OWL: Understanding and Detecting Concurrency Attacks

Shixiong Zhao , Rui Gu, Haoran Qiu , Tsz On Li , Yuexuan Wang

Heming Cui , and Junfeng Yang

Computer Science, The University of Hong Kong Computer Science, Columbia University

What are Concurrency Attacks?

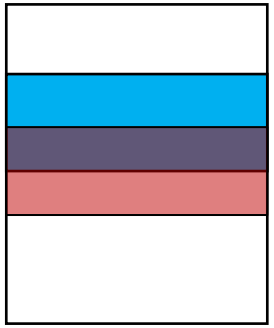
- Concurrency bug caused by data race

```
//Thread 1
int inotify_handle_event(…)
  ...
  len = sizeof(ievent)+strlen(inode->file_name)
  ievent event =kmalloc(len, …);
  ...
```

```
// Thread 2

rename(inode->file_name,longer_name)
```

```
if(len)
  strcpy(event->name, inode->file_name);
```

Cras h!!!



Kernel Heap

2017-April-20	We found this bug
2017-May-31	Reported to Kernel Patchwork by Alibaba.inc

What are Concurrency Attacks?

- Root privilege escalation leveraging the concurrency bug

```
//Thread 1
int inotify_handle_event(…)
  ...
  len = sizeof(ievent)+strlen(inode->file_name)
  ievent event =kmalloc(len, …);
  ...

if(len)
  strcpy(event->name, inode->file_name);
```

```
// Thread 2

rename(inode->file_name,longer_name)
```

```
//Thread 3
...

secattr = kmalloc(sizeof(netlbl_secattr))
secattr.free_func_ptr = free_func

...

free(secattr-
```

Root!!!

Kernel Heap

2017-April-20	We found this bug
2017-May-31	Reported to Kernel Patchwork by Alibaba.inc
2017-July-02	We reported this privilege escalation to RedHat
2017-July-06	Confirmed by Linux, RedHat in Linux 3.0 - 4.9.3
2017-July-06	Assigned as **CVE-2017-7533**
2017-July-07	Fixed in Linux Kernel 4.11.0
2017-Dec-01	Confirmed by Android and reward from Google

What are Concurrency Attacks?

- Concurrency Attacks: Attackers can leverage corrupted memory caused by concurrency bugs to conduct severe security consequences to systems
  - E.g., privilege escalation, code injection, bypassing security check
  - Much more severe than the traditional consequences caused by concurrency bugs (e.g., program crash, dead lock)
  - Much harder to tackle than TOCTTOU attacks (file system only)

- Concurrency attacks are realistic threats in many software
  - Dirty Cow/CVE-2016-5195: Data Race in Linux Kernel ➔ Privilege Escalation
  - CVE-2016-3841: Data race ➔ Privilege Escalation
  - CVE-2017-15649: Data race ➔ Privilege Escalation
  - CVE-2015-8963: Data race ➔ Privilege Escalation
  - CVE-2017-7533: Data race in Linux Kernel ➔ Privilege Escalation
  - CVE-2016-1000324: Data race in SSDB➔ Use after free
  - CVE-2017-12193: Data race in Linux Kernel ➔ DOS
  - Apache-25520: Data race in Apache ➔HTML Integrity Violation
  - Apache-45590: Data race in Apache ➔ DOS

Questions I

- Can existing concurrency bug detecting tools effectively detect concurrency attacks?

Tools		Platform	Application	
TSAN	Google's user-space data race sanitizer	Linux	Chrome	Web browser
KTSAN	Google's Kernel-space data race sanitizer		Apache	Web server
SKI	[SKI, OSDI' 2014]		Linux Kernel	Operating system kernel
Valgrind	Valgrind data race dynamic analysis tool		MySQL	Relational database
			SSDB	Key-value store library

Can existing tools effectively detect concurrency attacks?

