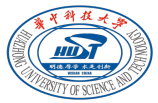
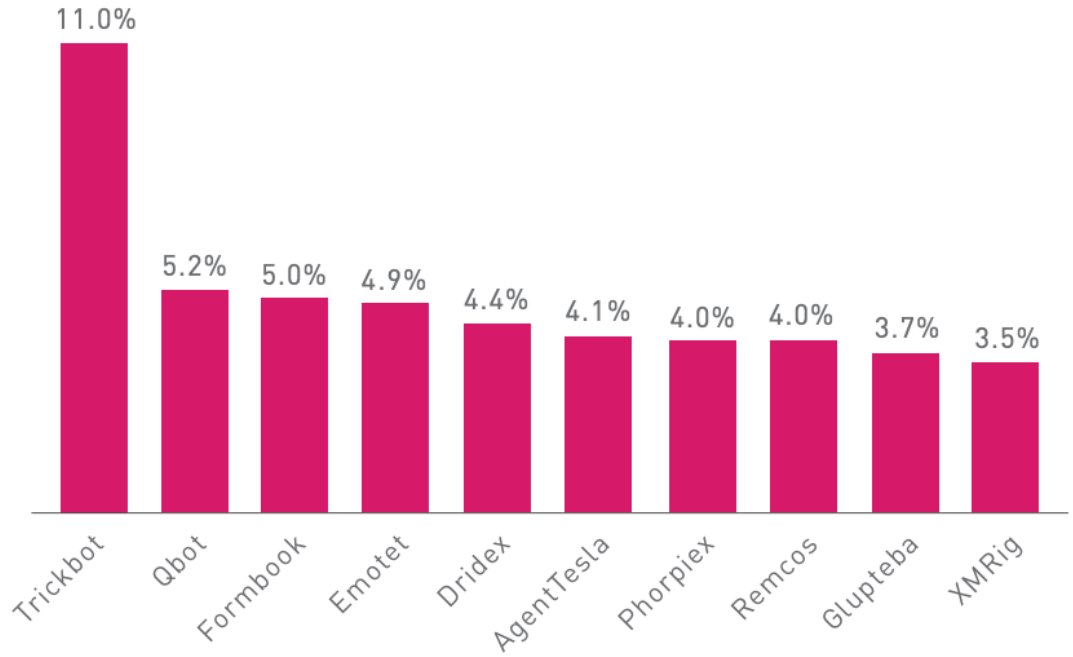
StateDiver: Testing Deep Packet Inspection Systems with State-Discrepancy Guidance

Zhechang Zhang Bin Yuan* Kehan Yang Deqing Zou Hai Jin

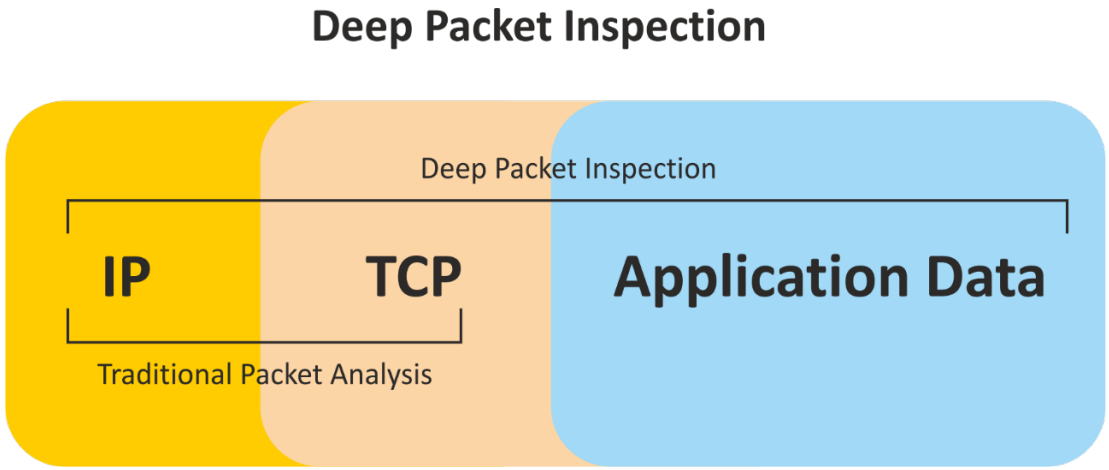


華中科技大學
Huazhong University of Science and Technology

Global Malware Statistics



CheckPoint Research. Percentage of Corporate Networks Attacked by Each Malware Family. Cyber Security Report 2022. 2022



Catchpoint. A Guide to Deep Packet Inspection. 2017

State-of-the-art Solutions

- Manual construction, Symbolic execution and Fuzzing

  INTANG [IMC'17] lib·erate [IMC'17]

  SymTCP [NDSS'20] DPIFuzz [ACSAC'20]

  Geneva [CCS'19] Themis [CCS'21] TCPFuzz [ATC'21]

