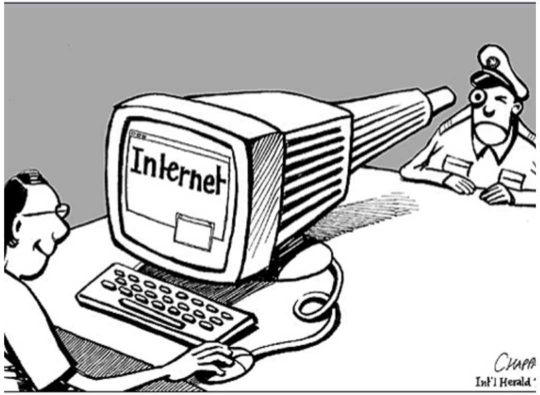
SymTCP: Eluding Stateful Deep Packet Inspection with Automated Discrepancy Discovery

Zhongjie Wang, Shitong Zhu, Yue Cao, Zhiyun Qian,
Chengyu Song, Srikanth Krishnamurthy, Kevin Chan, and Tracy Braun

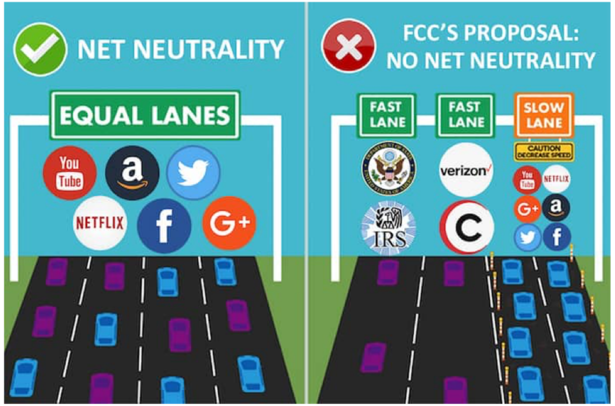
UC RIVERSIDE
UNIVERSITY OF CALIFORNIA

DEVCOM
ARMY RESEARCH LABORATORY

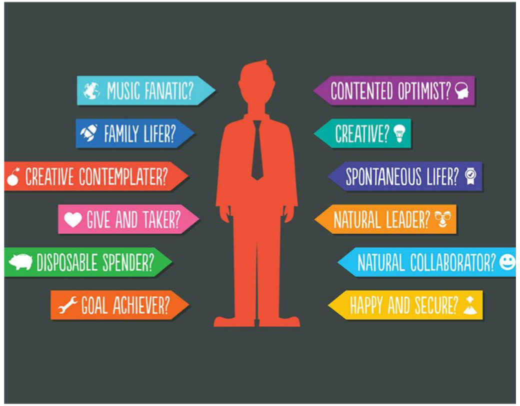
What is DPI (Deep Packet Inspection)?