- Our answer is NO
- Only a subset of reports generated are real concurrency bugs (there exists thread interleaving to trigger the bugs)
  - We built a tool using LLVM's LLDB tool to validate thread interleaving of concurrency bugs
  - Out of 31.8K reports , only 1.8K are real data races, including benign (e.g., spin-lock) and harmful data races
- Only a subset of real concurrency bugs can be used to conduct attacks
  - Only 182 of 1.8K real concurrency bug reports are reported for potential attack by OWL

Questions II

- What are the requirements to effectively detect concurrency attacks?

	Linux		Windows	Darwin	FreeBSD
Privilege Escalation	4	**4**	3	1	2
Inject Code	2	**2**	0	0	0
Bypass security check	1	**1**	0	0	1
Violate Integrity	1	**1**	1	1	0
DoS/Crash	9	**4**	2	2	1

- Answer: We summarize two requirements.

Requirement I

- Need to track how the corrupted memory caused by concurrency bugs propagates
  - In 14 out of 31 attacks we studied, concurrency bug triggering instructions and attack inducing instructions span across different functions
  - E.g., CVE-2015-1125 in the Libsafe library

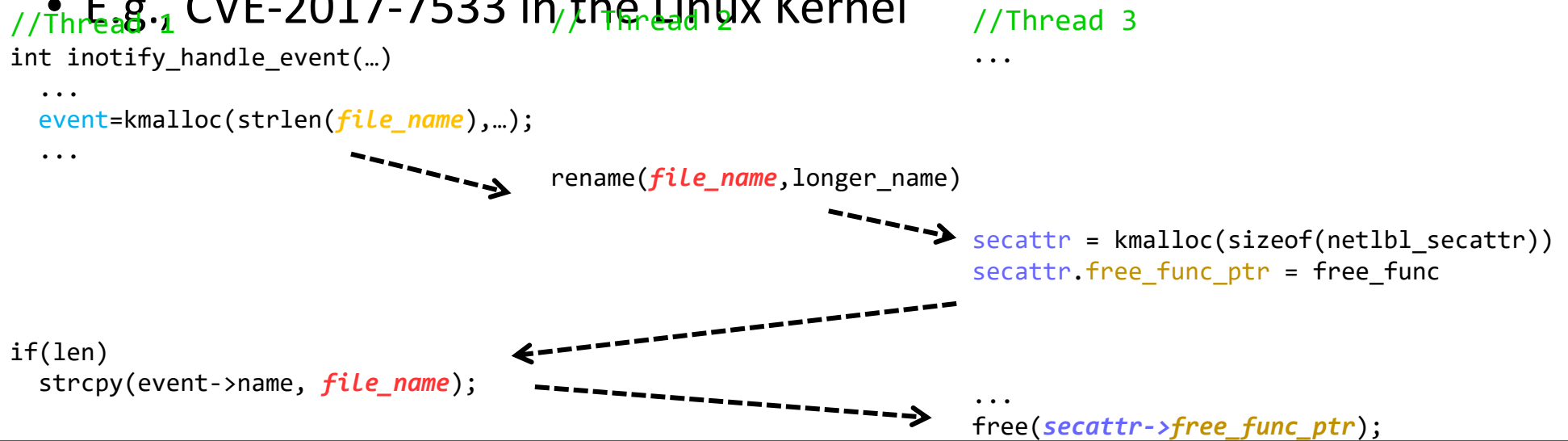
```
// Thread 1
117 uint stack_check(...) {
...
...
145   if(dying)
146     return 0; //Bypass check.
...   ...; //Check overflow.
    }

151 char *libsafe_strcpy(dst,src)

...

164   if(stack_check(dst)==0)

165     return strcpy(dst,src);
```

```
// Thread 2
1636 libsafe_die(){
...
1640   dying = 1;
     }
```