- Former packets have **rendered** the DPI to an **abnormal** state
  - discrepancy — directly guide fuzzing process
  - no existing work

Our work: StateDiver

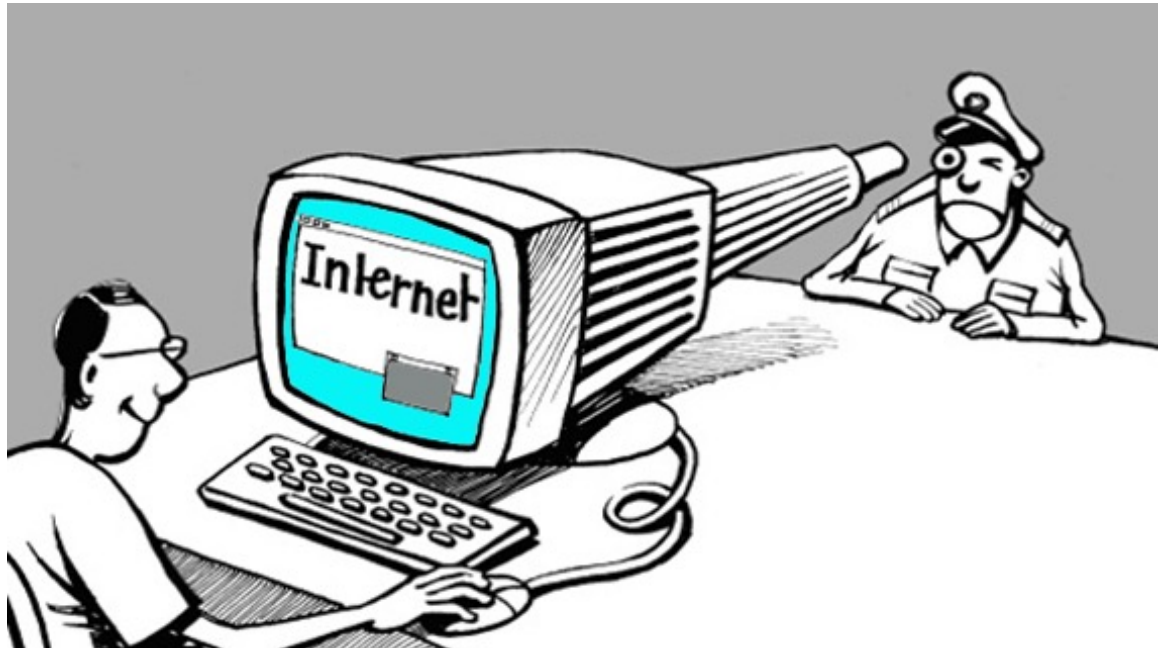
- A fuzzing framework that automatically discovers DPI-bypass strategies using state-discrepancies

- Contributions
  - new feedback mechanism
  - end-to-end system
  - evaluated against 3 most famous open-source DPI systems, and found 16 bypass strategies (8 new and 8 previously known)

Deep Packet Inspection System (DPI)

- Powerful data-processing technique
  - conventional packet filtering (e.g. iptables) : at or below transport layer
  - DPI : application layer or even encrypted data

- Customized protocol stacks
  - track the state of each connection
  - rebuild data streams

