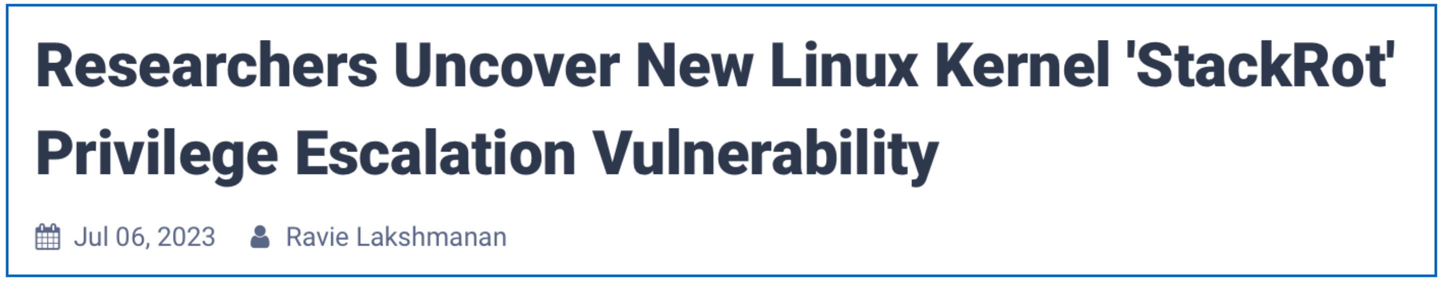
Refcount-guided Fuzzing for Exposing Temporal Memory Errors in Linux Kernel

Shuangpeng Bai Zhechang Zhang Hong Hu

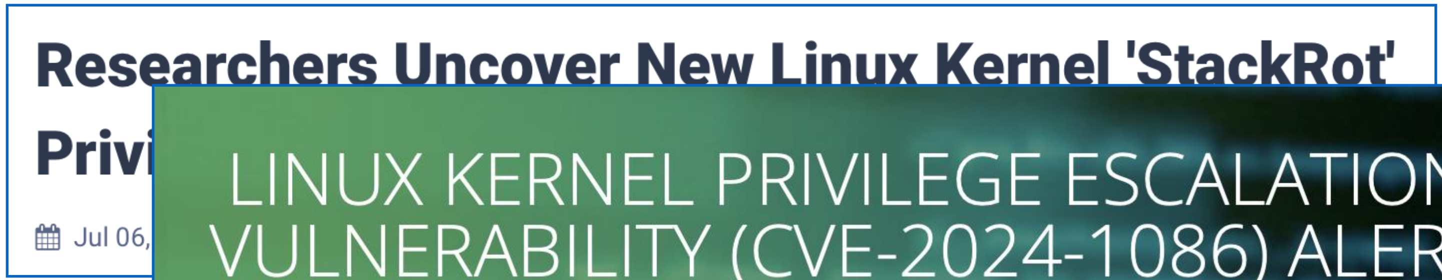
Background—Kernel Use-After-Free Bugs



Researchers Uncover New Linux Kernel 'StackRot' Privilege Escalation Vulnerability

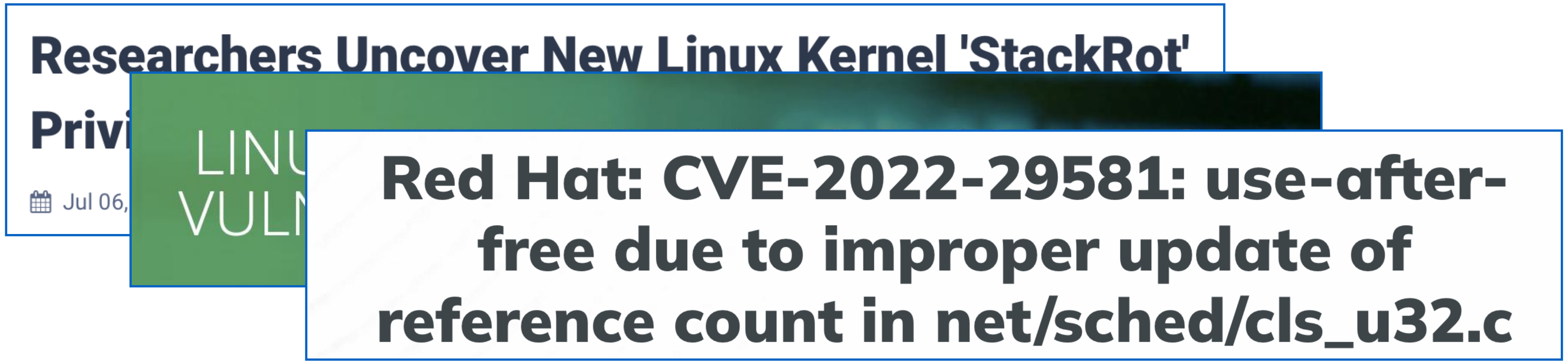
Jul 06, 2023 Ravie Lakshmanan

Background—Kernel Use-After-Free Bugs


Researchers Uncover New Linux Kernel 'StackRot'
Privi
Jul 06,

LINUX KERNEL PRIVILEGE ESCALATION VULNERABILITY (CVE-2024-1086) ALERT

Background—Kernel Use-After-Free Bugs



Researchers Uncover New Linux Kernel 'StackRot'

Privi

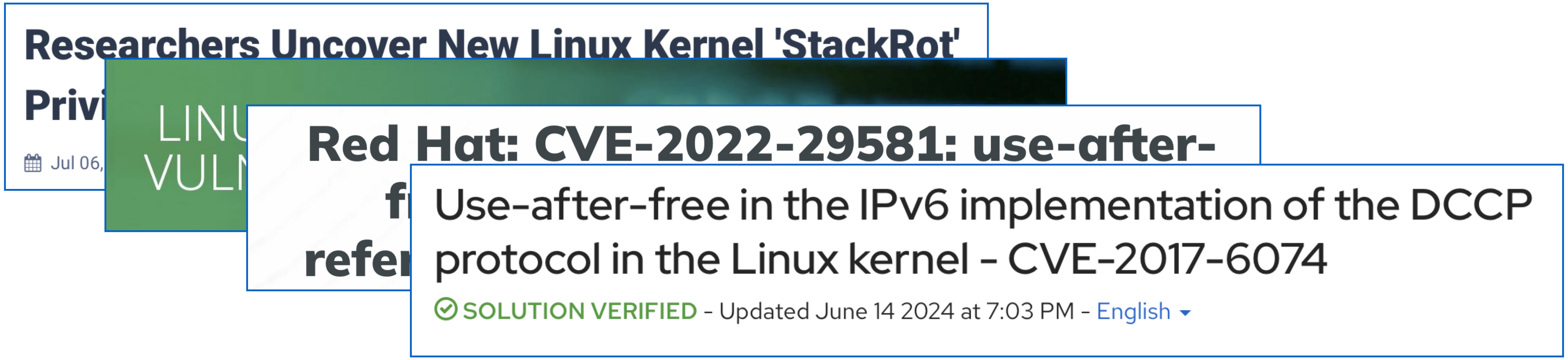
Jul 06,

LINU

VULN

Red Hat: CVE-2022-29581: use-after-free due to improper update of reference count in net/sched/cls_u32.c

Background—Kernel Use-After-Free Bugs



Researchers Uncover New Linux Kernel 'StackRot'

Privi

Jul 06,

LINU

VULN

Red Hat: CVE-2022-29581: use-after-

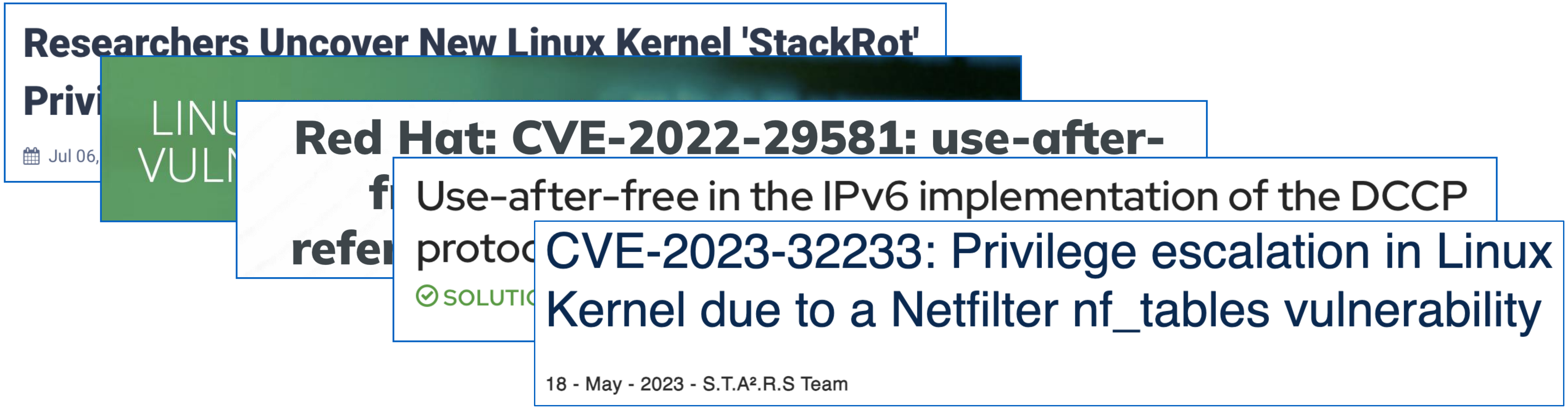
fr

refer

Use-after-free in the IPv6 implementation of the DCCP protocol in the Linux kernel - CVE-2017-6074