Stack Overflow

Requirement II

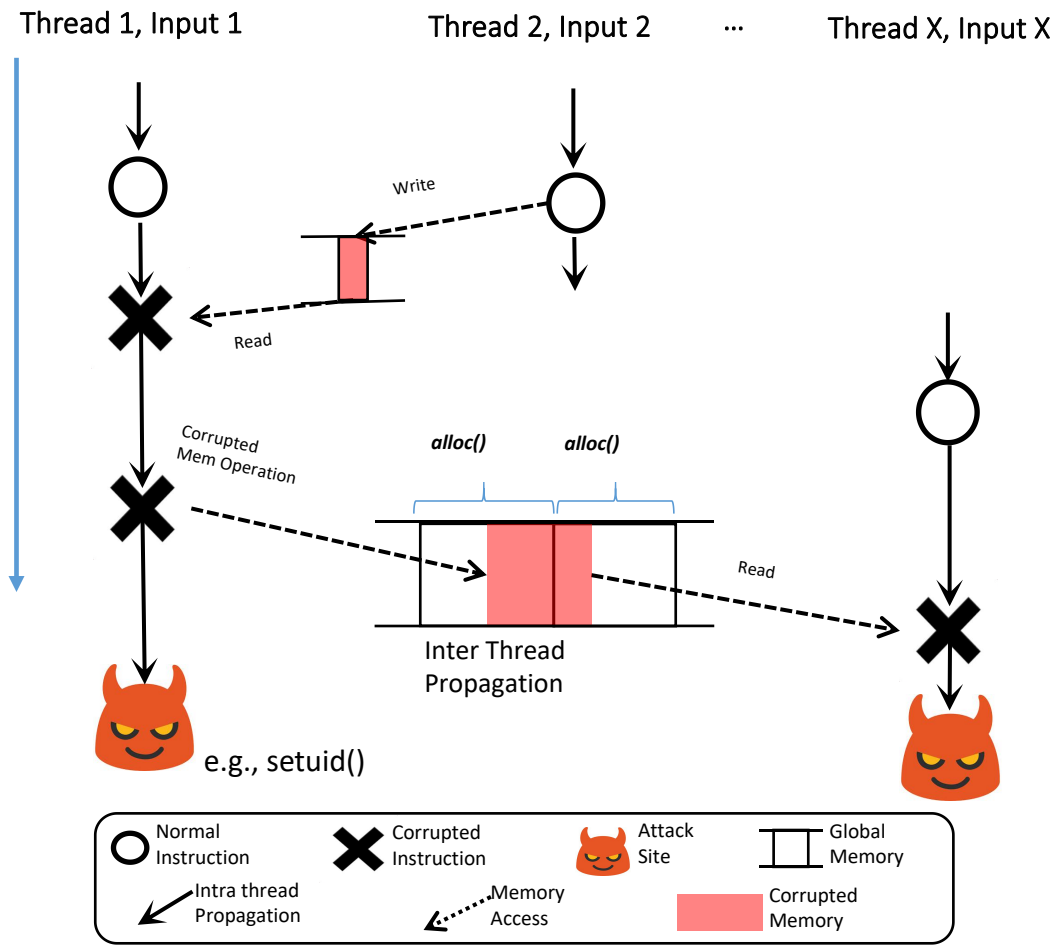
- In addition to the threads and their inputs to trigger a concurrency bug, extra threads and their inputs are often needed to trigger an attack
  - In 10 out of 12 concurrency attacks we have source code, corrupted memory propagates to additional threads and we need extra inputs to conduct concurrency attacks
  - E.g., CVE-2017-7533 in the Linux Kernel



Outline

- Concurrency Attack Model
- OWL: The first effective and general concurrency attack detecting tool
- Evaluation Results
- Summary and Future Work

