TCP-Fuzz: Detecting Memory and Semantic Bugs in TCP Stacks with Fuzzing

Yong-Hao Zou[1], Jia-Ju Bai[1], Jielong Zhou[2], Jianfeng Tan[2], Chenggang Qin[2], Shi-Min Hu[1]

[1]Tsinghua University, [2]Ant Group

Background

- TCP protocol
  - Transport-layer network protocol
  - Carry over 85% network traffic
- Different implementations
  - Kernel-level TCP stacks
    - Linux, FreeBSD, NetBSD
  - User-level TCP stacks
    - mTCP, TLDK, F-Stack

Background

- TCP stacks are complex and error-prone
  - Rich functionalities: reliable transmission, congestion control
  - Complex state model
  - Various kinds of possible exceptions
- Common bugs in TCP stacks
  - Memory bugs: null-pointer dereference, use after free
  - Semantic bugs: RFC violation, bad handling of exceptional packet

Bug study

- Study of TCP stack commits
  - 87% of bug-fixing commits are related to **semantic bugs**

Time	FreeBSD		mTCP		TLDK	
	Memory	Semantic	Memory	Semantic	Memory	Semantic
2017	2	26	2	6	1	11
2018	9	51	0	4	0	4
2019	9	65	1	3	2	5
Total	20	**142**	3	**13**	3	**20**

Example

- Semantic bug fix in FreeBSD TCP stack
  - RFC 7323 violation

FILE: FreeBSD/sys/netinet/tcp_syncache.c

```
 int syncache_expand(...) {
        ......
+       /* RFC 7323 PAWS: If we have a timestamp on this segment and
+        * it is less than ts_recent, drop it.
+       if (sc->sc_flags & SCF_TIMESTAMP && to->to_flags & TOF_TS &&
+           TSTMP_LT(to->to_tsval, sc->sc_tsreflect)) {
+               SCH_UNLOCK(sch);
+               if ((s = tcp_log_addrs(inc, th, NULL, NULL))) {
+                       log(LOG_DEBUG, ...);
+                       free(s, M_TCPLOG);
+               }
+               return (-1);  /* Do not send RST */
+       }
        ......
 }
```

TCP stack features

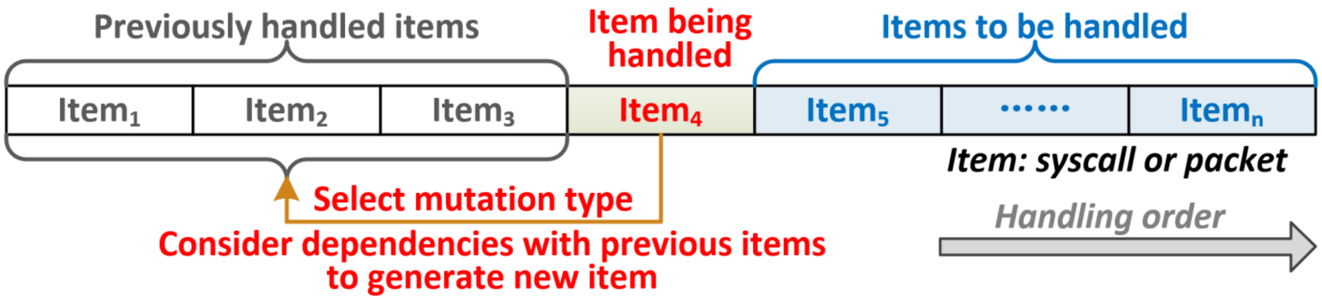
- ***F1: two-dimensional inputs with dependencies***
  - Inputs: syscalls, packets
  - Syscalls and packets have dependencies with each other
- ***F2: state model***
  - Basic model in RFC 793: 11 states and 20 state transitions
  - Real-world TCP stacks have many states and state transitions
- ***F3: semantic rules***
  - Stipulate how syscalls and packets should be handled
  - Explicit rules: defined in RFC documents
  - Implicit rules: no explicit description in RFC documents

Key Techniques

- ***F1: two-dimensional inputs with dependencies***
  - ***Dependency-based strategy*** to generate effective inputs
- ***F2: state model***
  - ***Transition-guided fuzzing approach*** to improve the coverage of states and state transitions
- ***F3: semantic rules***
  - ***Differential checker*** to detect semantic bugs

Dependency-based strategy

- Input sequence mutation
  - Item type: packet and syscall
  - Select mutation type: *deletion*, *addition*, *replacement* and *change*
  - Generate new items: consider dependencies with previous items