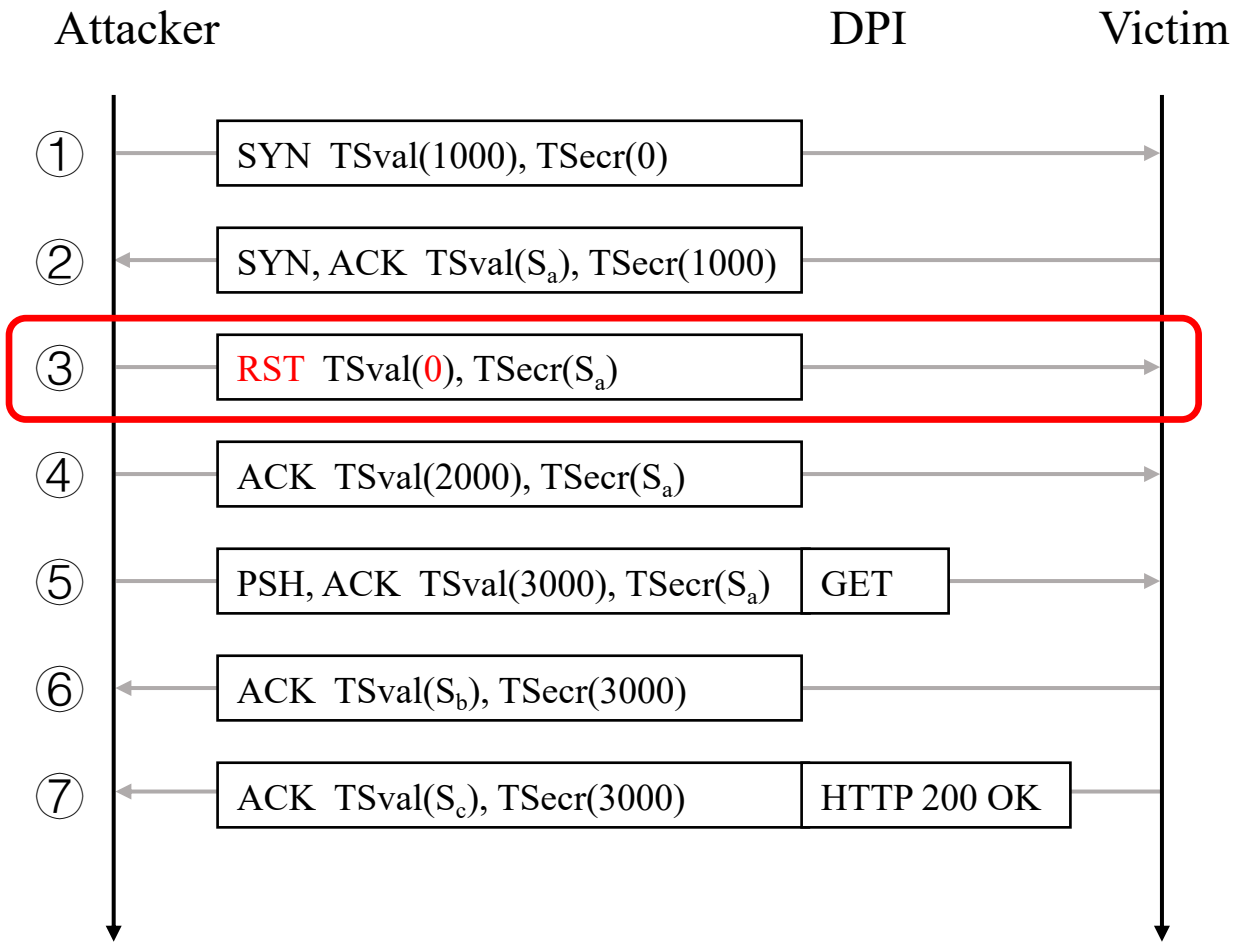


Customized Protocol Stacks

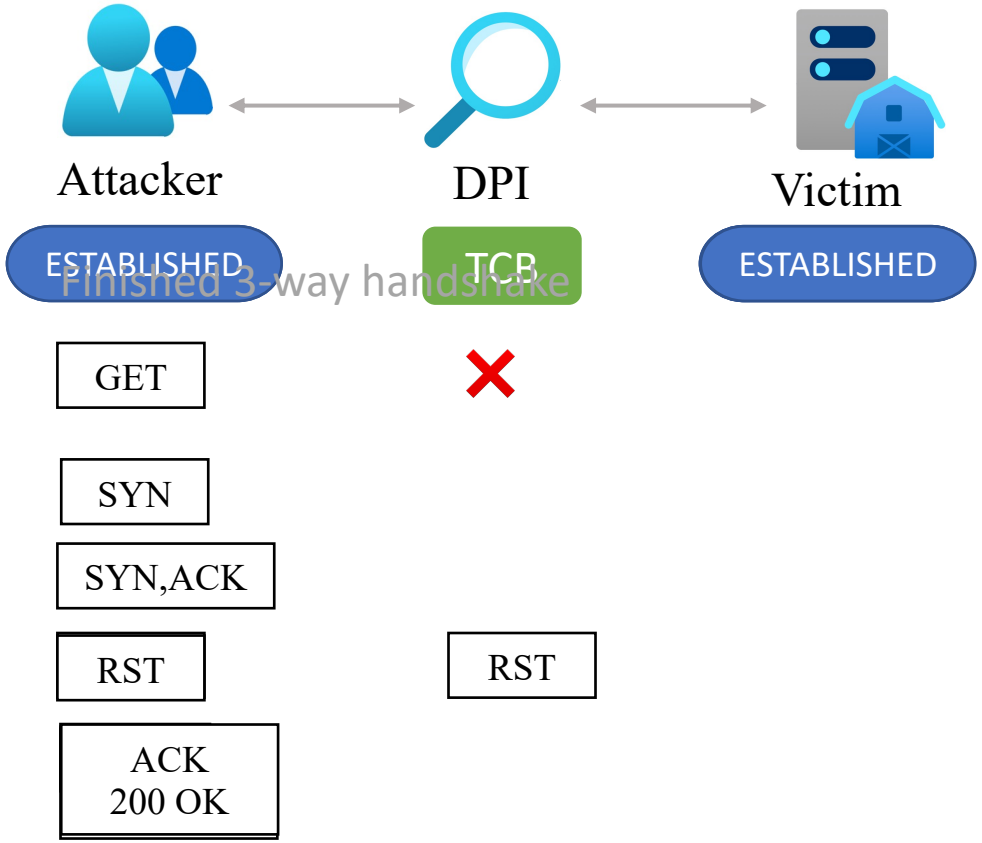
- Reasons
  - generality requirement
  - high throughput requirement
- Risks
  - ambiguous details in RFC files — vulnerable implementation
  - simplified protocol stacks — ignore complicated checks