SOLUTION VERIFIED - Updated June 14 2024 at 7:03 PM - English

Background—Kernel Use-After-Free Bugs



Researchers Uncover New Linux Kernel 'StackRot'

Privi

Jul 06,

LINU
VULN

Red Hat: CVE-2022-29581: use-after-

fr

refer

Use-after-free in the IPv6 implementation of the DCCP

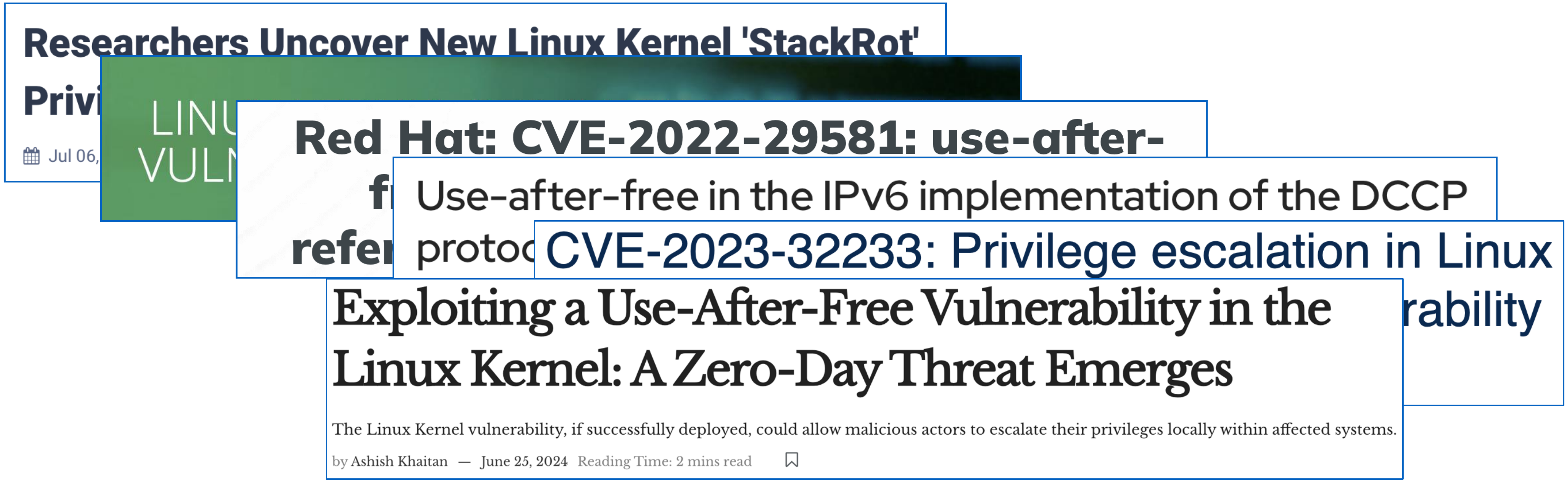
protoc

SOLUTIO

CVE-2023-32233: Privilege escalation in Linux Kernel due to a Netfilter nf_tables vulnerability

18 - May - 2023 - S.T.A².R.S Team

Background—Kernel Use-After-Free Bugs


Researchers Uncover New Linux Kernel 'StackRot'

Privi

Jul 06,

LINU

VULN

Red Hat: CVE-2022-29581: use-after-

fr

refe

Use-after-free in the IPv6 implementation of the DCCP

protoc

CVE-2023-32233: Privilege escalation in Linux

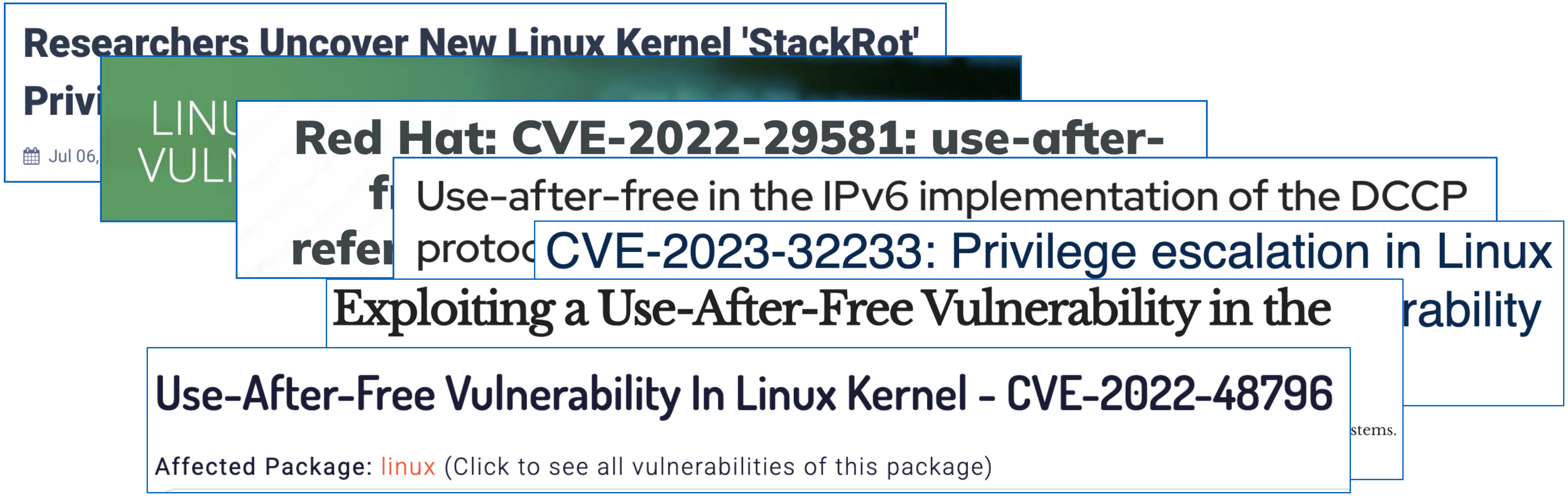
rability

Exploiting a Use-After-Free Vulnerability in the Linux Kernel: A Zero-Day Threat Emerges

The Linux Kernel vulnerability, if successfully deployed, could allow malicious actors to escalate their privileges locally within affected systems.

by Ashish Khaitan — June 25, 2024 Reading Time: 2 mins read

Background—Kernel Use-After-Free Bugs


Researchers Uncover New Linux Kernel 'StackRot'