Censorship and Surveillance

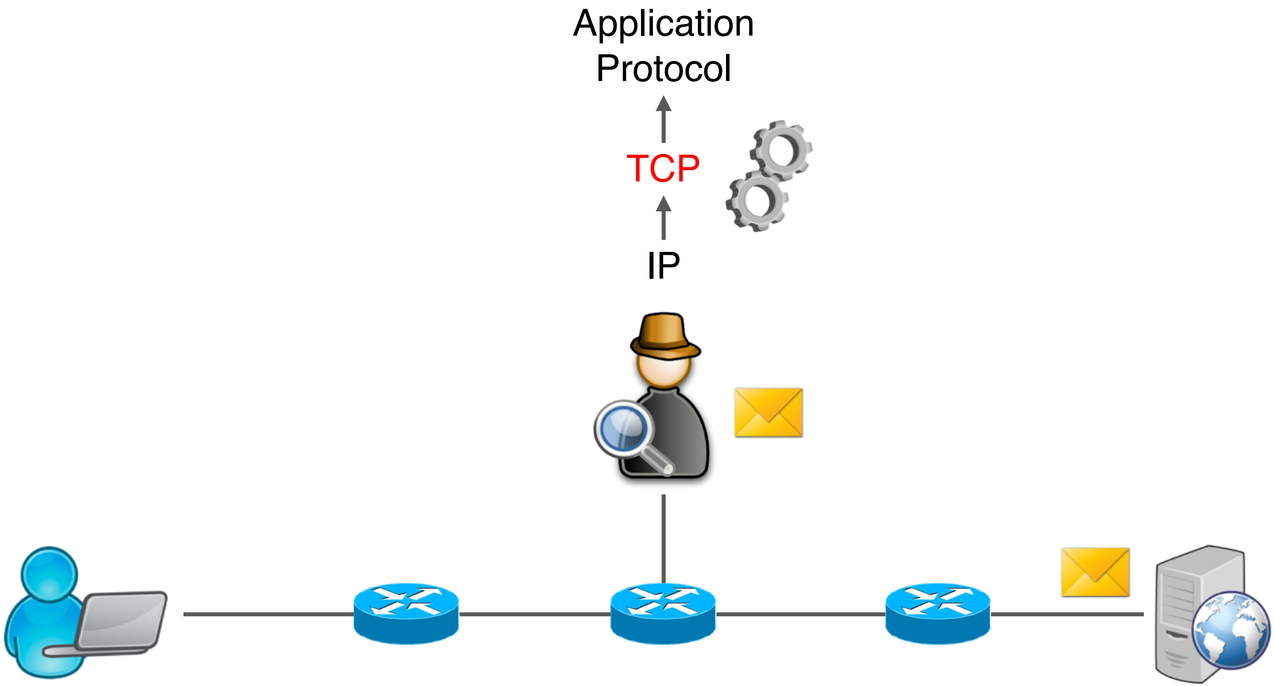


ISP Traffic Differentiation

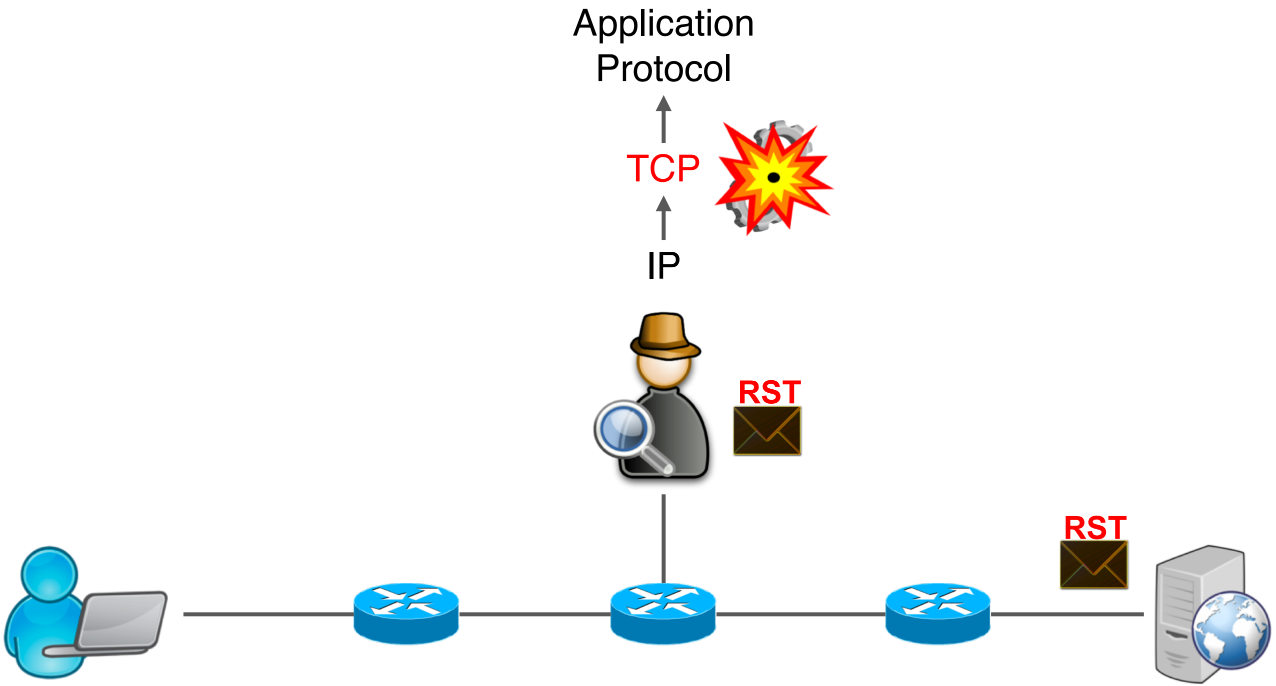


Modeling Users for Online Ads

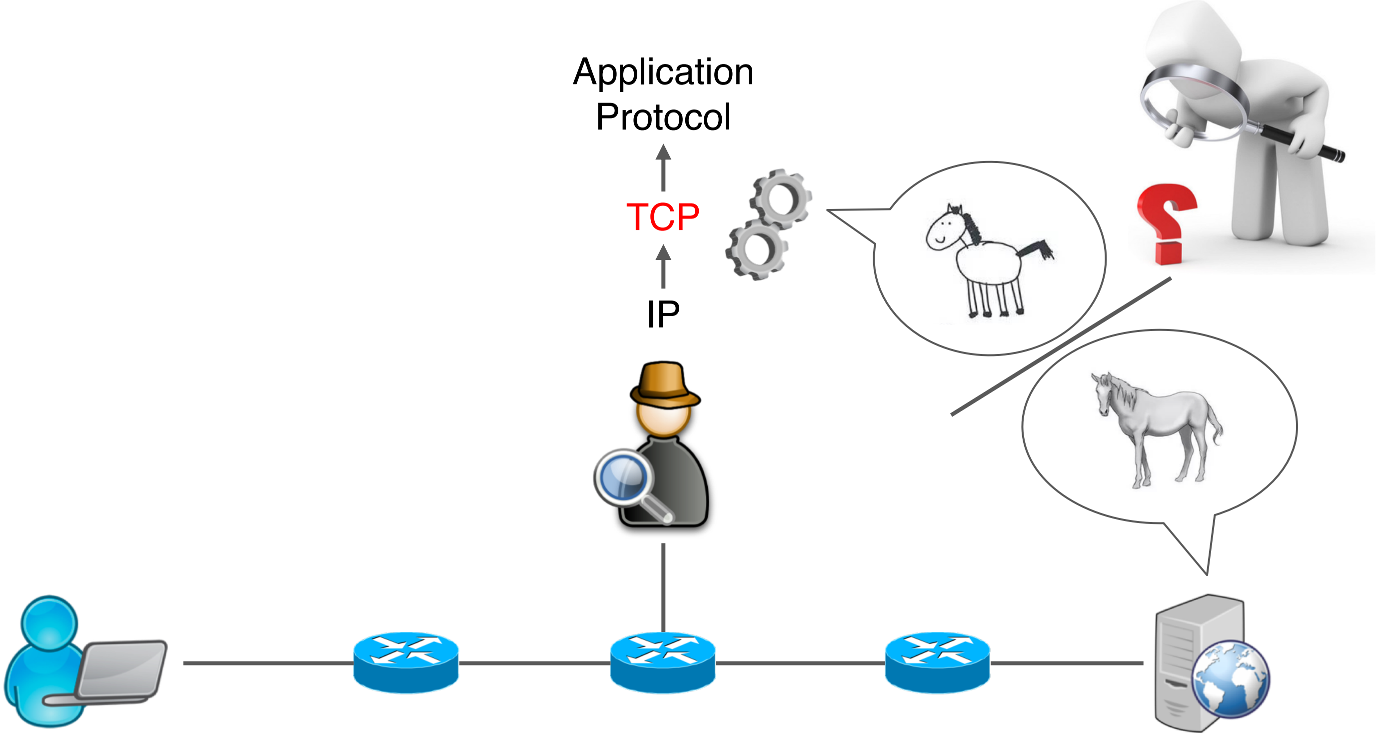
How does DPI work?



