server
July 2019
Kazakhstan launched an HTTPS man-in-the-middle attack that lasted several weeks.
Geneva is fast

July 2019
Kazakhstan launched an HTTPS man-in-the-middle attack that lasted several weeks

Within 1 hour
Geneva found strategies to circumvent it
Censoring regime

Client

Geneva

Server
Server-side evasion

Censoring regime

Client

Server

Geneva
Server-side evasion

Censoring regime

Potentially broadens reachability without *any* client-side deployment
Server-side evasion “shouldn’t” work
Server-side evasion “shouldn’t” work

Censored keyword

SYN
SYN/ACK
ACK

PSH/ACK (query)

ACK
PSH/ACK (response)
Server-side evasion “shouldn’t” work

All a server does before client is censored

Censored keyword

Client

SYN

SYN/ACK

ACK

PSH/ACK

Server

(query)

PSH/ACK

(response)

(response)
Server-side evasion “shouldn’t” work

All a server does before client is censored

Censored keyword

Fortunately, the AI doesn’t know it “shouldn’t” work
Server-side evasion “shouldn’t” work

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

Server-side results

China

8 strategies
Server-side evasion “shouldn’t” work
Server-side results

China
8 strategies

India
1 strategy
Server-side evasion “shouldn’t” work

Server-side results

China
8 strategies

India
1 strategy

Kazakhstan
3 strategies
Server-side evasion “shouldn’t” work

Server-side results

China
8 strategies

India
1 strategy

Kazakhstan
3 strategies

None of these require any client-side deployment
Server-side evasion “shouldn’t” work

All a server does before client is censored

Censored keyword

Client

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

Server

response

Client

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

Server

All a server does before client is censored

Censored keyword
Simultaneous Open-based Desynchronization

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

Server
Simultaneous Open-based Desynchronization

TCP simultaneous open
Simultaneous Open-based Desynchronization

TCP simultaneous open

Client sends a SYN/ACK
Simultaneous Open-based Desynchronization

TCP simultaneous open

Client sends a SYN/ACK

Censor de-synchronizes
Double-benign GETs

Client

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

Server
Double-benign GETs

Server sends uncensored GETs inside two SYN/ACKs
Double-benign GETs

Censor confuses connection direction

Server sends uncensored GETs inside two SYN/ACKs
Automating the arms race

AI has the potential to fast-forward the arms race for both sides
Automating the arms race

AI has the potential to fast-forward the arms race for both sides

- Bugs in implementation: Easy for censors to fix the low-hanging fruit
- Gaps in logic: Harder for censors to fix systemic issues
Automating the arms race

AI has the potential to fast-forward the arms race for both sides.

- Bugs in implementation: Easy for censors to fix the low-hanging fruit
- Gaps in logic: Harder for censors to fix systemic issues

What is the logical conclusion of the arms race?
Automatically learning how to evade censorship

Geneva
Genetic Evasion

Server-side evasion
Finds strategies quickly
Dozens of strategies
Evasion advantage

Geneva code and website: geneva.cs.umd.edu