Privi

Jul 06,

LINU
VULN

Red Hat: CVE-2022-29581: use-after-

fr

refer

Use-after-free in the IPv6 implementation of the DCCP

protoc

CVE-2023-32233: Privilege escalation in Linux

rability

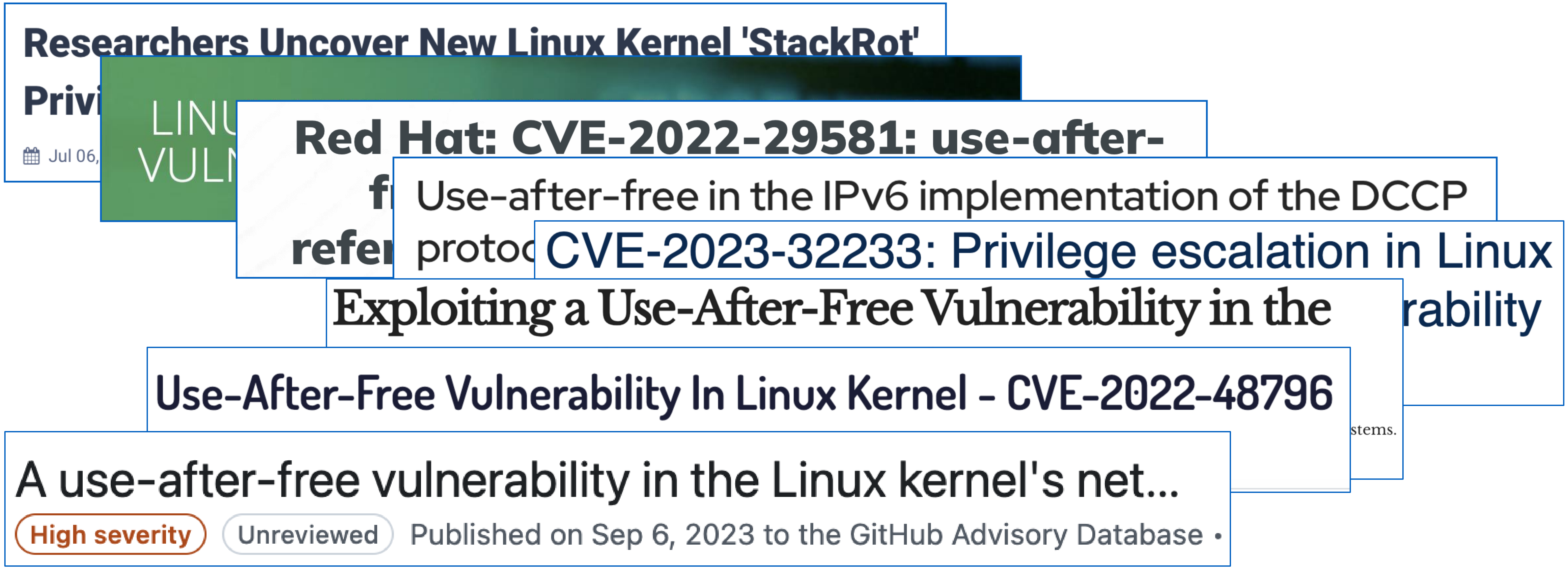
Exploiting a Use-After-Free Vulnerability in the

stems.

Use-After-Free Vulnerability In Linux Kernel - CVE-2022-48796

Affected Package: linux (Click to see all vulnerabilities of this package)

Background—Kernel Use-After-Free Bugs



Researchers Uncover New Linux Kernel 'StackRot'

Privi

Jul 06,

LINU

VULN

Red Hat: CVE-2022-29581: use-after-

fr

refer

Use-after-free in the IPv6 implementation of the DCCP

protoc

CVE-2023-32233: Privilege escalation in Linux

rability

Exploiting a Use-After-Free Vulnerability in the

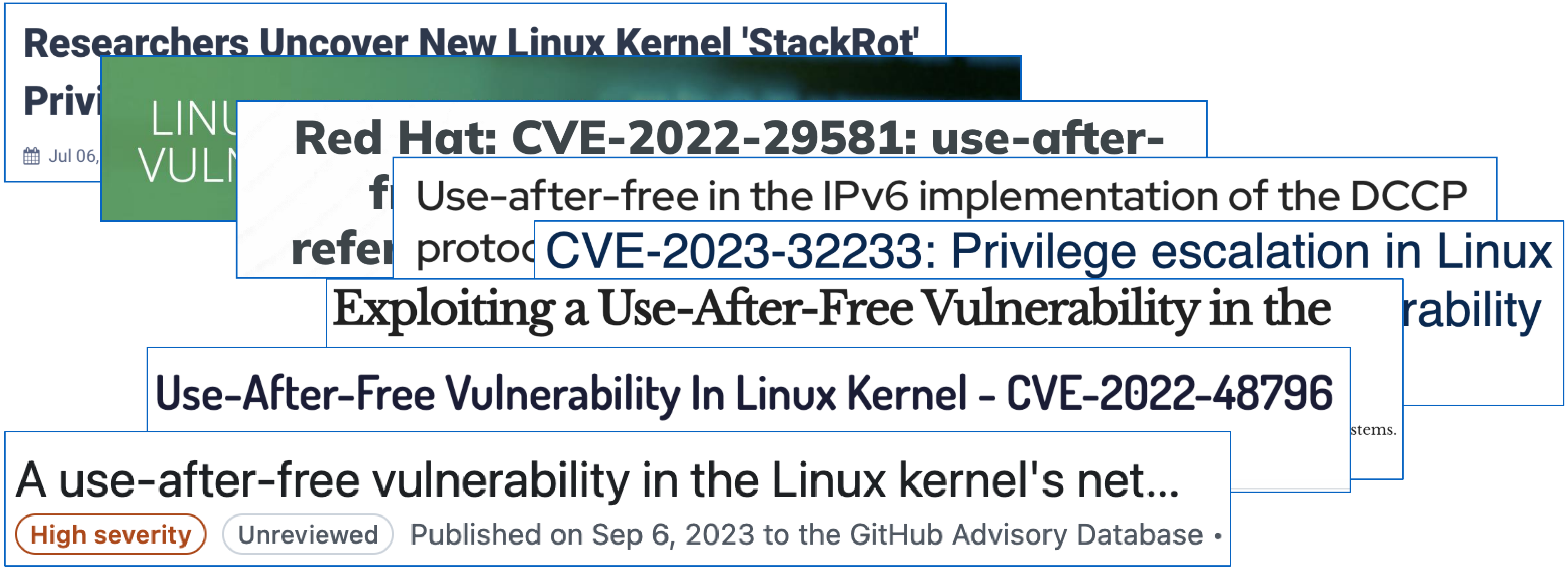
stems.

Use-After-Free Vulnerability In Linux Kernel - CVE-2022-48796

A use-after-free vulnerability in the Linux kernel's net...

High severity Unreviewed Published on Sep 6, 2023 to the GitHub Advisory Database •

Background—Kernel Use-After-Free Bugs


Researchers Uncover New Linux Kernel 'StackRot'

Privi

Jul 06,

LINU

VULN

Red Hat: CVE-2022-29581: use-after-

fr

refer

Use-after-free in the IPv6 implementation of the DCCP

protoc

CVE-2023-32233: Privilege escalation in Linux

rability

Exploiting a Use-After-Free Vulnerability in the

stems.

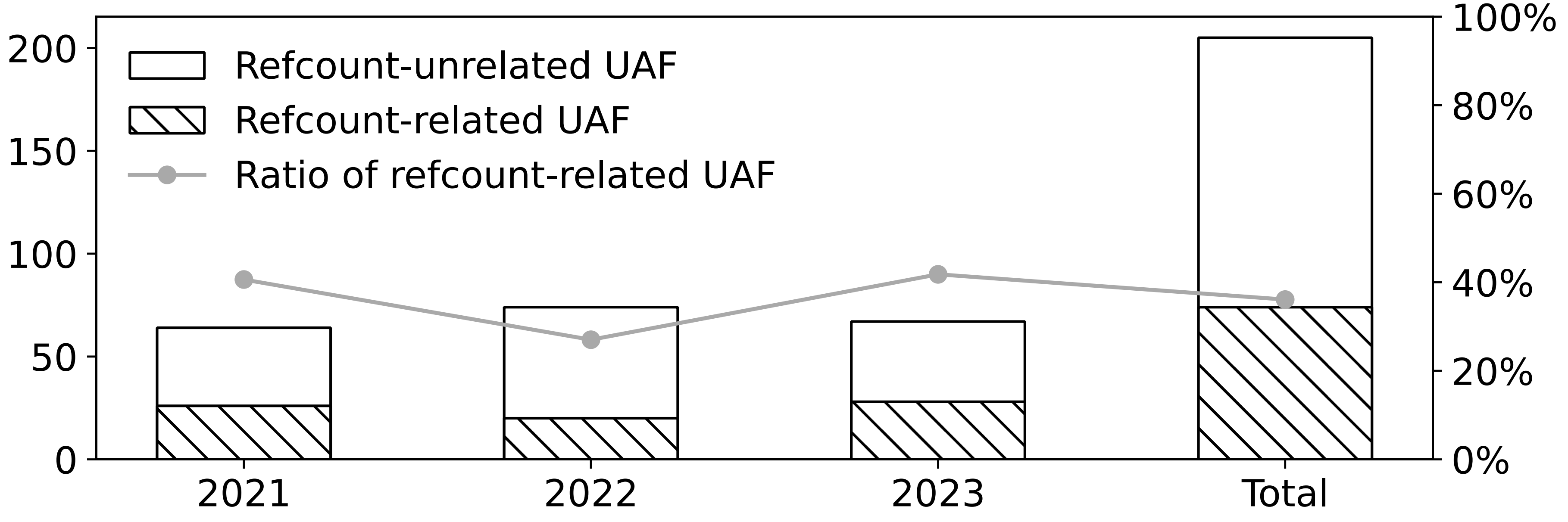
Use-After-Free Vulnerability In Linux Kernel - CVE-2022-48796

A use-after-free vulnerability in the Linux kernel's net...

