Geneva

Learning Nation-State Censorship with Genetic Algorithms

Kevin Bock, George Hughey, Dave Levin, *Xiao Qiang

Breakerspace undergrads: Tania Arya, Daniel Liscinsky, Louis-Henri Merino, Regina Pogosian

University of Maryland ***UC Berkeley**

Internet-Scale Censorship

- Deployed mostly *on-path* (man-on-the-side), not *in-path* (man-in-the-middle)
- Censor stores per-connection state until it believes the connection is closed
- **Client-side evasion**: Generate packets that make the censor's state inconsistent
  INTANG (IMC 2017), lib-erate (IMC 2017)

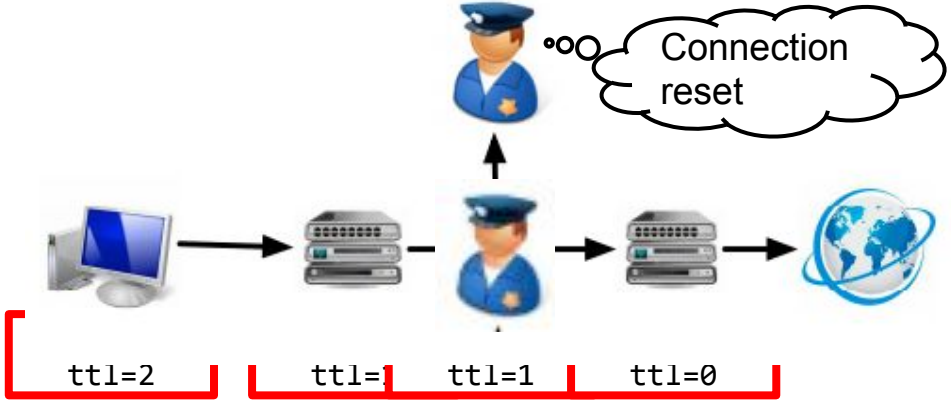
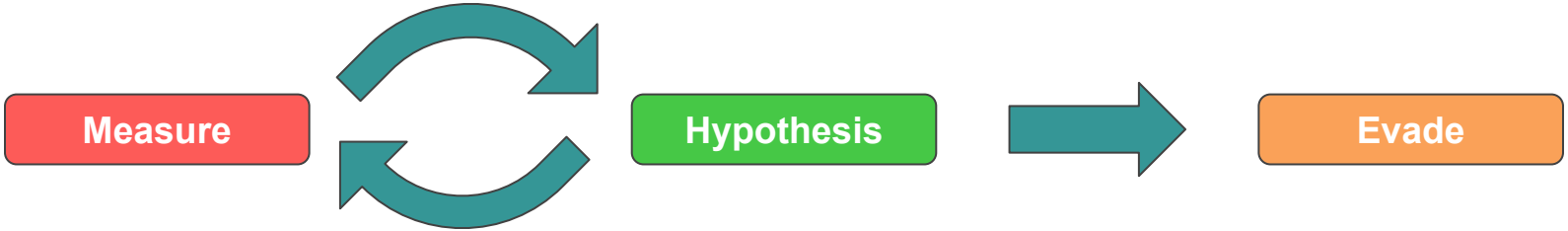


Image credit: Your State is Not Mine

Measuring Censorship

- Understanding censorship has historically been a prerequisite to evading censorship



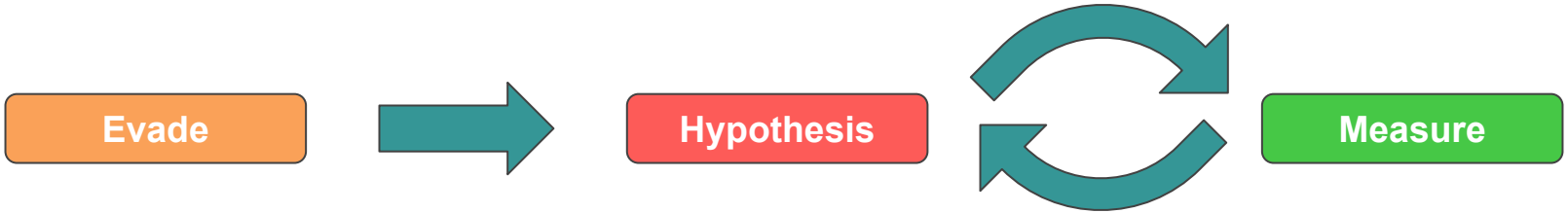
Cat/mouse game has historically **favored the censor**

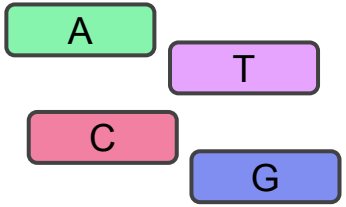
How can we break this manual evade/detect cycle?