Different Processing logic  Bypass Attacks

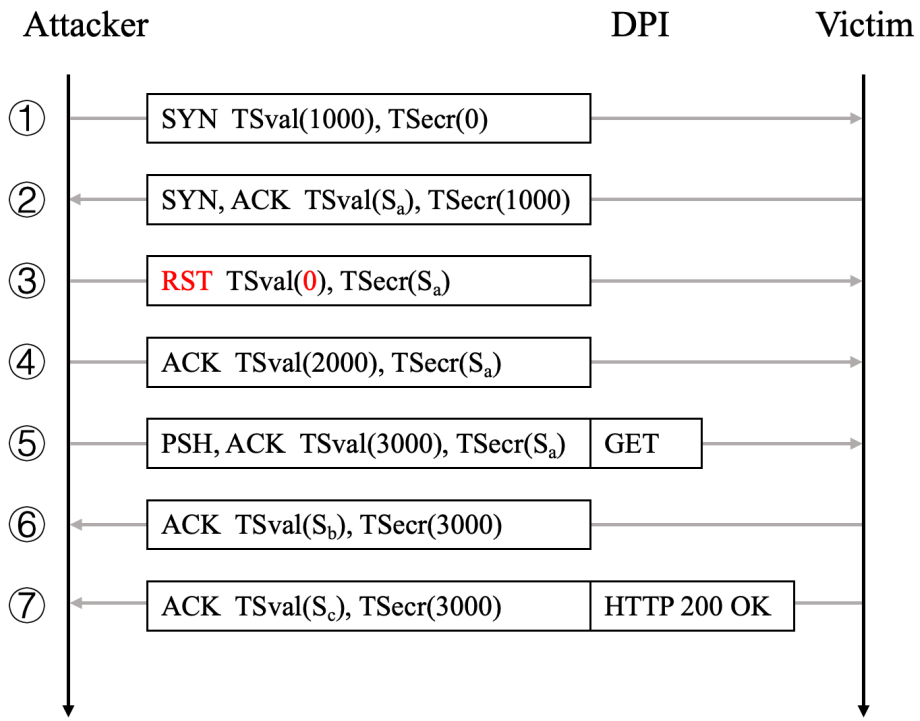
Bypass Attack Example



TCP Timestamps Option:
TSval: Timestamp Value
TSecr: Timestamp Echo Reply
PAWS Checking: TSval < last one -> ignore



Motivation Example



No.	Packet	Direction	Buggy DPI	Normal DPI
①	SYN TSval(1000) TSecr(0)	To Server	Sn_c: SYN_SENT Sn_s: LISTEN	Su_{st}: SYN_SENT
②	SYN, ACK TSval(S_a) TSecr(1000)	To Client	Sn_c: SYN_SENT Sn_s: LISTEN⇒SYN_RCVD	Su_{st}: SYN_SENT⇒SYN_RCVD
③	RST TSval(0) TSecr(S_a)	To Server	Sn_c: SYN_SENT⇒CLOSED Sn_s: SYN_RCVD	
④	ACK TSval(2000) TSecr(S_a)	To Server		Su_{st}: SYN_RCVD⇒ESTABLISHED
⑤	GET TSval(3000) TSecr(S_a)	To Server		
⑥	ACK TSval(S_b) TSecr(3000)	To Client		
⑦	HTTP 200 OK TSval(S_c) TSecr(3000)	To Client		