How does DPI work?



How does DPI work?



Implementation-level discrepancy

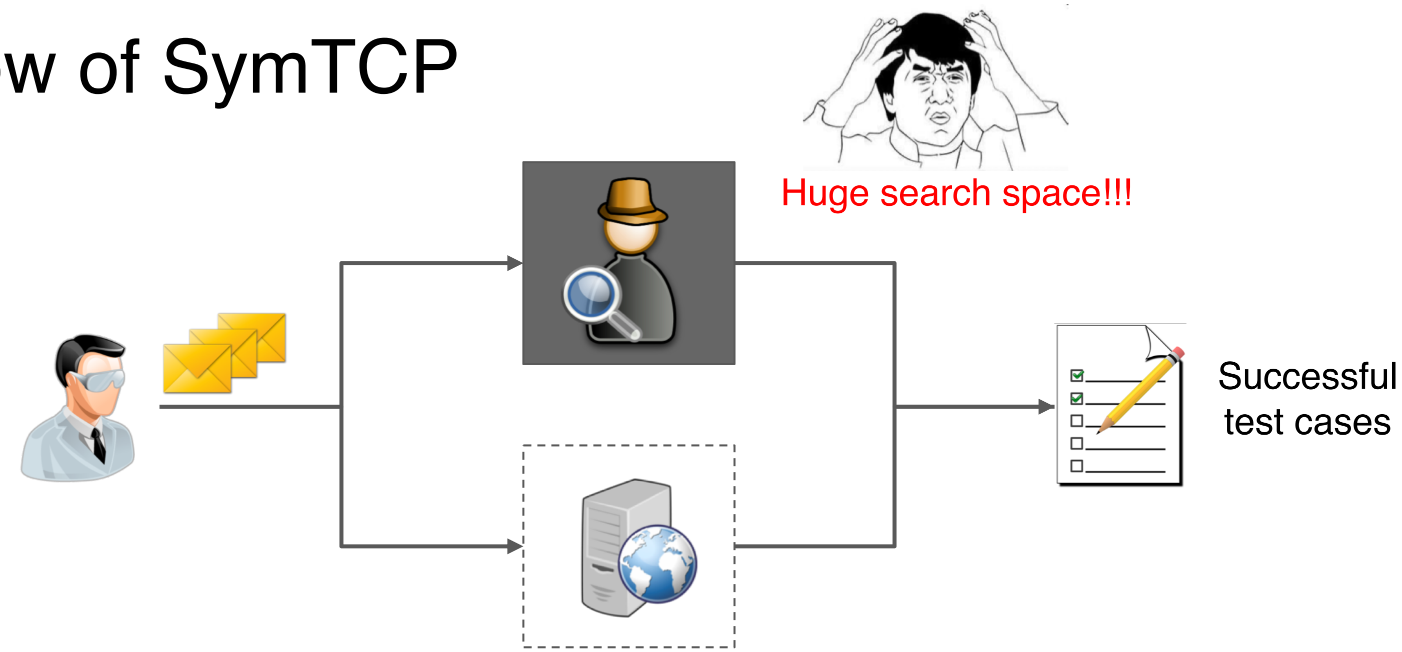
```
// Linux TCP timestamp validation
if ((signed int)(last_tsval - current_tsval) <= 1) {
    // succeed
} else {
    // fail
}
```