High severity Unreviewed Published on Sep 6, 2023 to the GitHub Advisory Database •

Highly exploitable kernel use-after-free (UAF) bugs

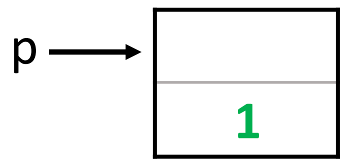
Background—Kernel Use-After-Free Bugs



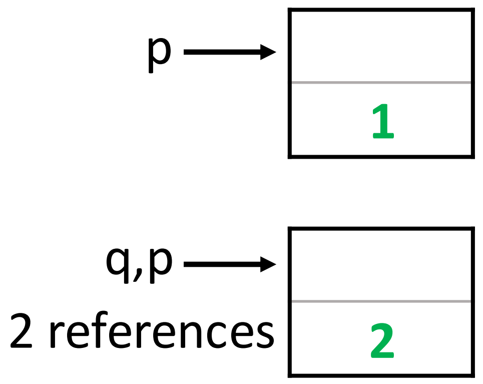
- 205 UAF bugs in past 3 years by syzbot
- 36% involving refcount issues

Background—Refcounts in Kernel

Background—Refcounts in Kernel

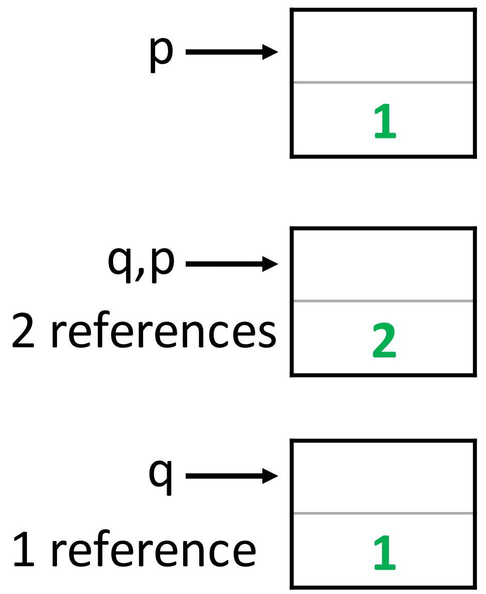


Background—Refcounts in Kernel



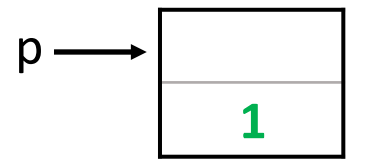
copy reference q = p; refcount += 1

Background—Refcounts in Kernel

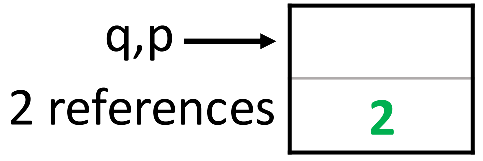


copy reference	q = p;	refcount += 1
destroy reference	p = null;	refcount -= 1

Background—Refcounts in Kernel

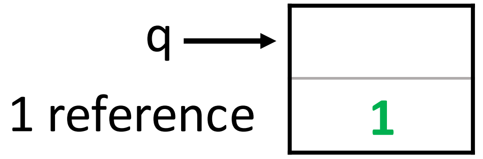


copy reference q = p;



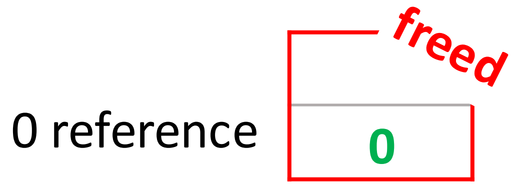
refcount += 1

destroy reference p = null;



refcount -= 1

destroy reference q = null;



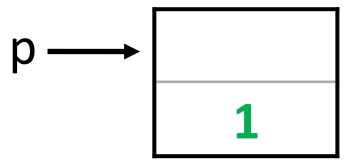
refcount -= 1

Background—Refcounts in Kernel

- refcount ≠ reference number

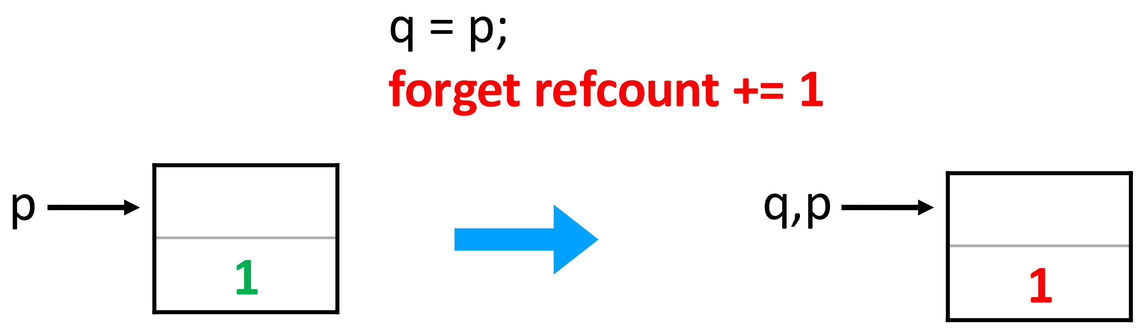
Background—Refcounts in Kernel

- refcount ≠ reference number



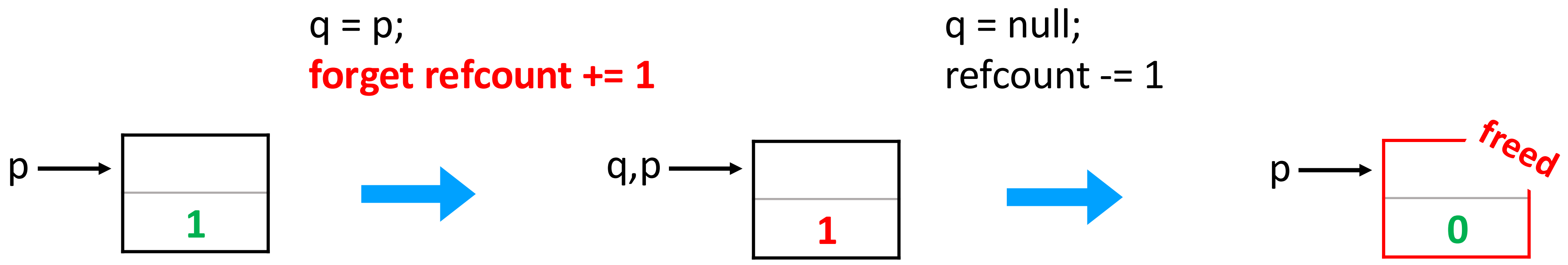
Background—Refcounts in Kernel

- refcount ≠ reference number



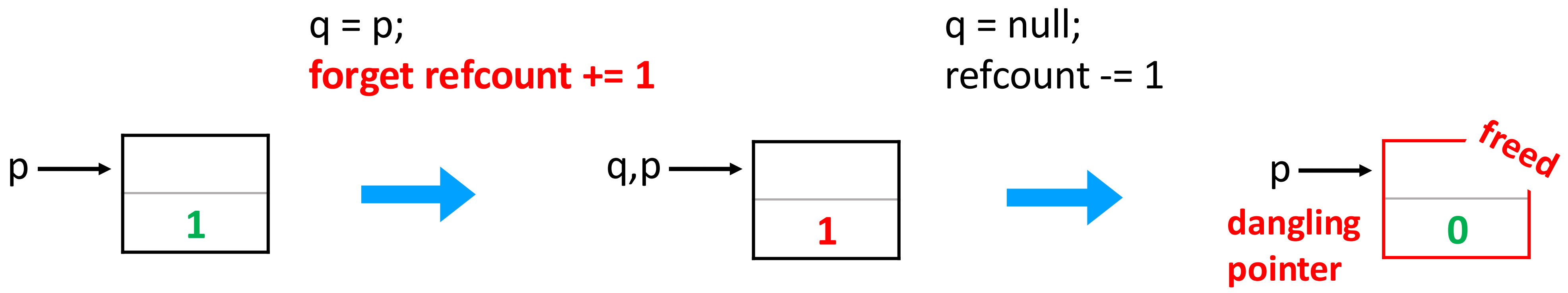
Background—Refcounts in Kernel

- refcount ≠ reference number



Background—Refcounts in Kernel

- refcount ≠ reference number => trigger use-after-free bugs



Previous Solutions for Bug Detection

- Coverage-guided fuzzing (e.g., Syzkaller [1] , Moonshine [2] and Healer [3])
- Heap-operation-guided fuzzing (e.g., Actor [4])
  - Unaware of refcount
    - Ignore progress of triggering such bugs
    - Low chance to find refcount-related UAF bugs
- Rule-based static analysis (e.g., Pungi [5] , RID [6] , CID [7] and LinKRID [8])
  - High false positives
    - LinKRID [8] produces around **40%** false positives

