# Towards illuminating a Censorship Monitor's Model to Facilitate Evasion


Sheharbano Khattak*, ***Mobin Javed***◆, Philip D. Anderson* and Vern Paxson◆★

* Independent Researcher
◆ U.C. Berkeley
★ International Computer Science Institute

# In the next 19.5 mins..

I'm going to talk about:

- ***How to*** *Reverse Engineer a Censor Monitor:*
  - Exhaustively *probing* ***stateful onpath*** censors to infer information about various elements

- *And an* ***exemplar***:
  - Evasion vulnerabilities we found in the Great Firewall of China

# A look at the Evasion landscape

## Existing evasion tools:

=> Clayton et al. (2006)

- ignore RSTs

=> WestChamber (2010)

- send fake RSTs

⇒ Brdgrd

- Exploit lack of TCP reassembly for TLS negotiations

## Our Work:

- A systematic investigation of evasion opportunities
- Goals:
  - Require *expensive* changes to system's basic model to remedy vulnerabilities
  - Require only client-side or server-side traffic manipulation

# Design of a Censor

What to Censor?

**Analysis Model**
(how to look for sensitive stuff in traffic)

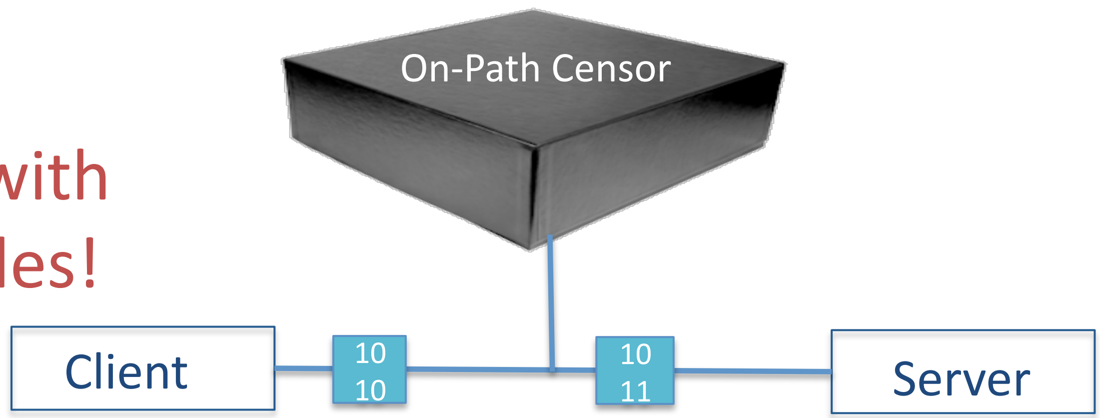
How to execute censorship?

- Tradeoff between completeness of analysis and scalability.
- Same problem of ‘traffic reconstruction’ as NIDS.

*We draw our work mainly on the body of knowledge established by the NIDS community.*