$\text{last\_tsval} - 1 \le \text{current\_tsval} \le \text{last\_tsval} + 2^{31}$

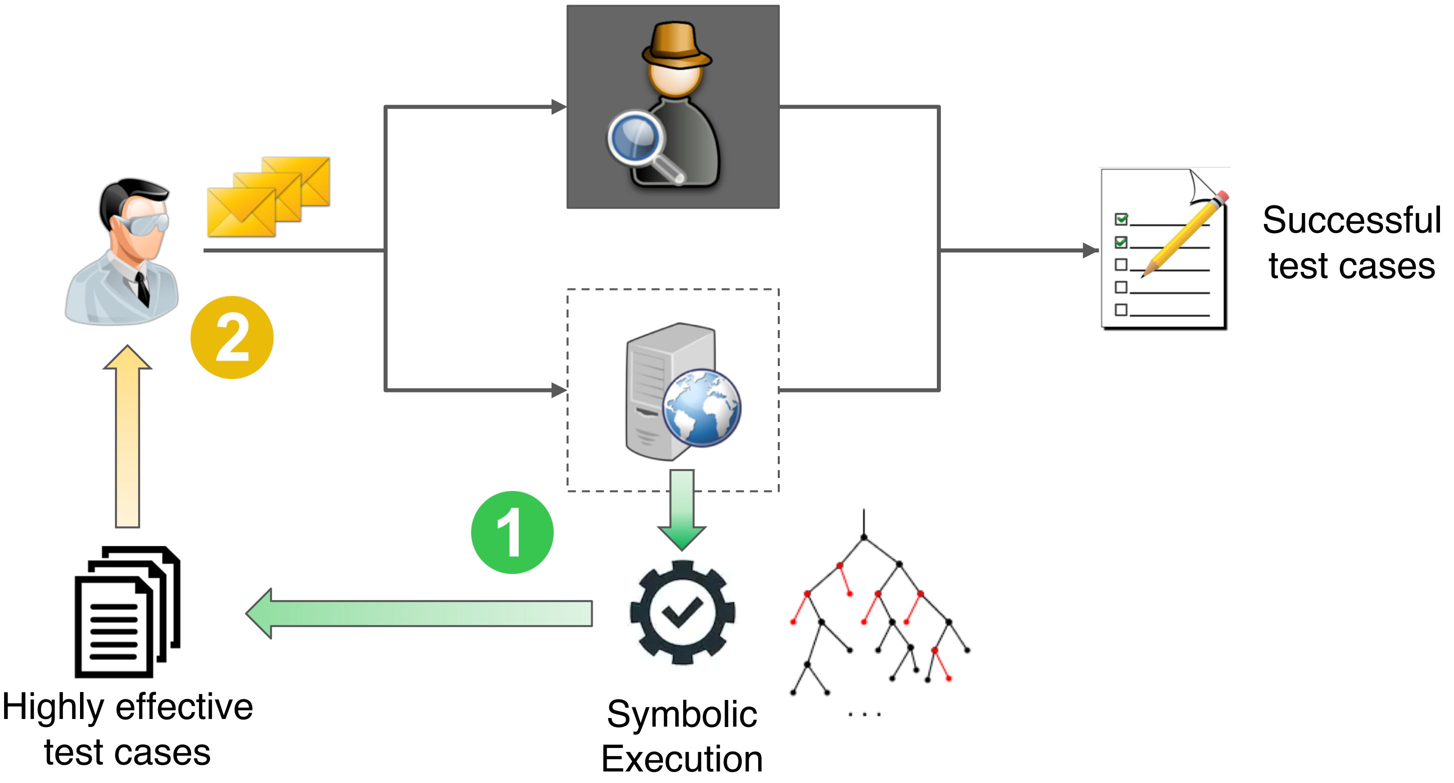
```
// Snort TCP timestamp validation
if ((signed int)((current_tsval - last_tsval) + 1) < 0) {
    // fail
} else {
    // succeed
}
```