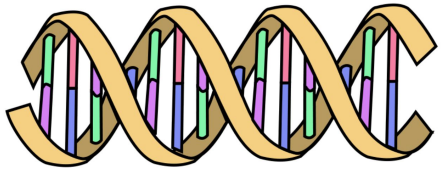
Breaking the Cycle

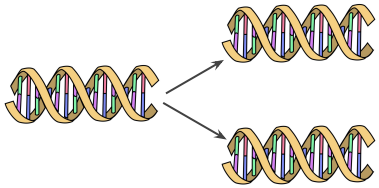
- Idea: develop an AI to **adaptively probe** how to defeat the censor
  - Geneva - ***GEN***etic ***EVA***sion

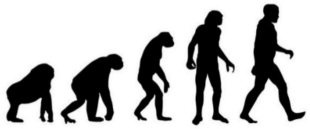
- Runs exclusively on the client side by manipulating inbound and outbound packets







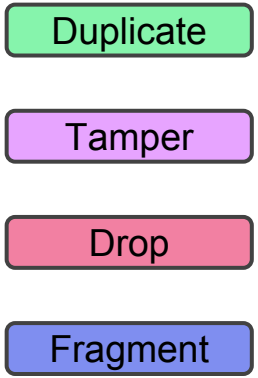




Building Blocks

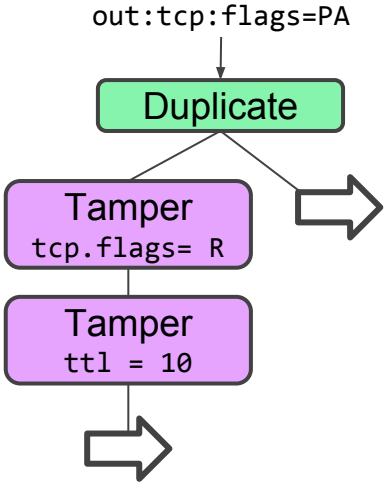
Triggers: Packet filters

Actions: Packet manipulators



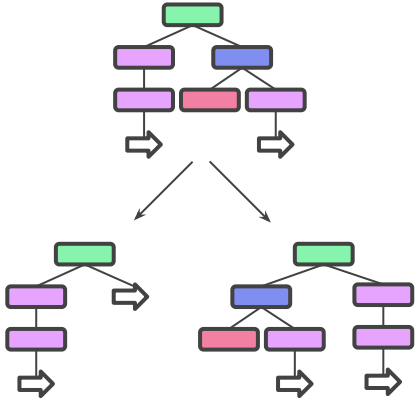
Composition

Action forests



Mutation

Randomly alter type and values of manipulation



Fitness

Progress through an HTTP GET

Strategy size

Goal: Find the smallest necessary set of actions to evade censorship

In-lab evaluation

Strategy	Species	Prior Work	Found?
TCB Creation w/ low TTL (TCP1)	TCB Creation	[20, 37]	Yes
TCB Creation w/ corrupt checksum	TCB Creation	[37]	Yes
Improved TCB Creation and Resync/Desync	TCB Creation	[37]	Yes
TCB Teardown w/ RST and low TTL (TCP6a)	TCB Teardown	[20, 37, 30]	Yes
TCB Teardown w/ RST and corrupt checksum	TCB Teardown	[37, 30]	Yes
TCB Teardown w/ RST and invalid timestamp	TCB Teardown	[37]	Yes
TCB Teardown w/ RST and invalid MD5 Header	TCB Teardown	[37]	Yes
TCB Teardown w/ RST/ACK and corrupt checksum (TCP6a)	TCB Teardown	[37]	Yes
TCB Teardown w/ RST/ACK and low TTL	TCB Teardown	[20, 37, 30]	Yes
TCB Teardown w/ RST/ACK and invalid timestamp	TCB Teardown	[37]	Yes
TCB Teardown w/ RST/ACK and invalid MD5 Header	TCB Teardown	[37]	Yes
TCB Teardown w/ FIN and low TTL (TCP6b)	TCB Teardown	[37, 20]	Yes
TCB Teardown w/ FIN and corrupt checksum	TCB Teardown	[37]	Yes
Improved TCB Teardown	TCB Teardown	[37]	Yes
TCB Teardown and TCB Reversal	TCB Teardown	[37]	Yes
State Exhaustion (send > 1KB of traffic) (TCP9)	State Exhaustion	[20]	No