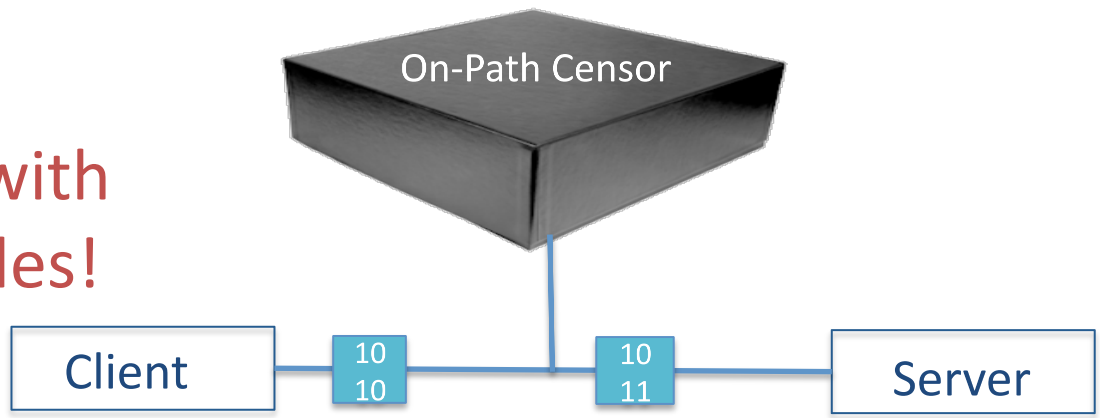
# Probing a Censor to infer model

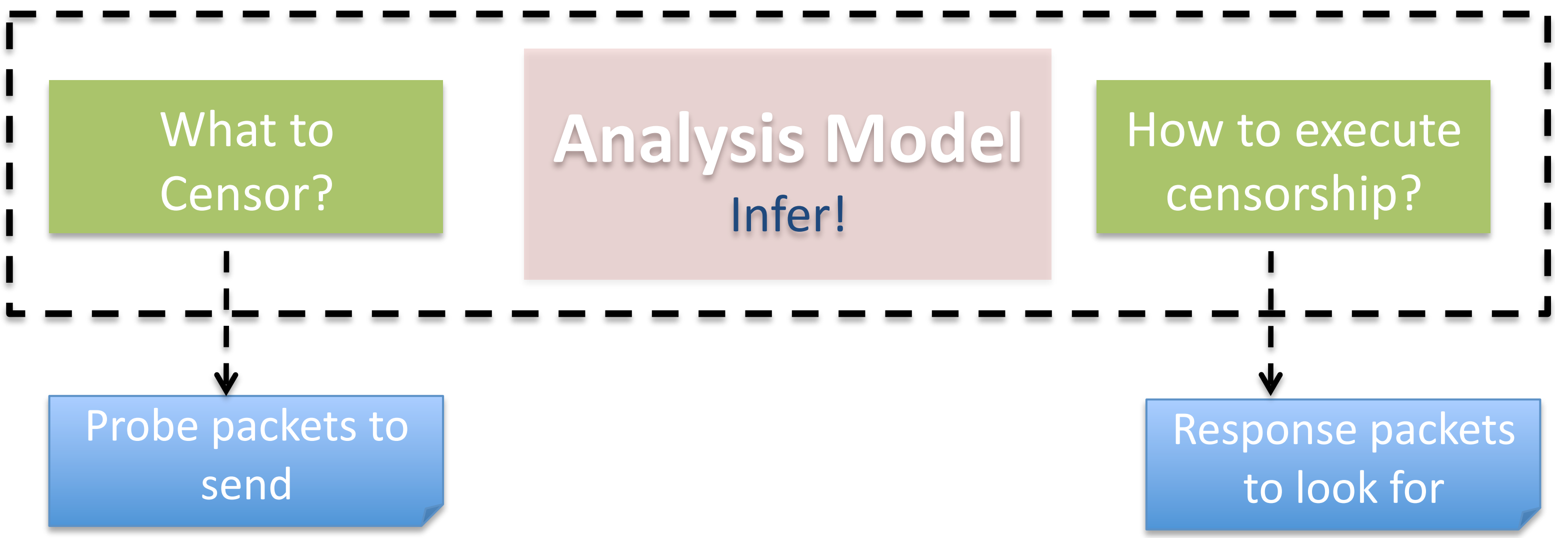
A censor is a black-box, but with a few observables!



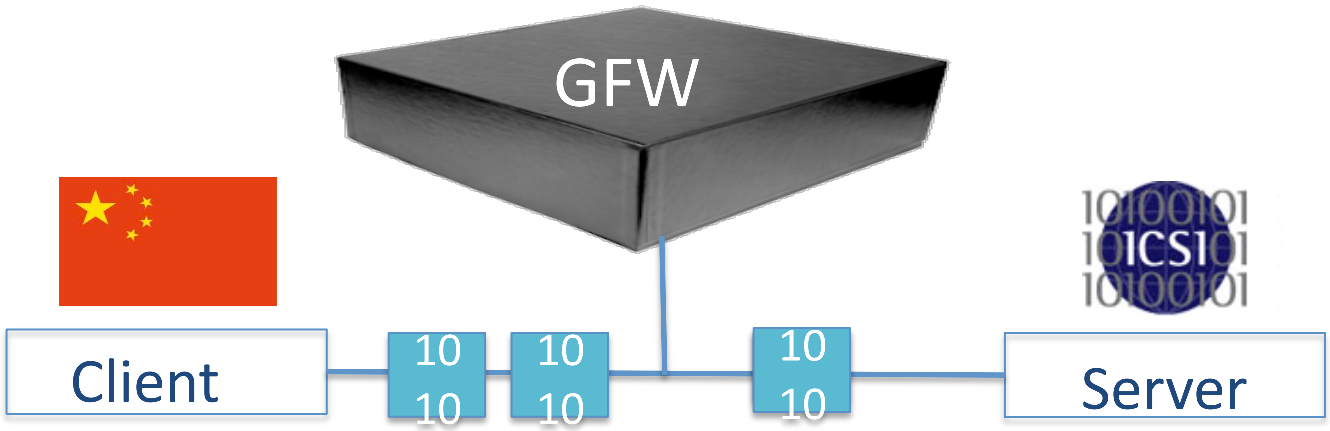
# Probing a Censor to infer model

A censor is a black-box, but with a few observables!