$\text{last\_tsval} - 1 \le \text{current\_tsval} \le \text{last\_tsval} + 2^{31} \mathbf{- 2}$

Workflow of SymTCP



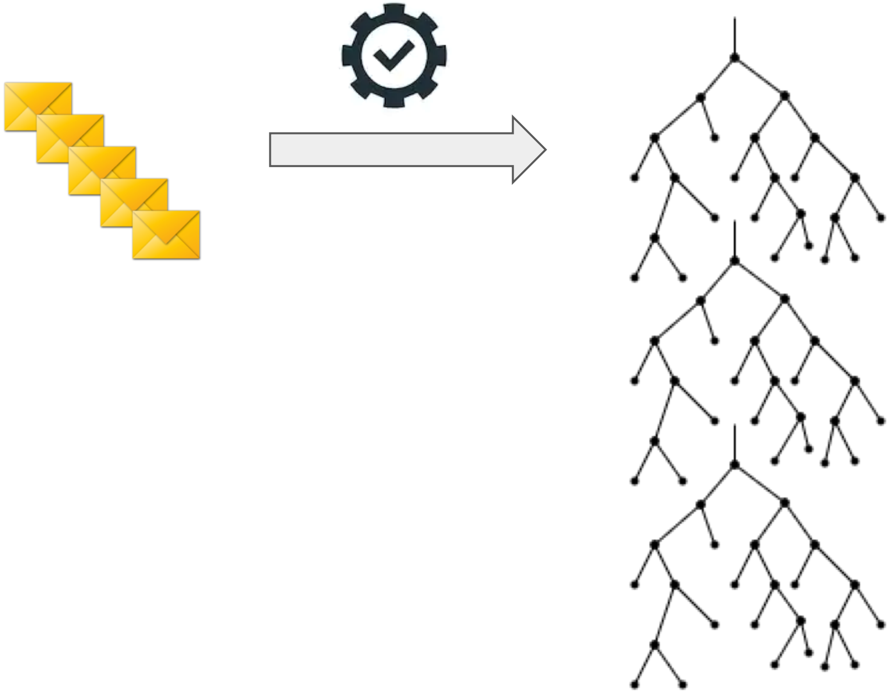
Workflow of SymTCP

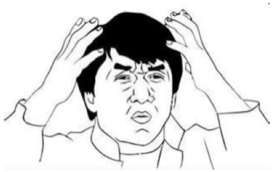


Problem with symbolic execution

All possible packets

All possible execution paths





Path explosion!!!

Pruning decisions

Labeling “drop” / “accept” points

In the program, we label where a packet gets dropped or accepted (i.e. TCP state changed). We try to cover these accept/drop points.