Identify and prioritize these packets/packet sequences

Challenge 1: Identify Abnormal State

1. Without a deep understanding of the DPI system
2. Without the protocol details

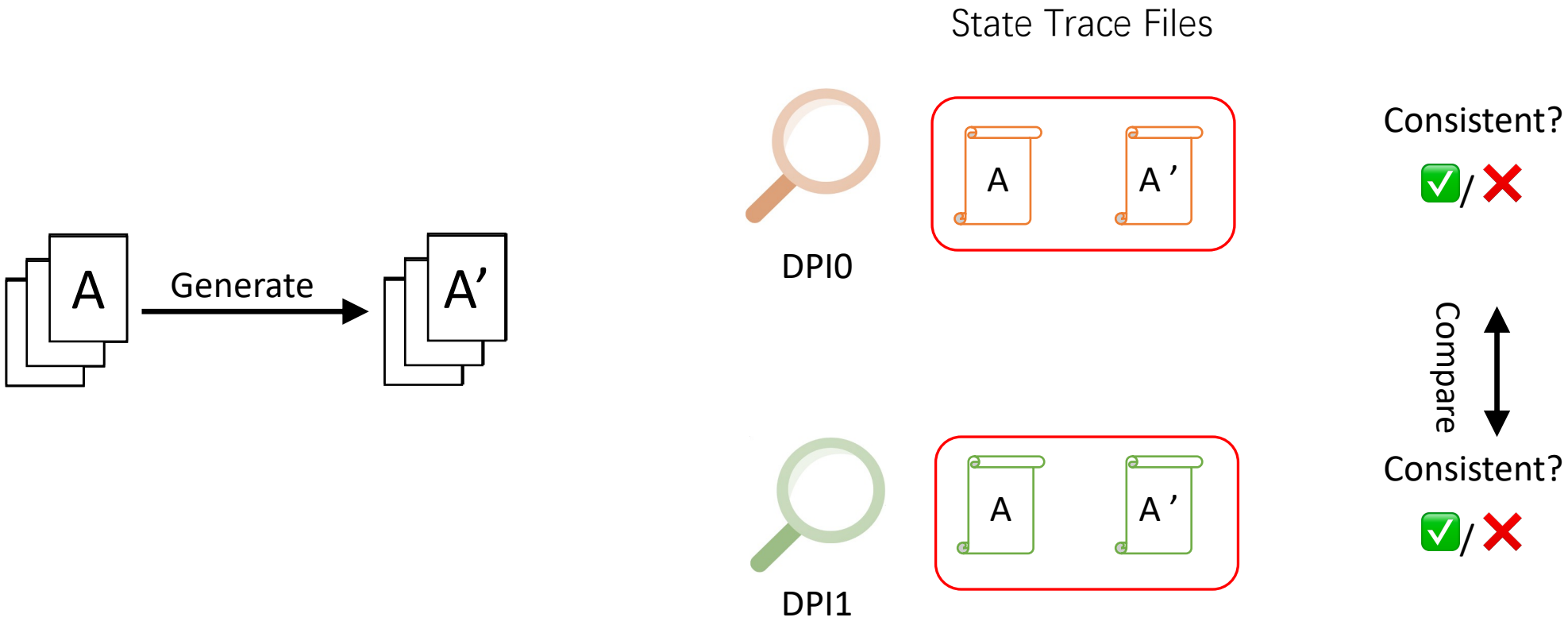
Solutions: monitor **multiple** DPI systems

- just one input ❌
- just one DPI system ❌

one pair of inputs ✅

multiple DPI systems ✅

Differential Analysis Algorithm



Differential Analysis Algorithm

Algorithm 1 Differential Analysis

```
1: DPI0StatAlt ← CmpDPI(parDPI0Stat, currDPI0Stat)
2: DPI1StatAlt ← CmpDPI(parDPI1Stat, currDPI1Stat)
3: if DPI0StatAlt is True xor DPI1StatAlt is True then
4:     evaluation ← best_score
5: else if DPI0StatAlt is True and DPI1StatAlt is True then
6:     evaluation ← good_score
7: else if DPI0StatAlt is False and DPI1StatAlt is False then
8:     evaluation ← moderate_score
9: end if
10: return evaluation
```

Challenge 2: State Extract

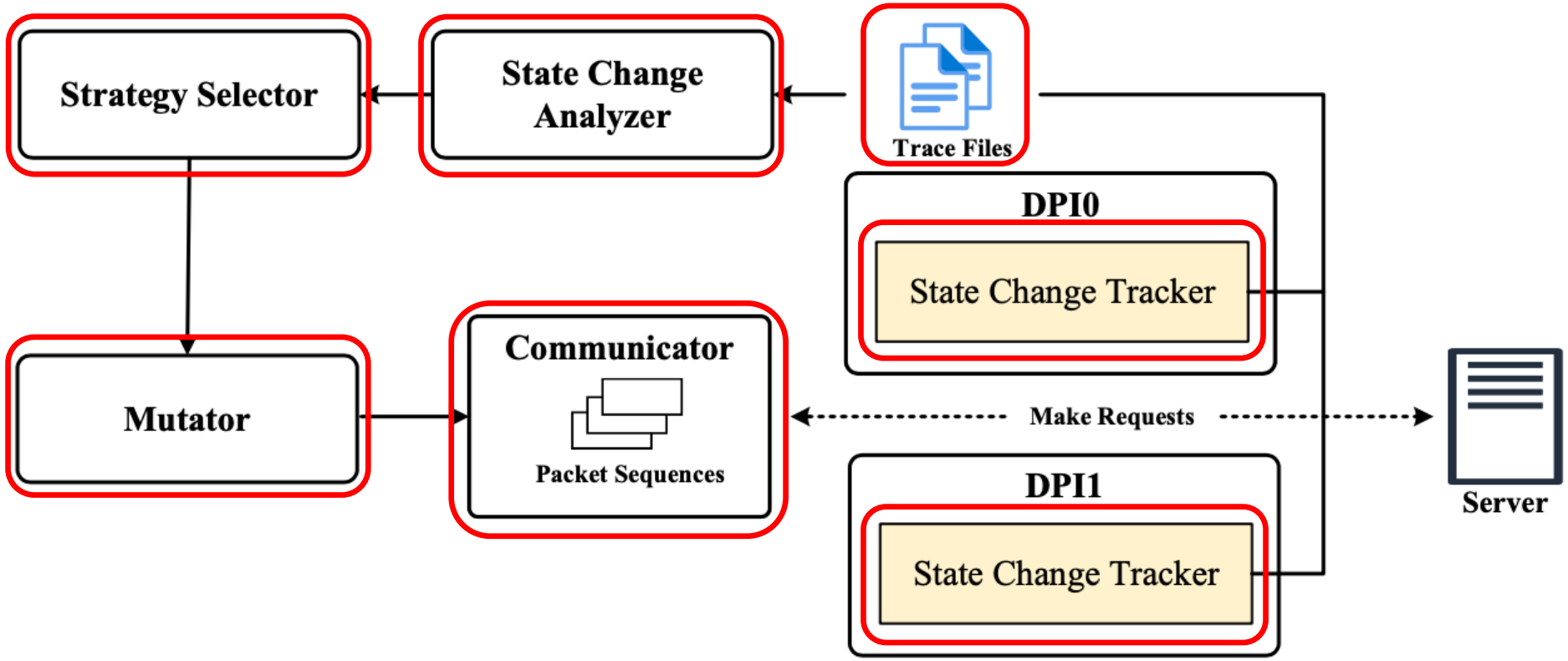
- State Instrumentation
  - Debug option
  - State processing functions