Classification Flushing (TCP10) - Delay	State Exhaustion	[20, 30]	No
GET with > 1 space between method and URI (HTTP1)	HTTP Incompleteness	[20]	No
GET with keyword at location > 2048 (HTTP2)	HTTP Incompleteness	[20]	No
GET with keyword in 2nd of multiple requests in single segment (HTTP3)	HTTP Incompleteness	[20]	No
GET with URL encoded (except %-encoding) (HTTP4)	HTTP Incompleteness	[20]	No
TCP Segmentation reassembly out of order data	Reassembly	[30, 37]	Yes
Overlapping fragments (IP2)	Reassembly	[20, 37]	Yes
Overlapping segments (TCP5)	Reassembly	[20, 37]	Yes
Reassembly in-order data w/ low TTL	Reassembly	[37]	Yes
Reassembly in-order data w/ corrupt ACK	Reassembly	[37]	Yes
Reassembly in-order data w/ corrupt checksum	Reassembly	[37]	Yes
Reassembly in-order data w/ no TCP flags	Reassembly	[37]	Yes
Reassembly out-of-order data w/ IP fragments	Reassembly	[37]	Yes
Reassembly out-of-order data w/ TCP segments	Reassembly	[37]	Yes
Improved In-order data overlapping	Reassembly	[37]	Yes
Payload splitting	Reassembly	[30]	Yes
Payload reordering	Reassembly	[30]	Yes
Inert Packet Insertion w/ corrupt checksum	Traffic Misclassification	[30]	Yes
Inert Packet Insertion w/o ACK flag	Traffic Misclassification	[30]	Yes

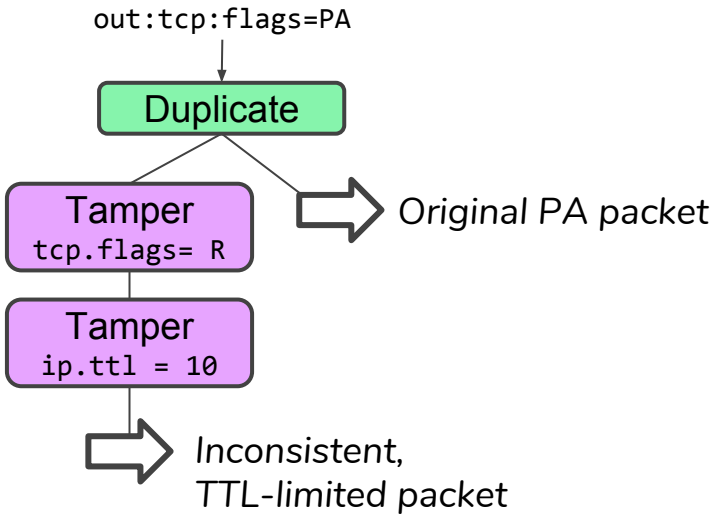
- Initially, gave Geneva access to IP, TCP, and UDP headers
- Geneva **rederived virtually all** of prior work
  - Except strategies we did not give it primitives to access (delay / application layer modifications)
- Found **bugs** in libraries
  - `scapy`, `docker`, and `netfilterqueue`

Evaluation against the Great Firewall of China

- Deployed from two vantage points
- Evolved **4 unique species** of HTTP strategies (two unknown to prior work)
  - Found generic and protocol-specific strategies
  - Every strategy was found in **3 hours or less**
- Geneva was able to **evolve extinct** strategies into **successful ones**