Model

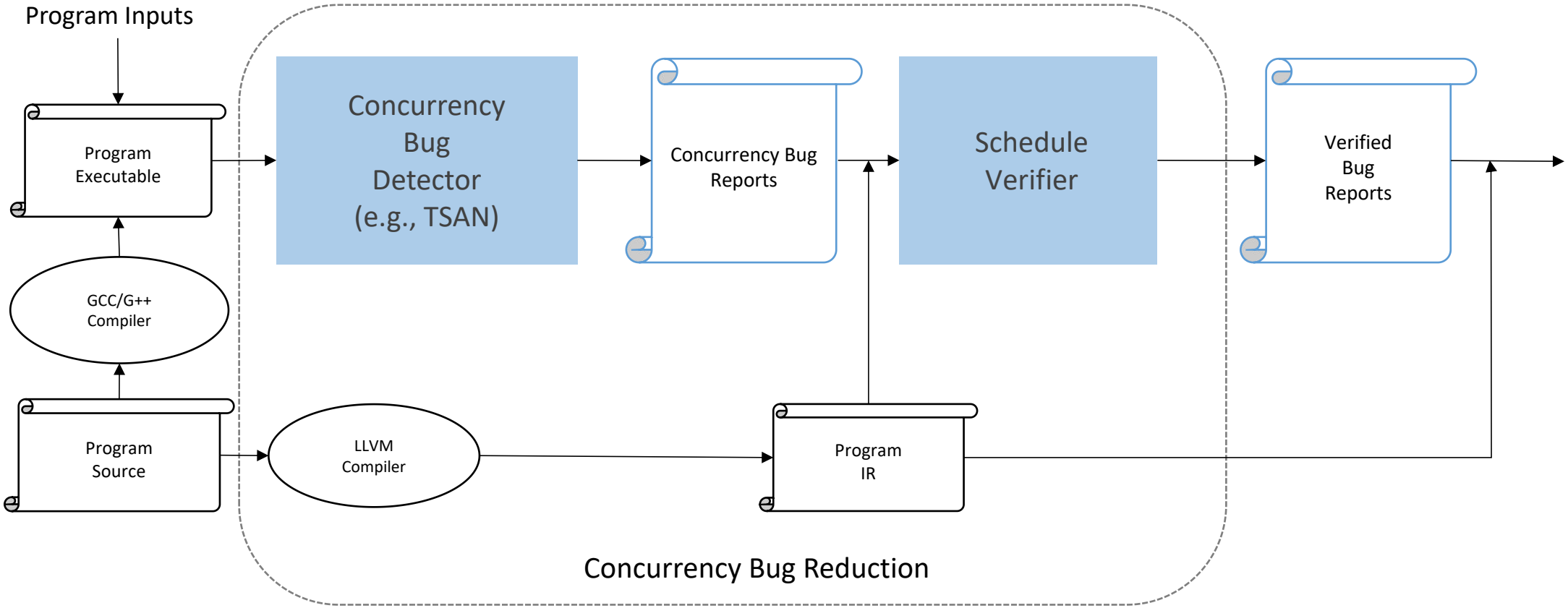


- How to find concurrency bugs that may cause concurrency attacks?
  - Solution: Attack-inducing Bugs are already contained in bug Reports produced by existing tools (e.g., TSAN)