# Probing Methodology




- Test sensitive keywords (for e.g. Falungong) in IP /TCP segment/ HTTP request / HTTP Reply
- GFW censors only once it has seen a complete HTTP request.

Trigger Packets

- Three RST packets with varying gaps in sequence numbers

GFW Response Packets

# Model Elements to Probe

1. TCB Creation
2. IP/TCP Reassembly
3. State Management
4. TCB Teardown
5. Protocol Message Interpretation
(Both network and higher layers)

*For this work we focused on stateful on-path monitors*

# 1. TCB Creation

- Three-way handshake or partial handshake?

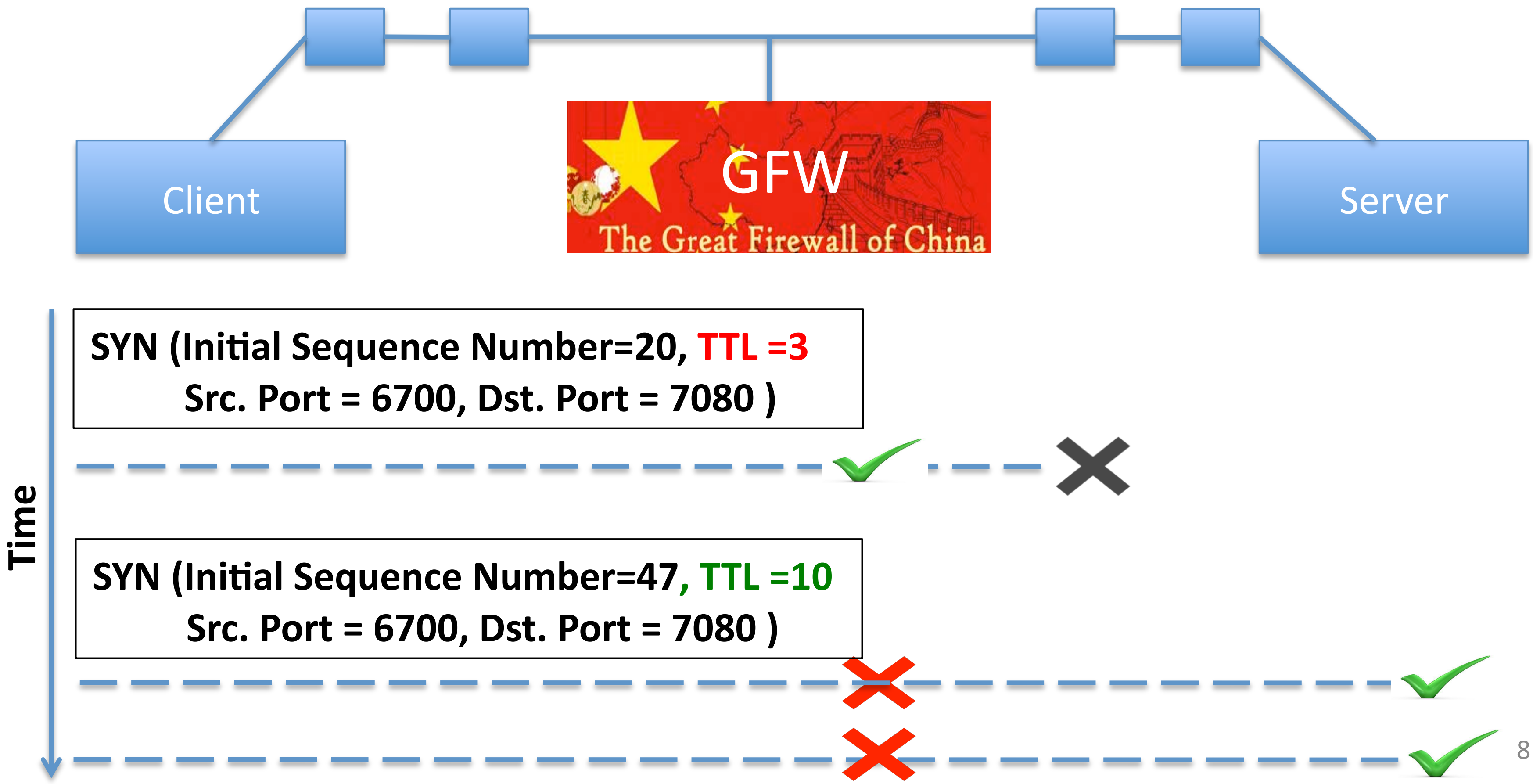
| |
|---|
| *Test 1a:* **SYN but no responding SYN-ACK** |
| *Test 1b:* SYN-ACK but no initial SYN |
| *Test 1c*: Both SYN and SYN-ACK |