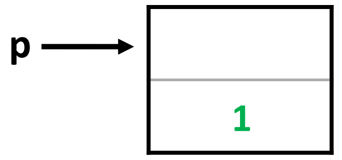
Our Contribution

- CountDown - Refcount-guided kernel fuzzer
  - Refcount-guided mutation
  - Refcount-aware input prioritization
- Results
  - 15 new kernel bugs, including 7 UAF bugs
- Open source
  - https://github.com/psu-security-universe/countdown

Motivating Example – CVE-2021-23134

```
int llcp_sock_bind(...) {
    llcp_sock->local = nfc_llcp_local_get(local);
    nfc_llcp_local_put(llcp_sock->local);

}
```

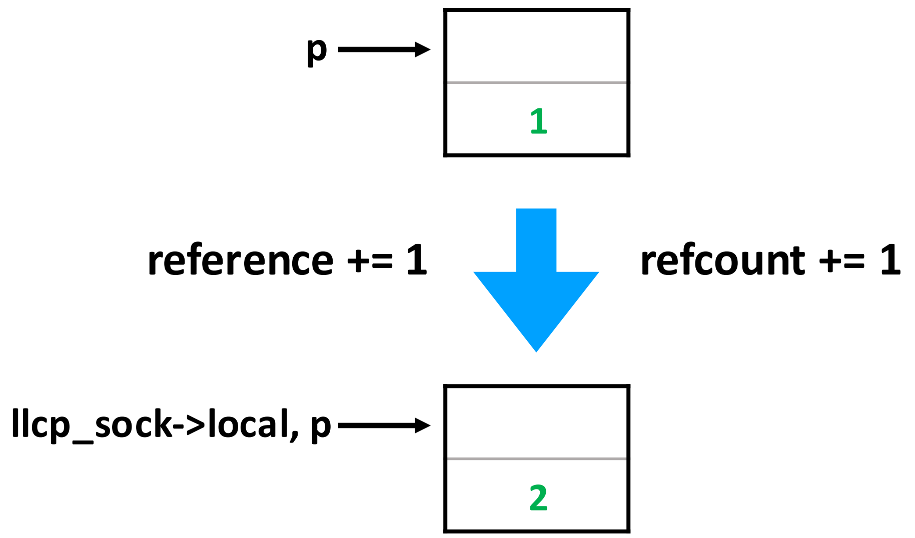
Motivating Example – CVE-2021-23134



```
int llcp_sock_bind(...) {
    llcp_sock->local = nfc_llcp_local_get(local);
    nfc_llcp_local_put(llcp_sock->local);

}
```

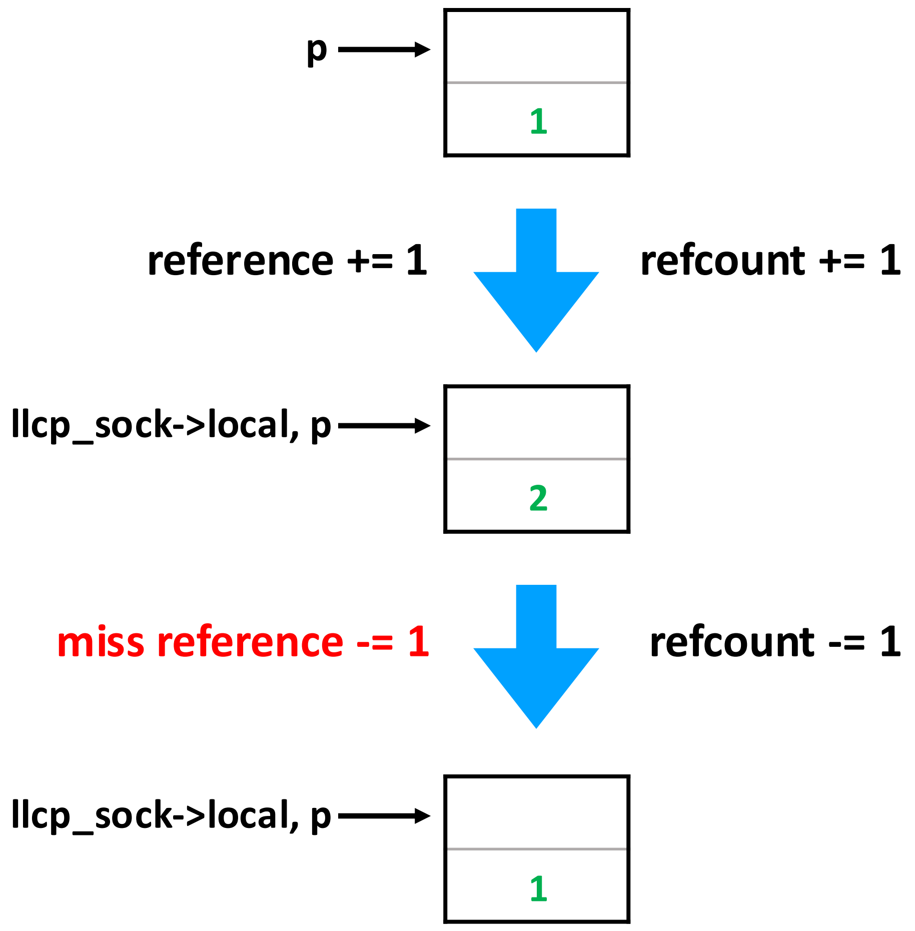
Motivating Example – CVE-2021-23134

```
int llcp_sock_bind(...) {
    llcp_sock->local = nfc_llcp_local_get(local);
    nfc_llcp_local_put(llcp_sock->local);


}
```



Motivating Example – CVE-2021-23134

```
int llcp_sock_bind(...) {
    llcp_sock->local = nfc_llcp_local_get(local);
    nfc_llcp_local_put(llcp_sock->local);
    // forget to destroy reference
}
```

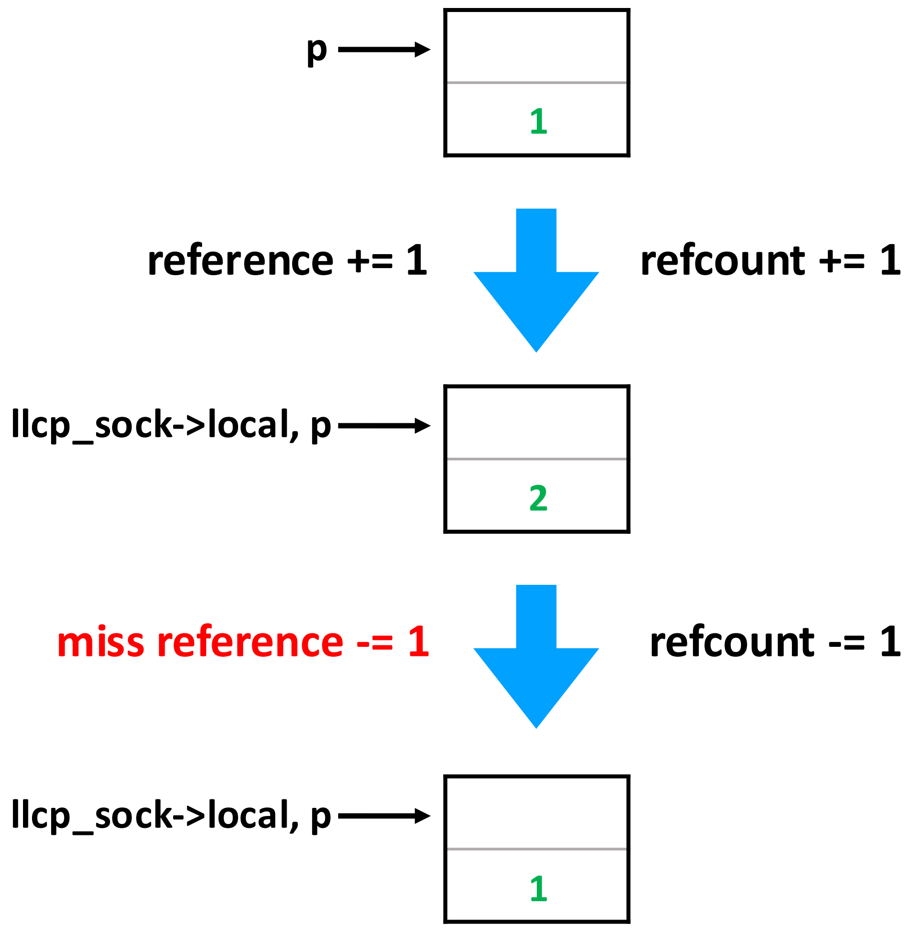


Motivating Example – CVE-2021-23134

```
int llcp_sock_bind(...) {
    llcp_sock->local = nfc_llcp_local_get(local);
    nfc_llcp_local_put(llcp_sock->local);
    // forget to destroy reference
}
```

bind: reference += 1, refcount += 0

Root cause: wrong refcount usage



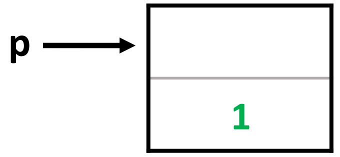
Motivating Example – CVE-2021-23134

Proof-of-Concept (PoC) to trigger CVE

```
void PoC(void) {
  int sock1 = socket(...);
  int sock2 = socket(...);
  bind(sock1, &addr, ...);
  bind(sock2, &addr, ...);
  close(sock1);
  close(sock2);
}
```