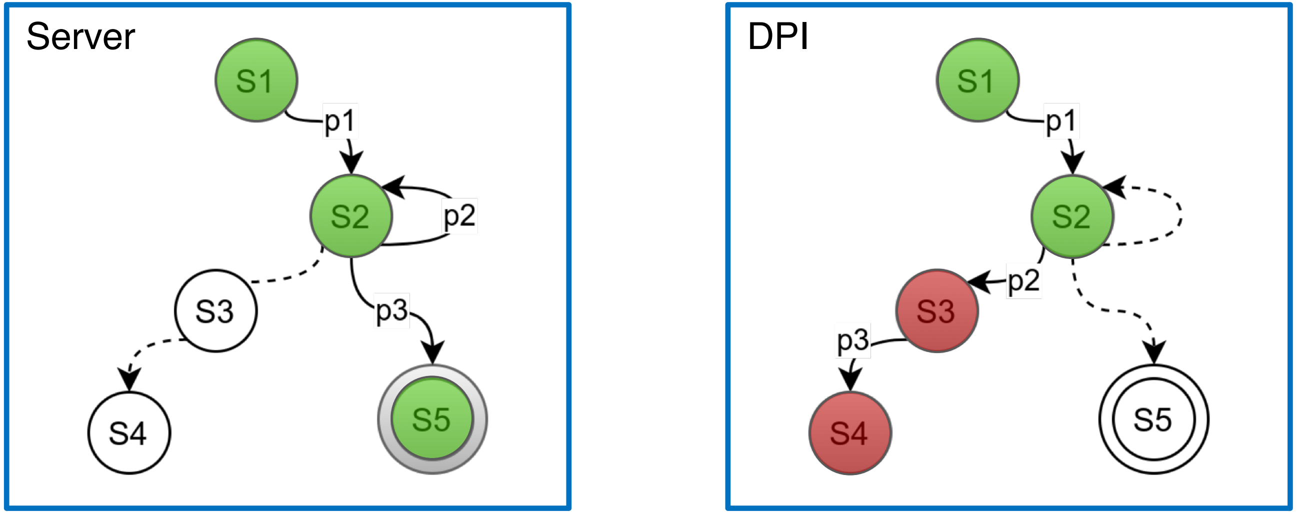
Bounding TCP options

We allow each TCP option to occur only once, and at most 5 different TCP options in a packet.

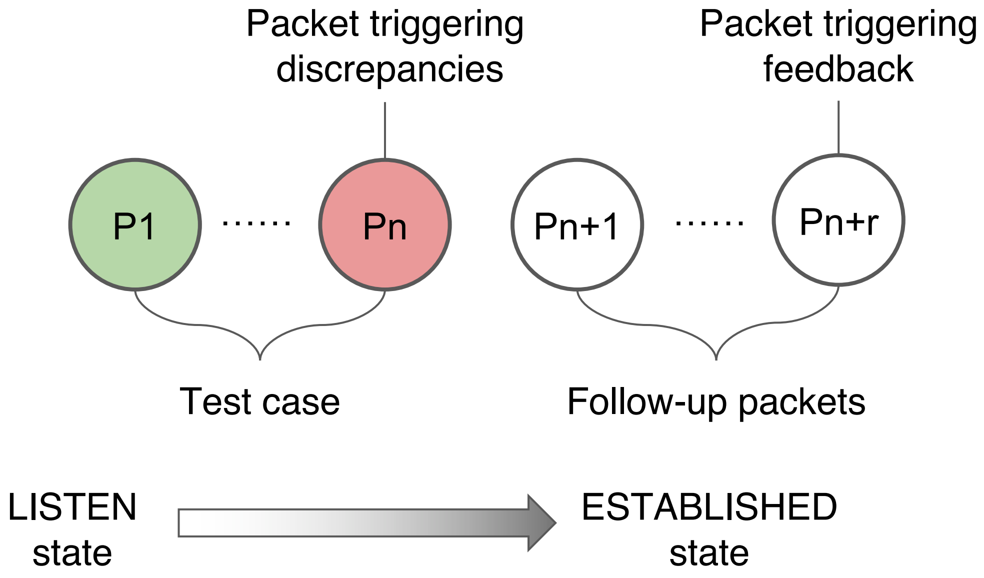
Pruning uninteresting TCP states

We terminate an execution path once it reaches any uninteresting TCP state (e.g., TIME_WAIT, CLOSED)

Differential testing DPI



Complete packet sequence



Symbolic execution performance

- Linux kernel v4.9.3
- 72 core Intel Xeon CPU and 256GB memory
- 1/2/3 symbolic packets
- 20/40/60 byte length packet

No TCP options

# of pkts	20-byte TCP pkts		40-byte TCP pkts		[illegible]	
	Time to cover	Covered drop points	Time to cover	Covered drop points	[illegible]	[illegible]
1	5s	8	5s	9	[illegible]	[illegible]
2	20s	16	20m	19	18m	18
3	50s	31	1h2m	39	40m	38

Time cost could vary due to randomness in path selection of symbolic execution.