(In all three tests, trigger packets follow handshake packets)

- Evasion Vulnerabilities:
  - SYN Flooding
  - Unsynchronized monitoring

# 1. TCB Creation (2)

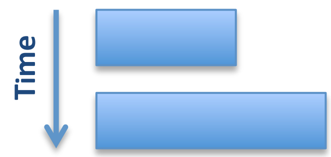
*Unsynchronized monitoring illustration*



# 2. IP/TCP Reassembly

- How to resolve ambiguous cases of temporally separated overlapping fragments/segments?

Example:



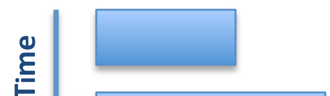

- Tested each of the 18 possible cases for ambiguous overlap.

- GFW prefers:
  - Original IP fragment for all cases except for one case
  - Subsequent TCP segments for a subset of cases
  - Lacks reassembly capability for other TCP segment cases

# 2. IP/TCP Reassembly

- How to resolve ambiguous cases of temporally separated overlapping fragments/segments?

Example:




To evade: Send sensitive keywords in overlapping fragments/ segments that evade GFW's reassembly policy !!

*(For evasion to work, server must reassemble as expected.)*

- GFW prefers:
  - Original IP fragment for all cases except for one case
  - Subsequent TCP segments for a subset of cases
  - Lacks reassembly capability for other TCP segment cases

# 3. State Management

- How long and how much state to keep?

- Send increasing amounts of time and volume of non-sensitive data prior to sensitive data

- GFW's state-keeping capabilities:
  - Without "holes": 10 hours (even with 1 GB+ worth of data)
  - With "holes": 1 hour/1 KB

# 3. State Management

- How long and how much state to keep?

To evade: Exploit GFW's buffering capabilities.
DoS or cause it to evict state!!

- GFW's state-keeping capabilities:
  - Without "holes": 10 hours (even with 1 GB+ worth of data)
  - With "holes": 1 hour/1 KB

# 4. TCB Teardown

- How to determine parties have torn down connection?

| |
|---|
| Test 4a: require RST (A) from one party |
| Test 4b: require RST (A) from both parties |
| Test 4c: require FIN (A) from one party |
| Test 4d: require FIN (A) from both parties |

- GFW tears down on:
  - FIN/RST packet (even ones without ACK bit set).

# 5. Protocol Message Interpretation

- Does the censor perform protocol validation?
  - Does it respect what different header field/values mean?
  - Is it complete?
  - How does it deal with ambiguous messages?

- Layer-by-layer header walk trying out possible values of each header field

- Here we report only interesting ones

# 5. Protocol Message Interpretation

**TCP Exemplars:**

- GFW accepts packets with incorrect TCP checksums
- GFW accepts packets that lack ACK/ have wrong ACK

# 5. Protocol Message Interpretation

TCP Exemplars:

- GFW acc
incorrect T

- GFW acc
ACK/ have

HTTP Exemplars (see paper for more):

- RFC Deviant HTTP Requests: Extra space between Request method and Request URI bypasses inspection
GET _ _ /falungong.html HTTP/1.1\r\n

- GFW inspects only first 2K bytes into the request URI

# Cost of Fixing Evasion Bugs

TCB Creation

Protocol Message Interpretation

Mostly Easy
But trade off completeness for scalability

Reassembly

Requires **inline** normalization
**Expensive!**

State Management

TCB Destruction

- Kill connections no longer monitored.
**Collateral Damage!**

# Future Work

- Automated Model Extraction
  - For a given censor over time
  - New censors in new countries
  - Assessment of Analysis Inconsistencies
- Evasion Tools

# Q & A!