Motivating Example – CVE-2021-23134

Proof-of-Concept (PoC) to trigger CVE

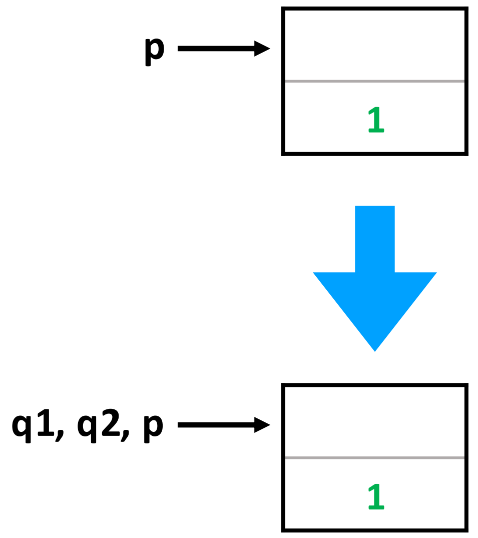


```
void PoC(void) {
  int sock1 = socket(...);
  int sock2 = socket(...);






}
```

Motivating Example – CVE-2021-23134

Proof-of-Concept (PoC) to trigger CVE

1. Introduce extra references



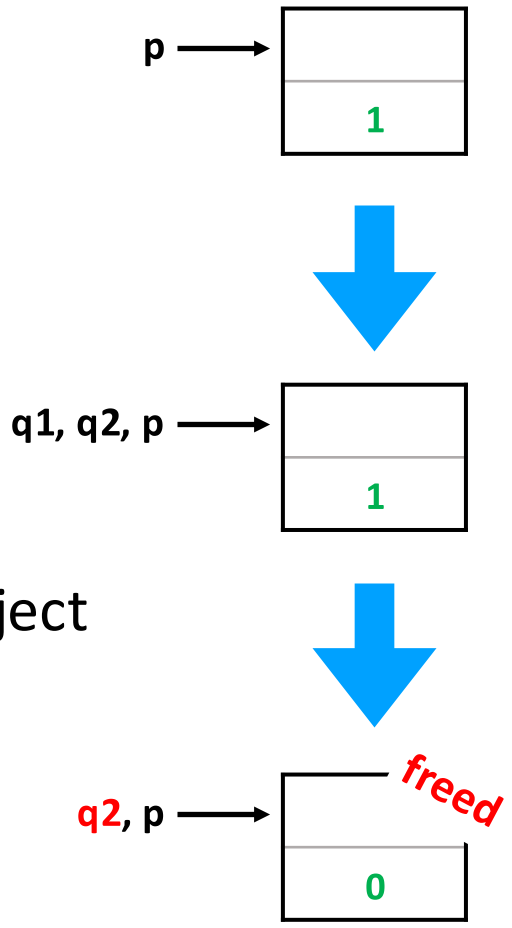
```
void PoC(void) {
  int sock1 = socket(...);
  int sock2 = socket(...);
  bind(sock1, &addr, ...);
  bind(sock2, &addr, ...);



}
```

Motivating Example – CVE-2021-23134

Proof-of-Concept (PoC) to trigger CVE

1. Introduce extra references

2. Decrease refcount to free object



```
void PoC(void) {
   int sock1 = socket(...);
   int sock2 = socket(...);
   bind(sock1, &addr, ...);
   bind(sock2, &addr, ...);
   close(sock1);

}
```

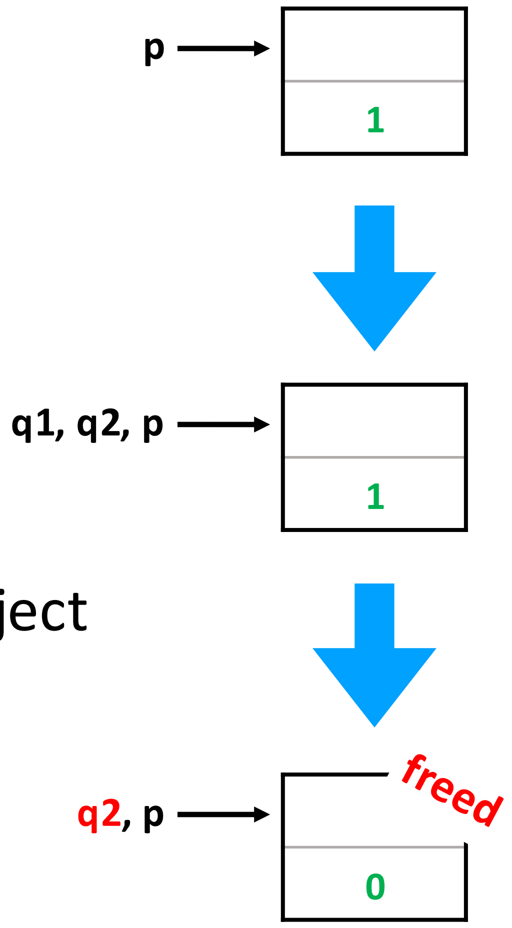
Motivating Example – CVE-2021-23134

Proof-of-Concept (PoC) to trigger CVE

1. Introduce extra references

2. Decrease refcount to free object

3. Access freed object



```
void PoC(void) {
   int sock1 = socket(...);
   int sock2 = socket(...);
   bind(sock1, &addr, ...);
   bind(sock2, &addr, ...);
   close(sock1);
   close(sock2);
}
```

Challenge of Bug Detection

Challenge of Bug Detection

Simple sequence:

socket-bind-close

Necessary sequence:

socket-bind-bind/close-close

PoC sequence:

socket-socket-bind-bind-close-close

Challenge of Bug Detection

Simple sequence:

socket-bind-close

Necessary sequence:

Unpreferred

socket-bind-bind/close-close

Same Coverage

PoC sequence:

socket-socket-bind-bind-close-close

- Code coverage guidance is not enough
  - No new coverage, no interest
  - Ignore refcount operations
    - Refcount access
    - Special refcount states

Challenge of Bug Detection

Simple sequence:

socket-bind-close

Necessary sequence:

Unpreferred

socket-bind-bind/close-close

Same Coverage

PoC sequence:

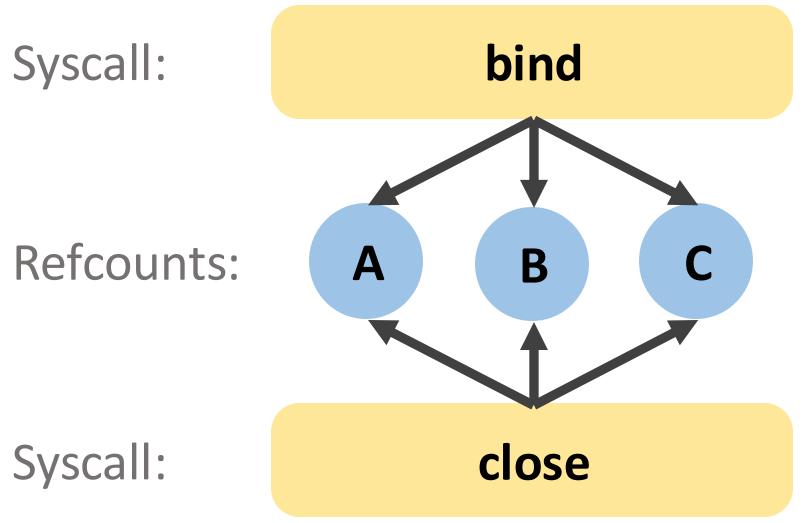
socket-socket-bind-bind-close-close

- Code coverage guidance is not enough
  - No new coverage, no interest
  - Ignore refcount operations
    - Refcount access
    - Special refcount states
- Static analysis
  - High false positives