Species	Subspecies	Variant	Genetic Code	Succ.
None	None	None	\/	3%
TCB Desync	Inc. Dataofs	Corrupt Chksum	[field:TCP:flags:PA]-duplicate(tamper{TCP:dataofs:replace:10}(tamper{TCP:chksum:replace:25776},),)-	98%
		Small TTL	[field:TCP:flags:PA]-duplicate(tamper{TCP:dataofs:replace:10}(tamper{IP:ttl:replace:10},),)-	98%
		Invalid Flags	[field:TCP:flags:PA]-duplicate(tamper{TCP:dataofs:replace:10}(tamper{TCP:flags:replace:FRAPUN},),)-	26%
		Corrupt Ack	[field:TCP:flags:PA]-duplicate(tamper{TCP:dataofs:replace:10}(tamper{TCP:ack:corrupt},),)-	94%
		Corrupt WScale	[field:TCP:flags:PA]-duplicate(tamper{TCP:options-wscale:corrupt}(tamper{TCP:dataofs:replace:8},),)-	98%
	Dec. Dataofs	Corrupt MSS [3]	[field:TCP:flags:PA]-duplicate(tamper{TCP:options-mss:corrupt}(tamper{TCP:dataofs:replace:5},),)-	98%
		Corrupt WScale	[field:TCP:flags:PA]-duplicate(tamper{TCP:options-wscale:corrupt}(tamper{TCP:dataofs:replace:5},),)-	97%
	Inv. Payload	Corrupt Chksum	[field:TCP:flags:PA]-duplicate(tamper{TCP:load:corrupt}(tamper{TCP:chksum:corrupt},),)-	98%
		Small TTL	[field:TCP:flags:PA]-duplicate(tamper{TCP:load:corrupt}(tamper{IP:ttl:replace:8}(duplicate(fragment{tcp:-1:False},),),),)-	98%
		Corrupt Ack	[field:TCP:flags:PA]-duplicate(tamper{TCP:load:corrupt}(tamper{TCP:ack:corrupt}(duplicate(fragment{tcp:-1:False},),),),),)-	93%
TCB Teardown	With RST	Corrupt Chksum	[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:R}(tamper{TCP:chksum:corrupt},))-	95%
			[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:R}(tamper{TCP:chksum:corrupt},),)-	51%
		Small TTL	[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:R}(tamper{IP:ttl:replace:10},))-	87%
			[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:R}(tamper{IP:ttl:replace:9},),)-	52%
		Inv. md5Header	[field:TCP:flags:A]-duplicate(,tamper{TCP:options-md5header:corrupt}(tamper{TCP:flags:replace:R},))-	86%
			[field:TCP:flags:A]-duplicate(tamper{TCP:options-md5header:corrupt}(tamper{TCP:flags:replace:RA},),)-	44%
	With RST/ACK	Corrupt Chksum	[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:RA}(tamper{TCP:chksum:replace:27925},))-	90%
			[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:RA}(tamper{TCP:chksum:replace:27925},),)-	66%
		Small TTL	[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:RA}(tamper{IP:ttl:replace:10},))-	94%
			[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:RA}(tamper{IP:ttl:replace:10},),)-	57%
		Inv. md5Header	[field:TCP:flags:A]-duplicate(,tamper{TCP:options-md5header:corrupt}(tamper{TCP:flags:replace:R},))-	94%
			[field:TCP:flags:A]-duplicate(tamper{TCP:options-md5header:corrupt}(tamper{TCP:flags:replace:R},),)-	48%
		Corrupt Ack	[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:RA}(tamper{TCP:ack:corrupt},),)-	43%
			[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:RA}(tamper{TCP:ack:corrupt},))-	31%
	Invalid Flags	Corrupt Chksum	[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:FRAPUEN}(tamper{TCP:chksum:corrupt},))-	89%
			[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:FRAPUEN}(tamper{TCP:chksum:corrupt},),)-	48%
		Small TTL	[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:FREACN}(tamper{IP:ttl:replace:10},))-	96%
			[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:FRAPUEN}(tamper{IP:ttl:replace:10},),)-	56%
		Inv. md5Header	[field:TCP:flags:A]-duplicate(,tamper{TCP:flags:replace:FRAPUN}(tamper{TCP:options-md5header:corrupt},))-	94%
			[field:TCP:flags:A]-duplicate(tamper{TCP:flags:replace:FRAPUEN}(tamper{TCP:options-md5header:corrupt},),)-	55%
Segmentation	With ACK	Offsets	[field:TCP:flags:PA]-fragment{tcp:8:False}-| [field:TCP:flags:A]-tamper{TCP:seq:replace:2258679050}-	92%
			[field:TCP:flags:PA]-fragment{tcp:8:False}-| [field:TCP:flags:A]-tamper{TCP:seq:replace:2258679050}(tamper{TCP:flags:replace:PA},)-	95%
	Reassembly	Offsets	[field:TCP:flags:PA]-fragment{tcp:8:True}(,fragment{tcp:4:True})-	98%
			[field:TCP:flags:PA]-fragment{tcp:4:True}(,fragment{tcp:19:True})-	96%