56,787 test cases
Sampled 10,000 test cases

Zeek (formerly Bro)

- 6082 successful test cases, 9 strategies, 2 novel strategies

TABLE IV. SUCCESSFUL STRATEGIES ON ZEEK V2.6

Strategy	TCP state	Insertion/Evasion packet	Linux	Zeek
† SYN with data	L/SR/E	(I) SYN packet with data	Ignore data	Accept data
† Multiple SYN	SR/E	(I) SYN packet with out-of-window SEQ num	Discard and send ACK	Reset TCB
† Pure FIN	E	(I) Pure FIN packet without ACK flag	Discard (may send ACK)	Flush and reset receive buffer
† Bad RST/FIN	SR/E	(I) RST or FIN packet with out-of-window SEQ num	Discard (may send ACK)	Flush and reset receive buffer
† Data overlapping	SR/E	(I) Out-of-order data packet, then overlapping in-order data packet	Accept in-order data	Accept first data
† Data without ACK	SR/E	(I) Data packet without ACK flag	Discard	Accept
† Data bad ACK	E	(I) Data packet with ACK > snd_nxt or < snd_una - window_size	Discard	Accept
∗ Big gap	SR/E	(I) Data packet with SEQ > rcv_nxt + max_gap_size (16384)	Accept	Ignore later data
∗ SEQ < ISN	SR/E	(E) Data packet with SEQ num < client ISN and in-window data	Accept in-window data	Ignore

* TCP State: L - Listen, SR - SYN_RECV, E - ESTABLISHED. (I) - Insertion, (E) - Evasion. † - Old strategy, ∗ - New strategy.

Snort

- 652 successful test cases, 11 strategies, 3 novel

TABLE V. SUCCESSFUL STRATEGIES ON SNORT V2.9.13

Strategy	TCP state	Insertion/Evasion packet	Linux	Snort
† Multiple SYN	E	(I) SYN packet with in-window SEQ num	Discard and send ACK	Teardown TCB
† In-window FIN	E	(I) FIN packet with SEQ num in window but $\neq$ rcv_nxt	Ignore FIN (may accept data)	Cut off later data
† FIN/ACK bad ACK	E	(I) FIN/ACK packet with ACK num $>$ snd_nxt or $<$ snd_una - window_size	Discard (may send ACK)	Cut off later data
† FIN/ACK MD5	SR/E	(I) FIN/ACK packet with TCP MD5 option	Discard	Cut off later data
† In-window RST	E	(I) RST packet with SEQ num $\neq$ rcv_nxt but still in window	Discard and send ACK	Teardown TCB
† RST bad timestamp	SR	(I) RST packet with bad timestamp	Discard	Teardown TCB
† RST MD5	SR/E	(I) RST packet with TCP MD5 option	Discard	Teardown TCB
† RST/ACK bad ACK num	SR	(I) RST/ACK packet with ACK num $\neq$ server ISN + 1	Discard	Teardown TCB
∗ Partial in-window RST	E	(I) RST packet with SEQ num $<$ rcv_nxt but partial data in window	Discard	Teardown TCB
∗ Urgent data	SR/E	(E) Data packet with URG flag and urgent pointer set	Consume 1 byte urgent data	Ignore all data before urgent pointer
∗ Time gap	SR/E	(E) Data packet timestamp = last timestamp + 0x7fffffff/0x80000000	Accept	Ignore

* TCP State: L - Listen, SR - SYN_RECV, E - ESTABLISHED. (I) - Insertion, (E) - Evasion. † - Old strategy, ∗ - New strategy.

Great Firewall of China (GFW)

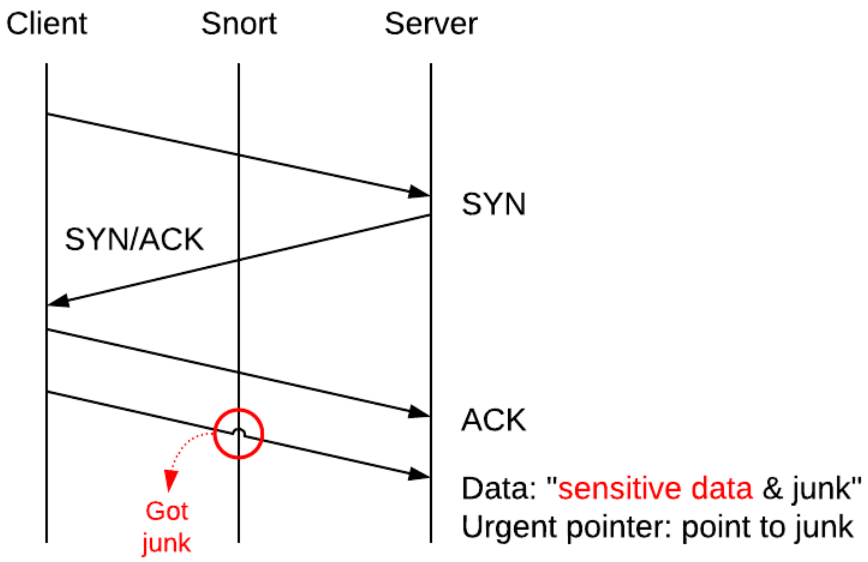
- 4587 successful test cases, 12 strategies, 9 novel