Core Idea – Refcount-guided Fuzzing

Core Idea – Refcount-guided Fuzzing

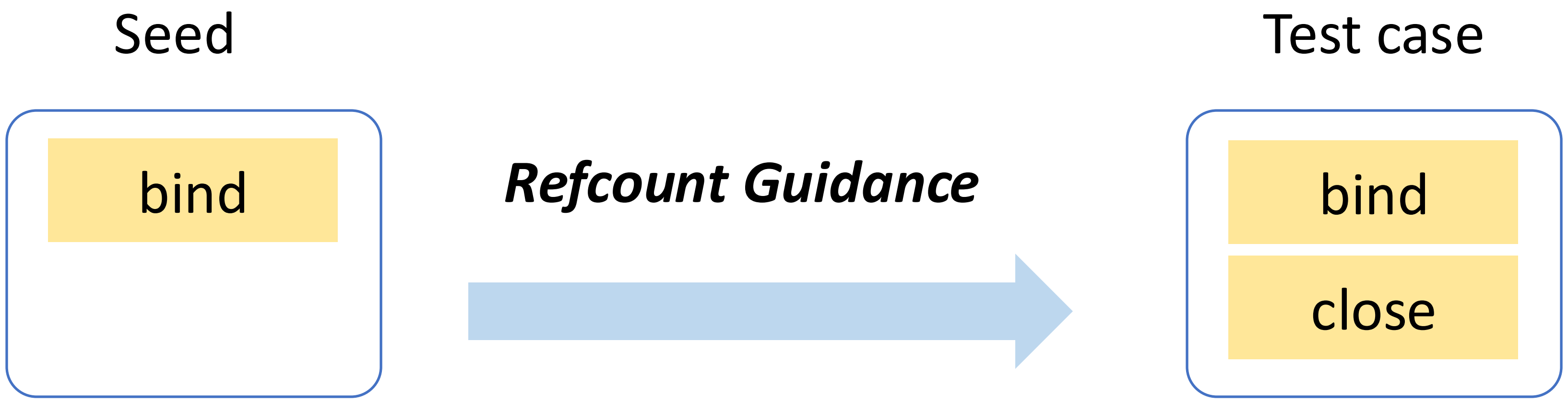
- **Refcount-based syscall relations**



Enhance relation **bind-close**

Core Idea – Refcount-guided Fuzzing

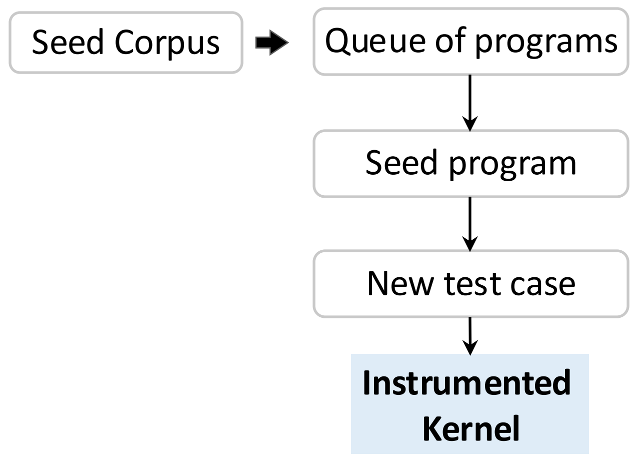
- Refcount-based syscall relations
- **Refcount-guided mutation**



Core Idea – Refcount-guided Fuzzing

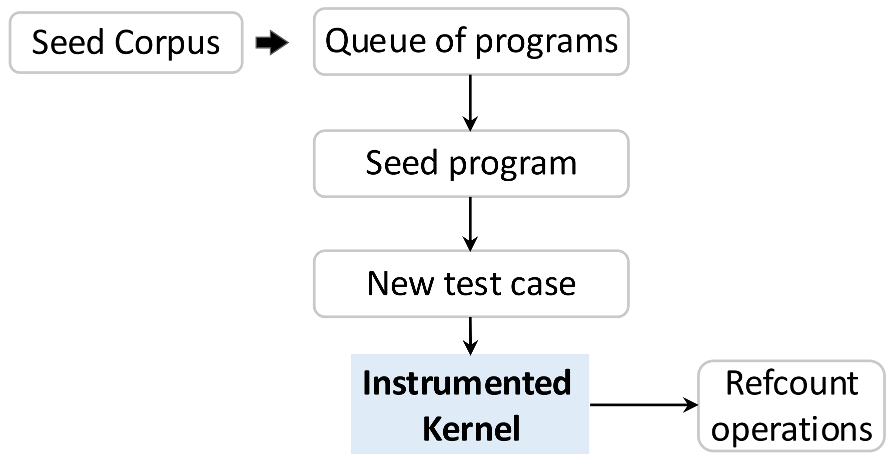
- Refcount-based syscall relations
- Refcount-guided mutation
- **Refcount-aware input prioritization**
    - Preserve unique refcount operation
    - *(syscall, refcount)*

Design

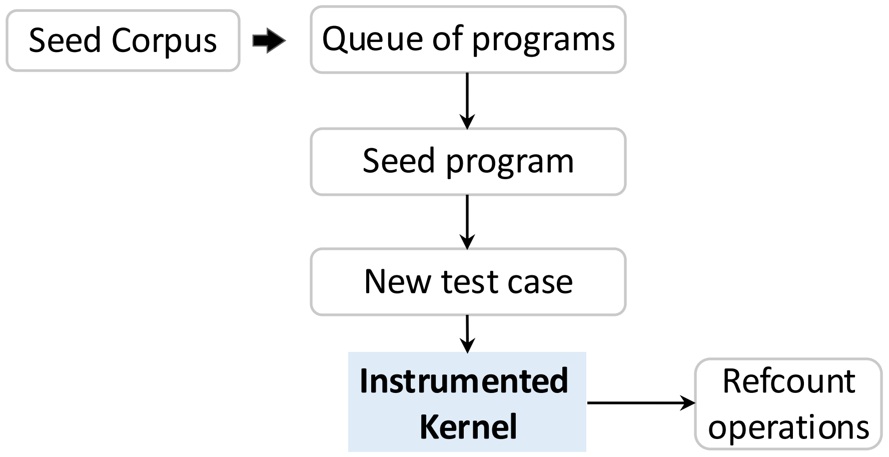

Seed Corpus
Queue of programs
Seed program
New test case
Instrumented Kernel

Design

- Refcount Operation Collection



Design



- Refcount Operation Collection

```
void Simple(void) {
   int sock = socket(...);
   bind(sock, &addr, ...);
   close(sock);
}
```

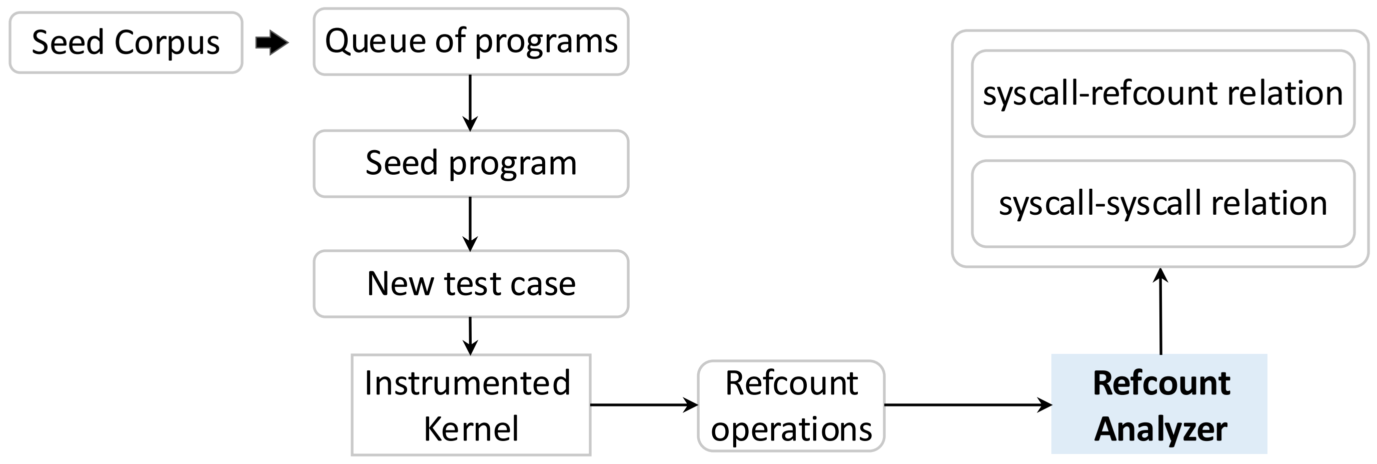
→ No refcount operation

→ bind operates refcount A; refcount += 0;