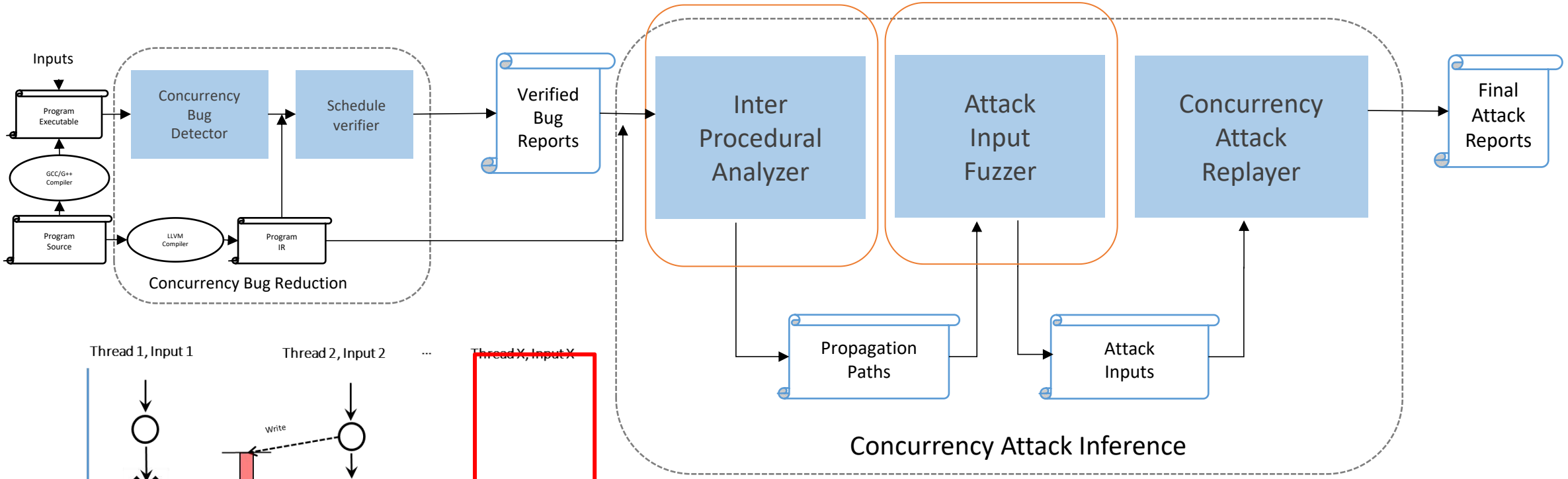
- (Requirement 1) How to analyze foot-print of corrupted memory and find the attack triggering sites?
  - Observation: Attack triggering sites are often explicit operations in the program code: privilege operation(e.g., setuid()), memory operation, file operation, …

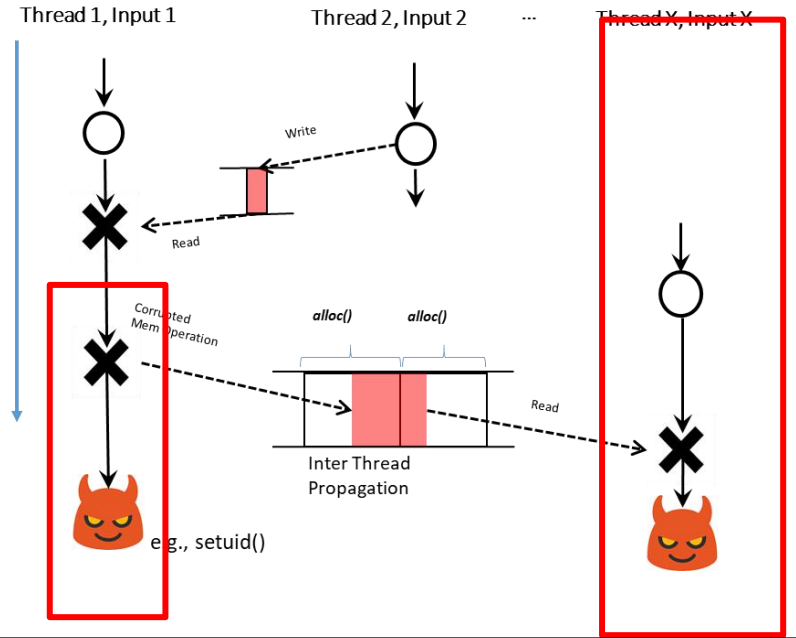
- (Requirement 2) How to infer extra inputs for conducting concurrency attacks?
  - Challenge: Symbolic execution (e.g., [UC-KLEE, security 15]) is hard to infer inputs in large software
  - Observation: Existing test suites for real-world software already have a reasonable code coverage. We can leverage these test suites to generate inputs and we analyze buffer overflow automatically at runtime