    e.g. accept/drop/return point

```
/* this env var uses the lower 32 bits of the flags: */
#define DEBUG_VARIABLE "SNORT_DEBUG"

#define DEBUG_INIT            0x0000000000000001LL
#define DEBUG_PARSER          0x0000000000000002LL
#define DEBUG_MSTRING         0x0000000000000004LL
#define DEBUG_PORTLISTS       0x0000000000000008LL
#define DEBUG_ATTRIBUTE       0x0000000000000010LL
#define DEBUG_PLUGIN          0x0000000000000020LL
#define DEBUG_PLUGBASE        0x0000000000000040LL
#define DEBUG_DECODE          0x0000000000000080LL
#define DEBUG_DATALINK        0x0000000000000100LL
#define DEBUG_CONFIGRULES     0x0000000000000200LL
#define DEBUG_RULES           0x0000000000000400LL
#define DEBUG_DETECT          0x0000000000000800LL
#define DEBUG_PATTERN_MATCH   0x0000000000001000LL
#define DEBUG_FLOW            0x0000000000002000LL
#define DEBUG_LOG             0x0000000000004000LL
#define DEBUG_FLOWBITS        0x0000000000008000LL
#define DEBUG_FILE            0x0000000000010000LL
#define DEBUG_CONTROL         0x0000000000020000LL
#define DEBUG_EXP             0x0000000080000000LL
```

StateDiver Architecture



Evaluation

- Q1: Bypasses found against real-world DPIs
- Q2: Efficacy of state-discrepancy guidance
- Q3: Performance compared with state-of-the-art evasion works

Table 4: Bypass Identified by StateDiver

Strategy Name	Illustration	Affected DPI		
		Snort	Snort++	Suricata
△ RST bad Timestamp	Send RST with invalid TCP timestamp option	✓	✓	✓
△ RST/ACK bad Timestamp	Send RST/ACK with invalid TCP timestamp option	✓	✓	✓
△ RST bad MD5	Send RST with invalid TCP MD5 option	✓	✓	
△ RST/ACK bad MD5	Send RST/ACK with invalid TCP MD5 option	✓	✓	
△ Timestamp gap	Send partial request with TCP timestamp option, then send the remaining request with TCP timestamp = last_timestamp + long gap (2147483648)	✓		
△ RST/ACK bad ACK number	Send RST with corrupted ACK number in ESTABLISHED state	✓	✓	
△ Multiple SYNs	Send another SYN with corrupted SEQ number in 3-way handshake		✓	
△ TCB Turnaround	Send SYN/ACK before sending the SYN packet		✓	
★ FIN with data	Send FIN with junk data in ESTABLISHED state, then send the request in TCP segments	✓		
★ SYN bad MD5	Send SYN with invalid TCP MD5 option in ESTABLISHED state, then send the request	✓	✓	
★ SYN Fragment	Send junk data in ESTABLISHED state, then send SYN with junk data, finally send the request	✓		
★ Fragment and Segment	Combination of multiple IP fragmentations and TCP segmentations	✓		
★ FIN bad ACK number	In SYN_RECV state, send FIN and ACK with same corrupted ACK number successively, then send the request		✓	
★ ACK bad ACK number and data with smaller Timestamp	Send ACK with corrupted ACK number in ESTABLISHED state, then send request with TCP timestamp = last_timestamp - gap (10)		✓	
★ ACK bad MD5 and data with smaller Timestamp	Send ACK with invalid TCP MD5 option in ESTABLISHED state, then send request with TCP timestamp = last_timestamp - gap (10)		✓	
★ PSH before SYN	Send PSH without data before 3-way handshake		✓	

△ means previous known strategies, and ★ means new strategies.