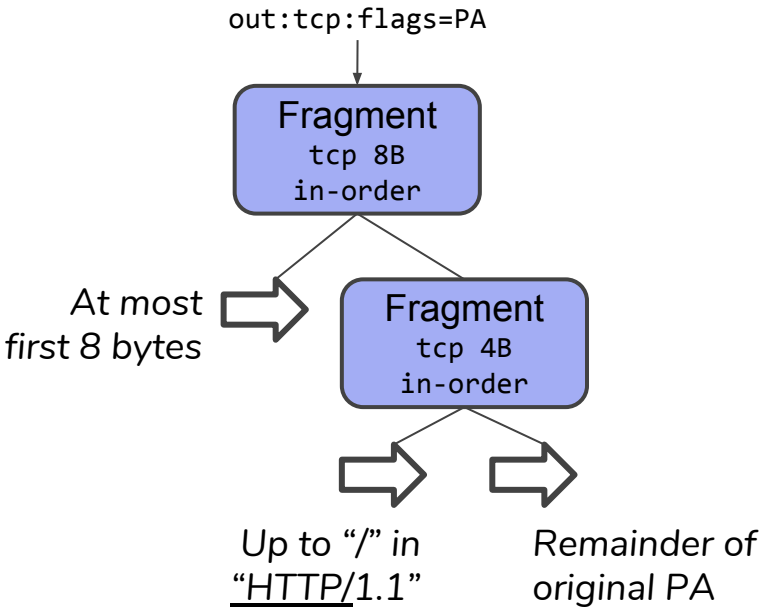
Example Strategies



During the three way handshake inject a TTL-limited RST packet

Gap in logic

GET /?q=ultrasurf HTTP/1.1

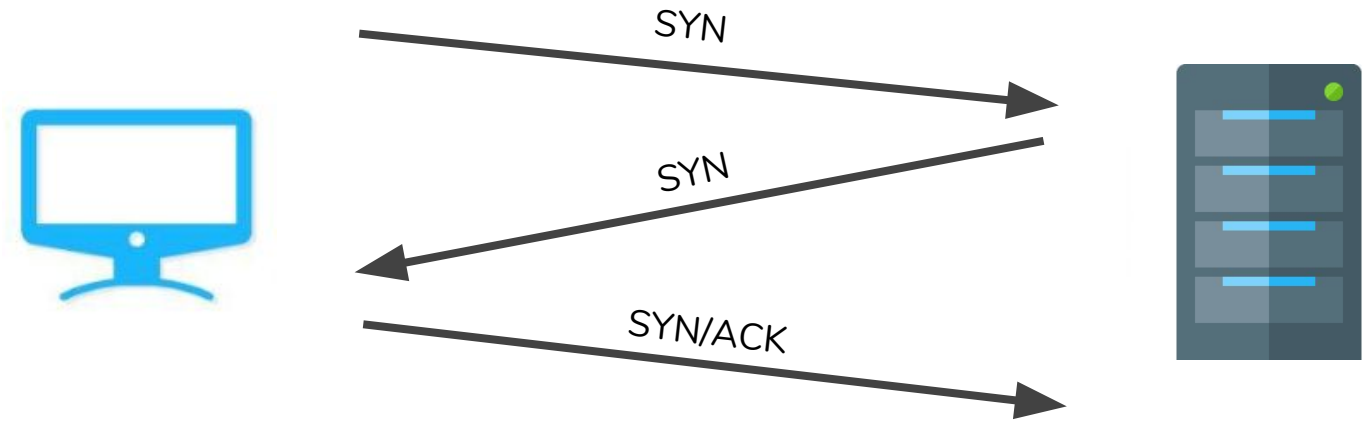


Segments the TCP payload, but *does not split up the censored keyword!*

Bug in implementation

Server-Side Evasion

- Deployed Geneva from the server side
- Geneva **independently rederived** a form of the TCP simultaneous open to reverse the roles of client and server
- Defeats censorship ***with no client involvement whatsoever***



Looking Forward

- New **locations**:
  - India, Russia, Egypt, Turkey, Iran
- More **protocols**:
  - IPv6, DNS, FTP, TLS, HTTPS
- Open to more!

New Approach to Active Measurement

- Envision this as a first step towards **AI-driven** active measurement

- Deploy AI to adaptively measure
  - Discover unexpected behavior
  - Derive the minimum behavior to recreate the issue
  - Hand off to human researchers to understand

Geneva

- Genetic algorithm that **evolves censorship evasion** strategies
- Rederived nearly all prior work's strategies against the GFW
- Developed new strategies on multiple protocols:
  - Client-side HTTP
  - Server-side HTTP
  - Client-side DNS
- Allows for new approaches to active measurement