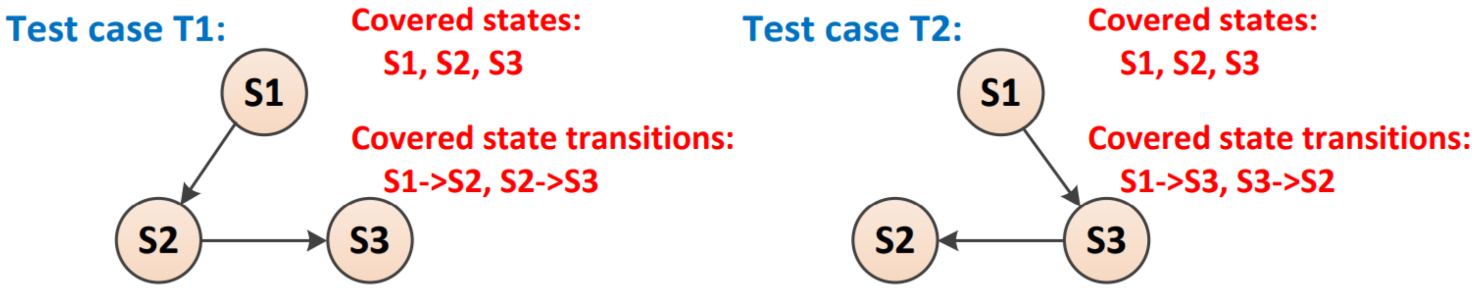


Dependency-based strategy

- Dependency
  - Refer to RFC documents and syscall-usage conventions
  - Type: syscall-syscall, packet-packet, syscall-packet
- Examples
  - *Syscall-syscall:* **socket**, **bind**, **listen** and **accept** are called in order when a connection is passive open
  - *Packet-packet*: the order and control flags of three-way handshake and four-way handshake packets are never changed
  - *Syscall-packet:* **accept** can be called only after the three-way handshake when a connection is passive open

Transition-guided fuzzing approach

- Old coverage metric: code coverage
  - Describe different states
  - Neglect different state transitions



Transition-guided fuzzing approach

- New coverage metric: ***Branch transition***

$$BrTran_n =< BrCov_n, BrCov_n - BrCov_{n-1} >$$

$BrTran_n$ *is the branch transition of* $input\_item_n$
$BrCov_n$ *is the branch coverage of* $input\_item_n$

Branch coverage0

Input item0:

BR1	BR2	BR3	BR4
1	0	1	0

Subtraction

Branch coverage1

Input item1:

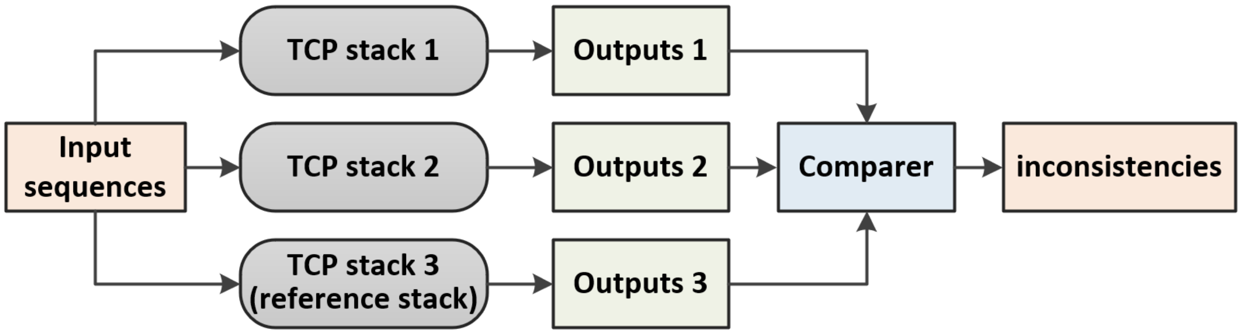
BR1	BR2	BR3	BR4
0	1	0	1

Branch transition1

BR1	BR2	BR3	BR4
0	1	0	1
BRC1	BRC2	BRC3	BRC4
-1	1	-1	1

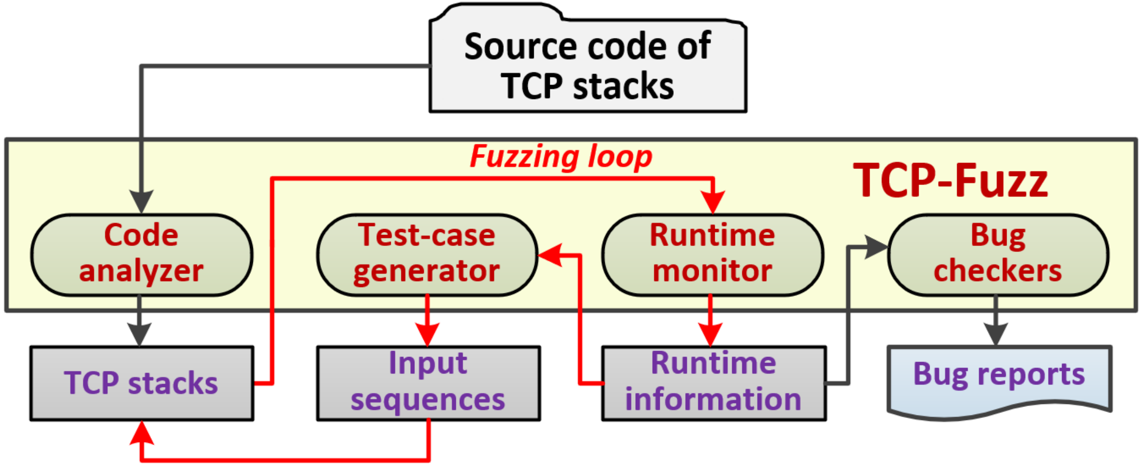
Differential checker

- Basic idea
  - Different TCP stacks obey identical semantic rules and produce identical or similar outputs for the same inputs
- Design
  - Provide the same input sequences to different TCP stacks and compare their outputs



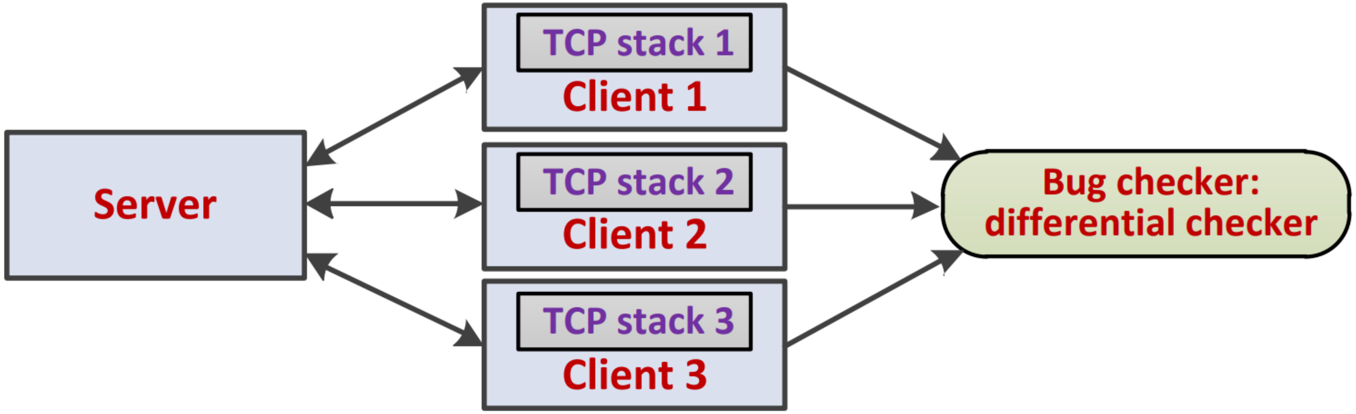
Framework

- TCP-Fuzz
  - Novel fuzzing framework for testing TCP stacks
  - Integrate the three key techniques
  - Detect memory and semantic bugs



Deployment

- Server-client mode
  - Server: generate test cases and validate data
  - Client: run the tested TCP stack with third-party sanitizers



Evaluation

- Experimental setup
  - Three user-level and two kernel-level TCP stacks
  - Each TCP stack is tested for 48 hours

Type	TCP stack	Version	LOC
User-level	TLDK	v2.0	15K
	F-Stack	Commit 8d21adc	25K
	mTCP	Commit 0463aad	18K
Kernel-level	FreeBSD	v12.1	171K
	Linux	v5.6	169K

Testing coverage and found bugs

- Branch transitions > branches
- 8 memory bugs and 48 semantic bugs
- 40 bugs have been confirmed