TABLE VI. SUCCESSFUL STRATEGIES ON THE GFW

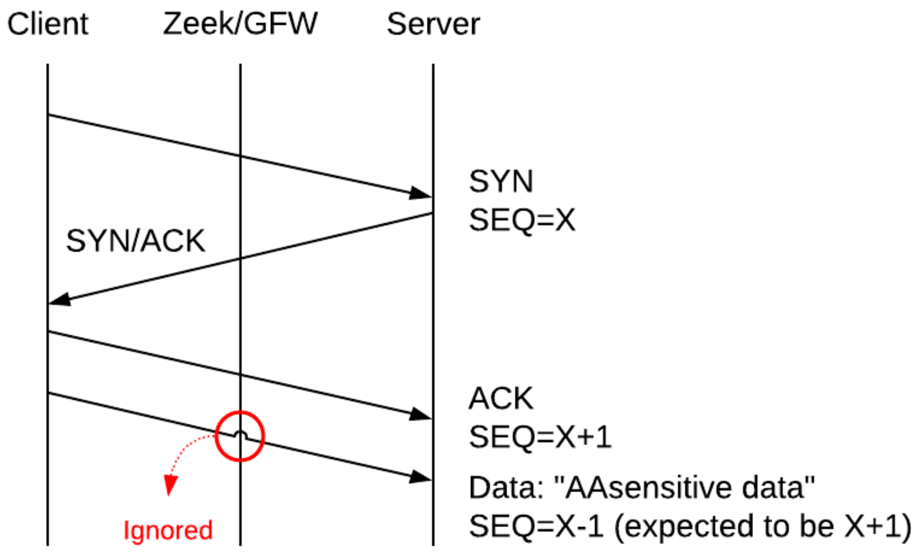
Strategy	TCP state	Insertion/Evasion packet	Linux	GFW
† Bad RST	SR/E	(I) RST packet with bad checksum or TCP MD5 option	Discard	Teardown TCB
† Bad data	SR/E	(I) Data packet with bad checksum or TCP MD5 option or bad timestamp	Discard	Accept
† Data without ACK	SR/E	(I) Data packet without ACK flag	Discard	Accept
$*$ SEQ $\leq$ ISN	SR/E	(E) Data packet with SEQ num $\leq$ client ISN and in-window data	Accept in-window data	Ignore
$*$ Small segments	SR	(E) Data packet with payload size $\leq$ 8 bytes	Accept	Ignore
$*$ FIN with data	SR/E	(I) FIN packet with data and without ACK flag	Discard	Teardown TCB
$*$ Bad FIN/ACK data	E	(I) FIN/ACK packet with data and bad checksum or TCP MD5 option or bad timestamp	Discard	Teardown TCB
$*$ FIN/ACK data bad ACK	E	(I) FIN/ACK packet with data and ACK num $>$ snd_nxt or $<$ snd_una - window_size	Discard	Teardown TCB
$*$ Out-of-window SYN data	SR	(I) SYN packet with SEQ num out of window and data	Discard and send ACK	Desynchronized
$*$ Retransmitted SYN data	SR	(I) SYN packet with SEQ num = client ISN and data	Discard	Desynchronized
$*$ RST bad timestamp	SR	(I) RST packet with bad timestamp	Discard	Teardown TCB
$*$ RST/ACK bad ACK num	SR	(I) RST/ACK packet with SEQ num $\neq$ server ISN + 1	Discard	Teardown TCB

* TCP State: L - Listen, SR - SYN_RECV, E - ESTABLISHED. (I) - Insertion, (E) - Evasion. † - Old strategy, $*$ - New strategy.

Case study



1. Urgent Pointer (Snort)



2. Underflow SEQ (Zeek & GFW)

Key contributions

- A novel approach that **combines whitebox and blackbox testing**
  - Whitebox: Extract a reference model from server with symbolic execution
  - Blackbox: Infer internal states of DPI with follow-up packets
- First to run symbolic execution on **full-fledged TCP implementation** and send **multiple symbolic packets**
- Highly **efficient** and **effective automated tool** to unearth discrepancies between different TCP implementations
  - Facilitate DPI elusion
  - Help developers fix implementation bugs

Conclusion

- A novel approach combines whitebox and blackbox testing to automatically discover TCP implementation-level discrepancies
- Evaluated against 3 well-known DPI systems, Zeek (Bro), Snort, and the GFW, and found 14 novel strategies
- A significant step in testing and eluding DPI systems

Thank You!

Email: zwang048@ucr.edu
Homepage: https://zhongjie.me

GitHub Repo