OWL: Tool for detecting Concurrency Attacks



OWL: Tool for detecting Concurrency Attacks





OWL: Tool for detecting Concurrency Attacks

- Inter-procedural Analyzer
  - Maintain a corrupted instruction set (instructions that may be affected by corrupted memory)
    - Initial – instructions corrupted by concurrency bugs
  - Statically traverse code (including callees of functions) to update the set
  - Report potential attacks when encounters attack sites (e.g., setuid())
- Attack Input Fuzzer: Bug-inducing inputs → attack-inducing inputs
  - Only infer extra attack-inducing inputs when the bugs cause memory overflow
  - Monitor the global memory layout (e.g., kmalloc32()) at runtime and record the potential attack-inducing inputs that allocate memory next to the memory corrupted by concurrency bug
  - Report these attack-inducing inputs when a buffer overflow attack is suspected

OWL Implementation Details

- Support Both Linux Kernel and User-space Programs
- Integrate 4 race detectors :
  - TSAN, ValGrind for User-space
  - SKI, KTSAN for Kernel-space
- Static analysis based on LLVM
- Thread interleaving verification based on LLVM’s LLDB
- Attack input information collection based on Kprobe, Uprobe in Linux

Evaluation Setup

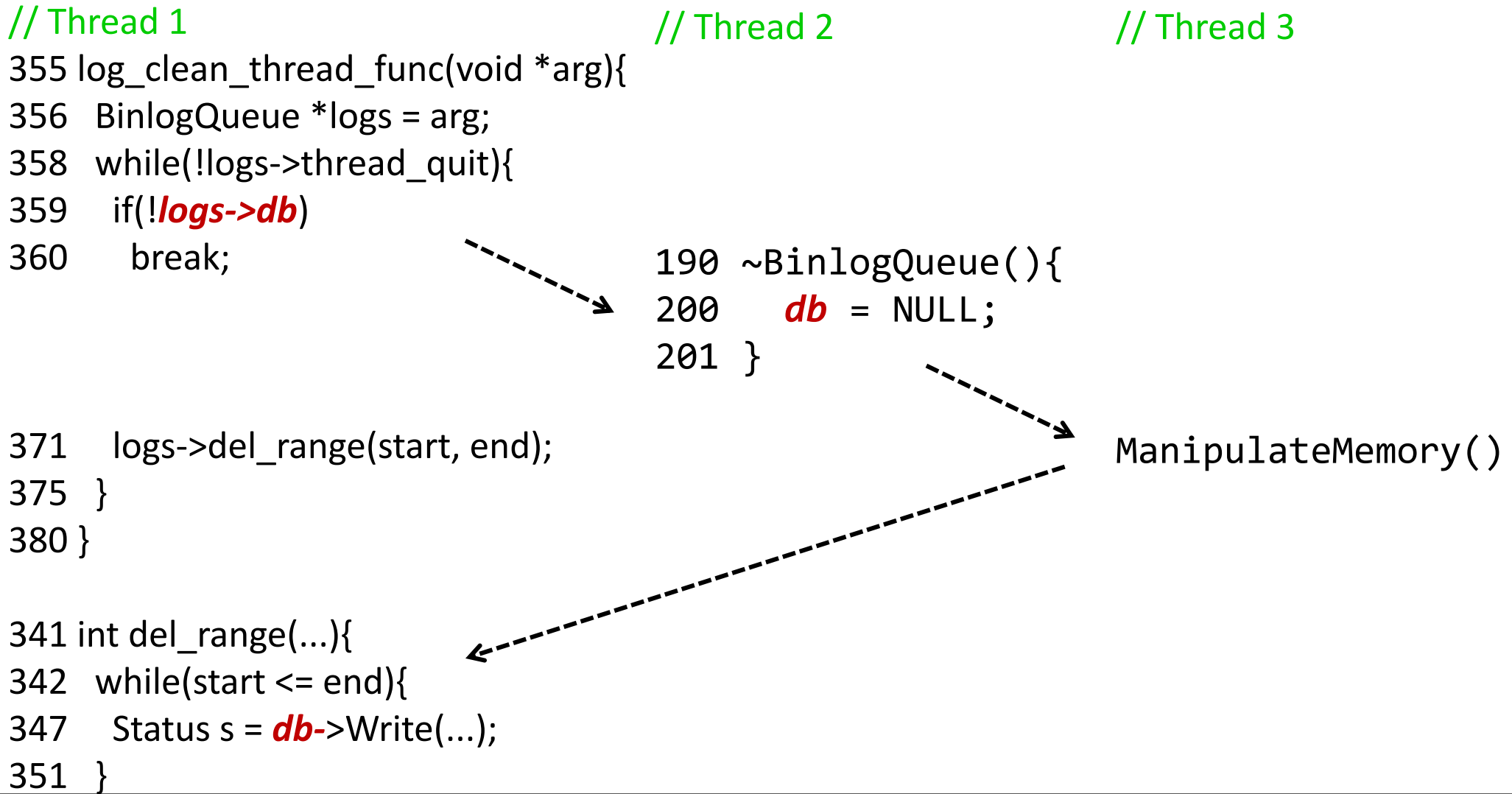
- Evaluated 5 large scale software and their test suites
  - Test suites have reasonable code coverage
- 7 exploitation scripts of known concurrency attacks