Q2: Contributions of State Discrepancies

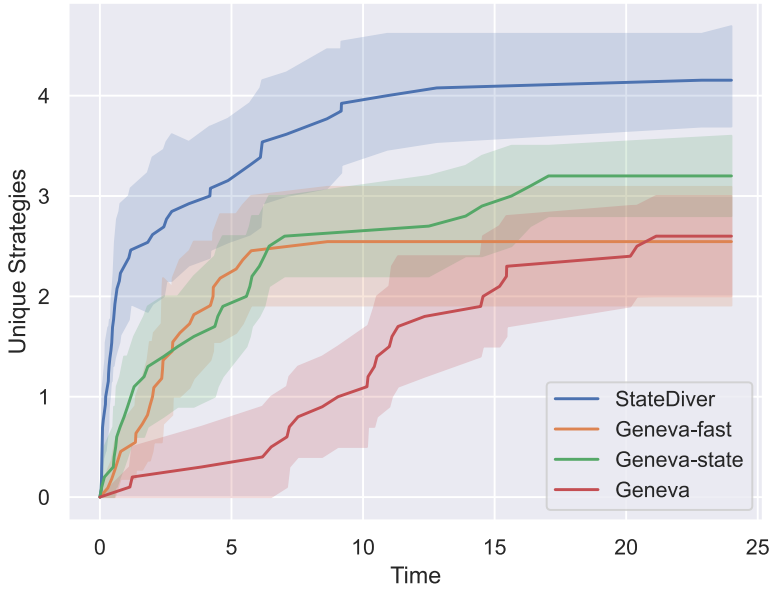
Tools	Illustration
StateDiver	Our tool using state-discrepancies guidance

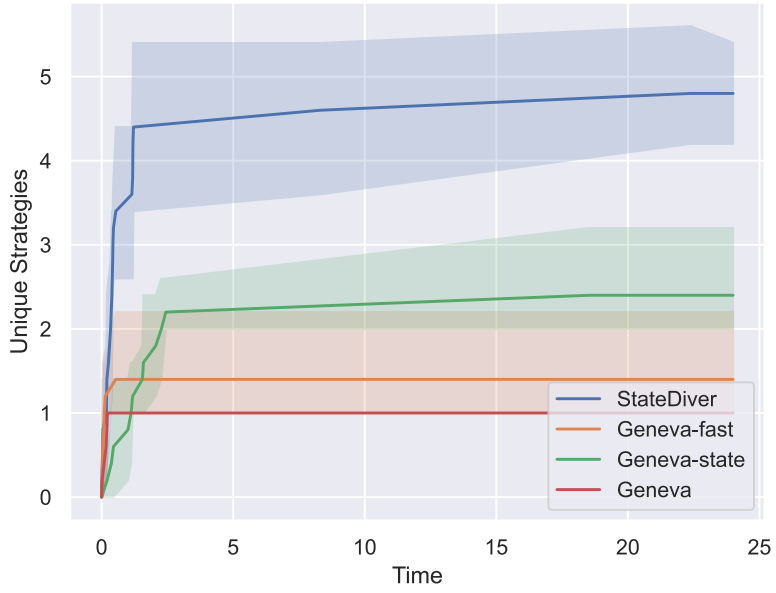
Q2: Contributions of State Discrepancies

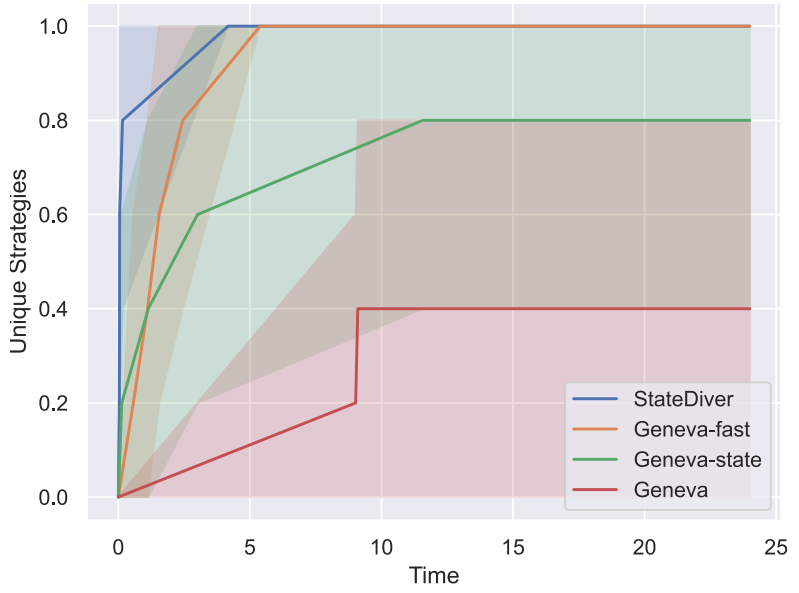
Tools	Illustration
StateDiver	Our tool using state-discrepancies guidance
Geneva (CCS'19)	Perform fuzzing on DPI leveraging server-side responses
Geneva-fast [added]	Geneva + local test enhancement
Geneva-state [added]	Geneva + state-discrepancies guidance

Run each tool 5 times, each time lasts 24h, on 3 DPI systems.