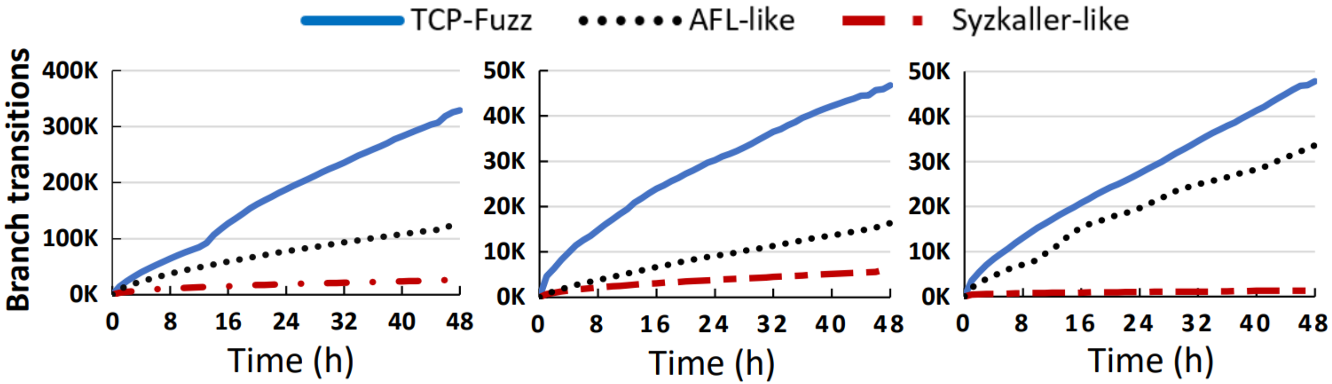
Stack	Testing coverage		Found bugs	
	Branch	**Transition**	**Memory/Semantic**	**Confirmed/Fixed**
TLDK	1.3K	329.4K	2/26	28/19
F-Stack	7.5K	46.8K	1/6	6/1
mTCP	1.2K	47.9K	5/9	0/0
FreeBSD	-	-	0/6	5/2
Linux	-	-	0/1	1/1
Total	**10.0K**	**424.1K**	**8/48**	**40/23**

Compared to state-of-the-art fuzzers

- Two classical and widely-used fuzzing approaches
  - AFL-like: only generates packet sequences
  - Syzkaller-like: only generates syscall sequences
- Three open sourced protocol fuzzing approaches
  - Boofuzz: github.com/jtpereyda/boofuzz
  - Fuzzotron: github.com/denandz/fuzzotron
  - AFLNet: github.com/aflnet/aflnet [ICST'20]

Compared to state-of-the-art fuzzers

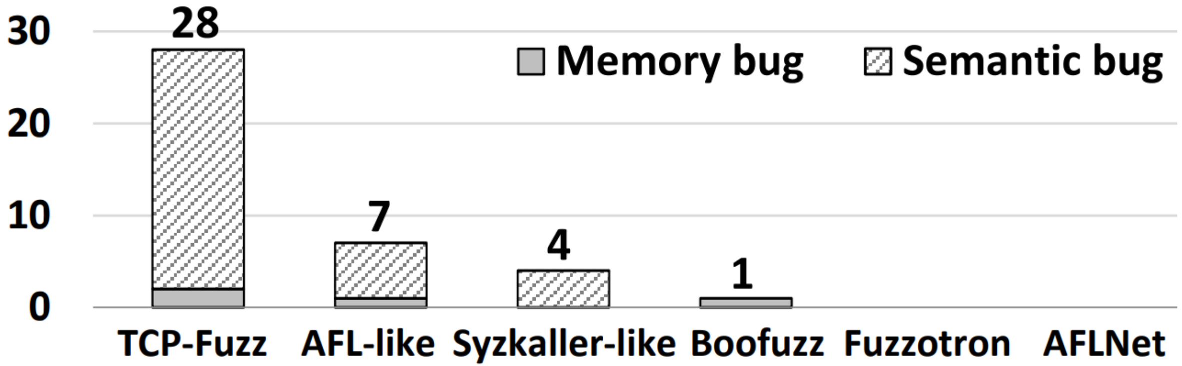
- Testing coverage
  - TCP-Fuzz covers more branch transitions than AFL-like and Syzkaller-like fuzzers



(a) TLDK (b) F-Stack (c) mTCP

Compared to state-of-the-art fuzzers

- Bug detection
  - TCP-Fuzz finds more bugs than other fuzzers



Conclusion

- TCP stacks are complex and error-prone
- TCP-Fuzz
  - ***Dependency-based strategy*** to generate effective inputs
  - ***Transition-guided fuzzing approach*** to improve the coverage of states and state transitions
  - ***Differential checker*** to detect semantic bugs
- Find 56 real bugs in 5 widely-used TCP stacks
- Outperform state-of-the-art fuzzers

Thanks

Yong-Hao Zou

E-mail: zouyh19@mails.tsinghua.edu.cn

清華大學 Tsinghua University

蚂蚁集团 ANT GROUP