Program Name	Test Suites
Linux	Trinity (system call benchmark)
SSDB	SSDB benchmark
Libsafe	Attack exploitation scripts
MySQL	DBT2 Benchmark
Chrome	Octane 2.0
Apache	Apache Bench

OWL Detection Results

- Detected 5 new concurrency attacks
  - 3 confirmed and fixed by the corresponding developers
    - CVE-2017-12193, DOS attack against Linux Kernel
    - CVE-2017-7533, root privilege escalation in Linux Kernel
    - CVE-2016-1000324, use-after-free in Linux Kernel
  - 2 attacks detected from well-studied bugs
    - Apache-25520, HTML integrity violation in apache web server
    - Apache-46215, DOS attack against apache web server
- Evaluated 7 known attacks and detected all of them without missing anyone
  - Covering programs: kernel, chrome, apache, mysql, libsafe

Example of new concurrency attack



How well can OWL reduce benign reports?

- OWL reduced 94.1% race reports to 182 final reports
- 12 attacks are confirmed with moderate manual inspection on the 182 reports

Name	Race reports	Final reports by OWL	Real attacks
Apache	715	10	3
Linux	24645	36	4
Chrome	1715	115	1
Libsafe	3	3	1
MySQL	1123	16	2
SSDB	12	2	1
Total	3.18K	182	12

How well can OWL find Extra Inputs?

- OWL is more precise to find extra inputs when memory allocation functions are more diverse, and vice versa.

Name	Mem Alloc Type	Test Cases #	Attack Inputs #	Attacks #
Apache	apr_palloc	243	58	3
Linux	kmalloc32	1153	29	4
Chrome	partalloc	432	123	0
Libsafe	malloc	4	4	1
MySQL	sql_alloc	814	409	2
SSDB	malloc	2	2	0
Total	n/a	2648	625	10

Summary and Future Work

- Concurrency attacks are severe threats but ignored in the past
- Research on concurrency attacks is emergent
  - E.g., [ACIDRain, SIGMOD' 17] detects concurrency attacks for SQL database
- OWL: The first effective and general concurrency attack detecting tool
- Future work is exciting
  - Add more types of concurrency bug detectors
  - Effective symbolic execution methods on finding attack-inducing inputs
  - Multi-process concurrency attacks