Q2: (1)(2) Unique Bypasses & Speed







Unique bypasses founded by evaluated fuzzers for 24h in Snort , Snort++, and Suricata respectively

Q2: (3) Unique State Transitions

LISTEN
SYN_SENT
SYN_RECV
ESTABLISHED
FIN_WAIT_1
CLOSE_WAIT
FIN_WAIT_2
LAST_ACK
TIME_WAIT
CLOSED
CLOSING

Target DPI	Variables	GENEVA	GENEVA-*state*	GENEVA-*fast*	STATEDIVER
Snort	Sn_c	15.8	19.8	25.0	29.4
	Sn_s	22.0	24.4	28.6	32.2
Snort++	Sp_c	12.6	20.4	15.8	25.4
	Sp_s	20.8	25.4	27.2	31.4
Suricata	Su_{st}	13.6	18.8	26.4	34.0

Q3: Comparison with State-of-the-art Tools

- State-of-the-art tools

  INTANG [IMC'17] lib·erate [IMC'17]

  Geneva [CCS'19] SymTCP [NDSS'20] Themis [CCS'21]

- Use attack dataset in Themis

  all the TCP-related evasion strategies in former works

Table 6: Prior Work's TCP-based Strategies and StateDiver Found on Suricata

Strategy Name	*INTANG*	*lib·erate*	*Geneva*	*SymTCP*	*Themis*	StateDiver
△ RST_Bad_MD5	✓		✓	✓	✓	✓
△ RST/ACK_Bad_MD5	✓		✓	✓	✓	✓
△ SEQ_Number_Before_ISN					✓	

Table 7: Prior Work's TCP-based Strategies and StateDiver Found on Snort

Strategy Name	*INTANG*	*lib·erate*	*Geneva*	*SymTCP*	*Themis*	StateDiver
△ RST_Bad_Timestamp	✓		✓	✓		✓
△ RST/ACK_Bad_Timestamp	✓		✓	✓		✓
△ In_Window_FIN				✓		
△ RST_Bad_MD5	✓		✓	✓		✓
△ RST/ACK_Bad_MD5	✓		✓	✓		✓
△ Timestamp_Gap				✓		✓
△ Urgent_data				✓		
△ In_Window_RST				✓		
△ MD5_FIN_ACK				✓		
△ MD5_FIN_Bad_ACK				✓		
△ Multiple_SYNs	✓			✓		
△ RST/ACK_Bad_ACK_Number			✓	✓		✓
△ RST_Bad_SEQ				✓		
△ No_ACK_Flag_FIN				✓		
△ RST_After_FIN					✓	
△ SYN+FIN		✓				
★ FIN_With_Data						✓
★ SYN_Bad_MD5						✓
★ SYN_Fragment			✓♯			✓
★ Fragment_And_Segment			✓♯			✓

Table 8: Prior Work's TCP-based Strategies and StateDiver Found on Snort++

Strategy Name	*INTANG*	*lib·erate*	*Geneva*	*SymTCP*	*Themis*	StateDiver
△ RST_Bad_Timestamp	✓		✓	✓		✓
△ RST/ACK_Bad_Timestamp	✓		✓	✓		✓
△ RST_Bad_MD5	✓		✓	✓	✓	✓
△ RST/ACK_Bad_MD5	✓		✓	✓	✓	✓
△ RST/ACK_Bad_ACK_Number			✓♯	✓		✓
△ Timestamp_Gap				✓	✓	
△ In_Window_RST				✓	✓	
△ RST_After_FIN					✓	
△ No_ACK_Flag_FIN				✓	✓	
△ In_Window_SYN					✓	
△ TCB_Turnaround	✓		✓		✓	✓
△ Multiple_SYNs	✓			✓	✓	✓
★ SYN_Bad_MD5						✓
★ FIN_Bad_ACK_Number						✓
★ ACK_Bad_ACK_Number_And_Data_With_Smaller_Timestamp						✓
★ ACK_Bad_MD5_And_Data_With_Smaller_Timestamp						✓
★ PSH_Before_SYN						✓

△ means previous known strategies
★ means new strategies

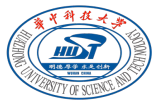
Conclusion: StateDiver

- End-to-end fuzzing framework uses state discrepancy between DPIs to discover bypass strategies
- Tested on 3 most famous open-source DPIs, and found 16 bypass strategies (8 new and 8 previously known)
- Help developers detect and fix implementation bugs

https://github.com/CGCL-codes/StateDiver

Thanks & Questions

zhangzhechang@hust.edu.cn



華中科技大學
Huazhong University of Science and Technology