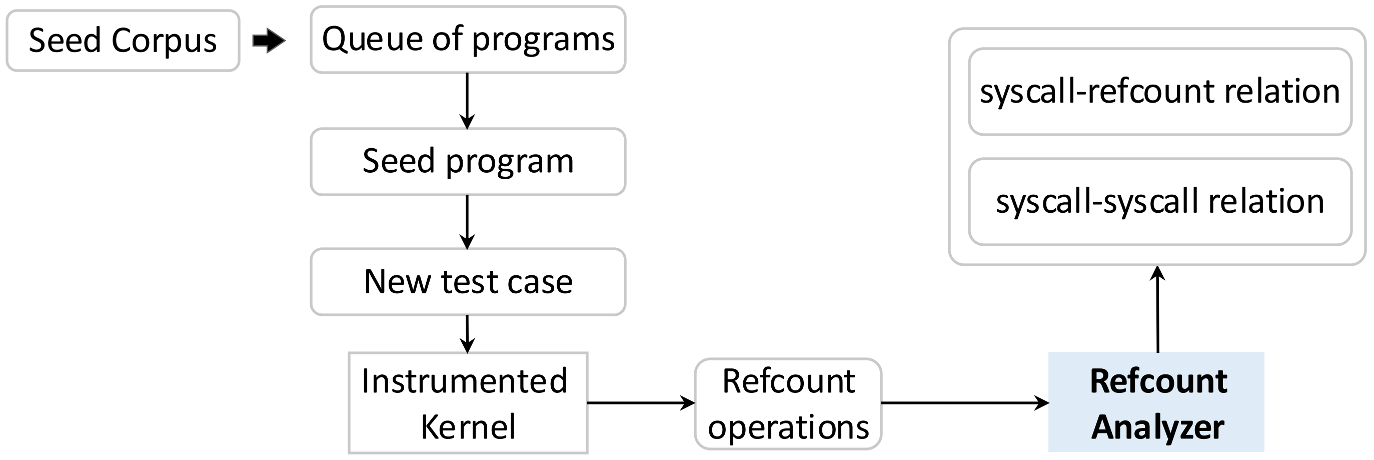
→ close operates refcount A; refcount -= 1;

Design

- Reshape Syscall Relation
    - Refcount relation

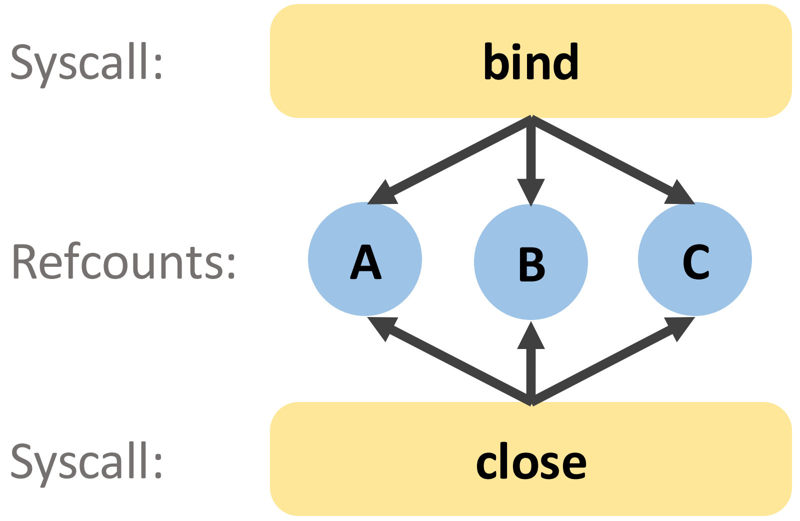


Design



- Reshape Syscall Relation
  - Refcount relation

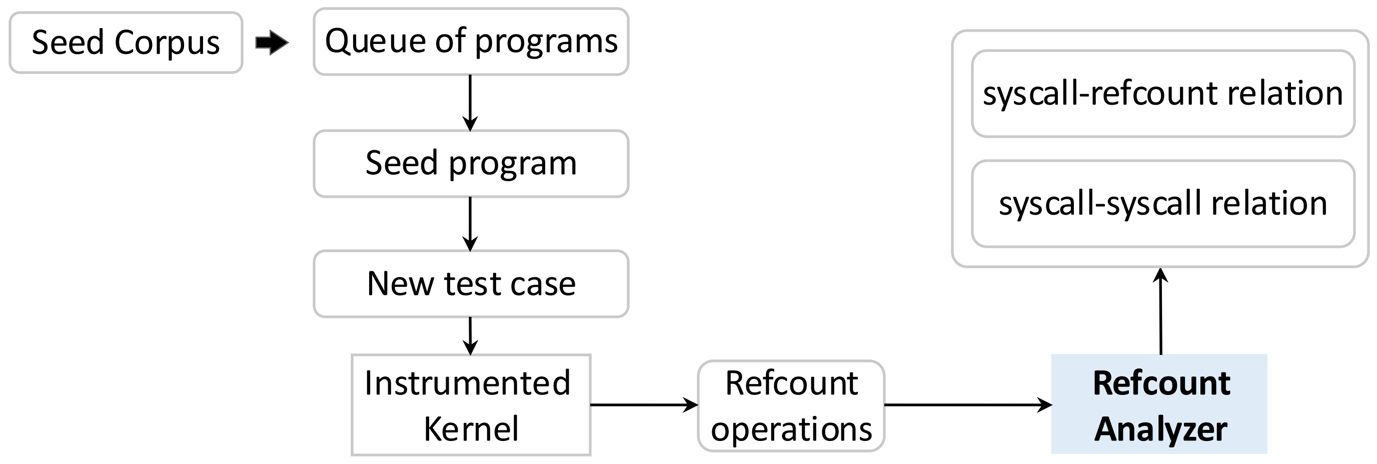
The number of unique refcounts operated by a syscall pair



Syscall Pair	Objects	Relation
bind-close	A, B, C	3

Design

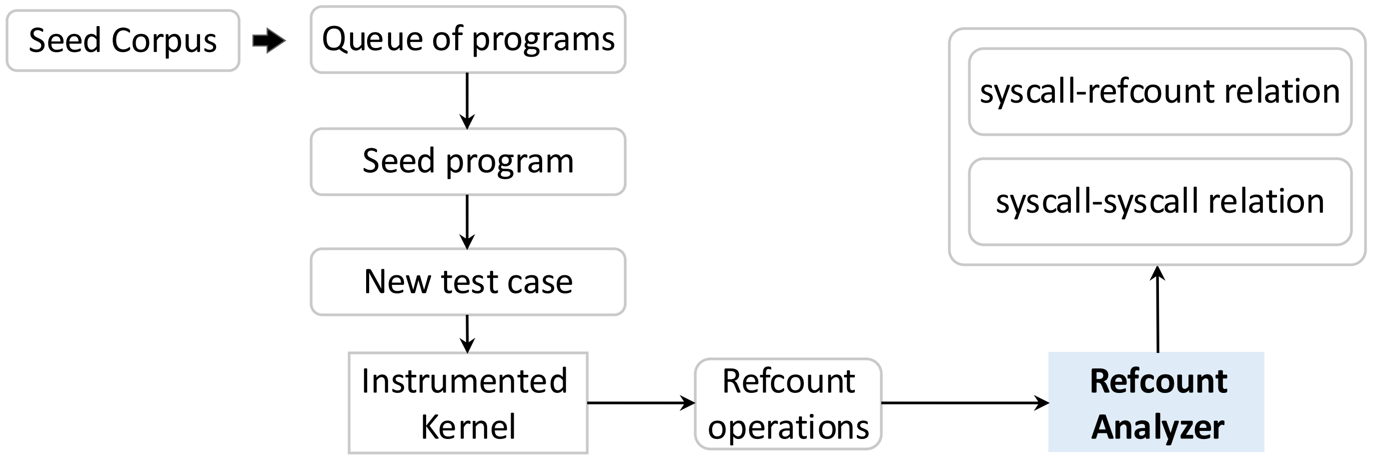
- Reshape Syscall Relation



$$OverallRelation = \log_2 SyzRelation + k * \log_2 RefcntRelation$$

Design

- Reshape Syscall Relation

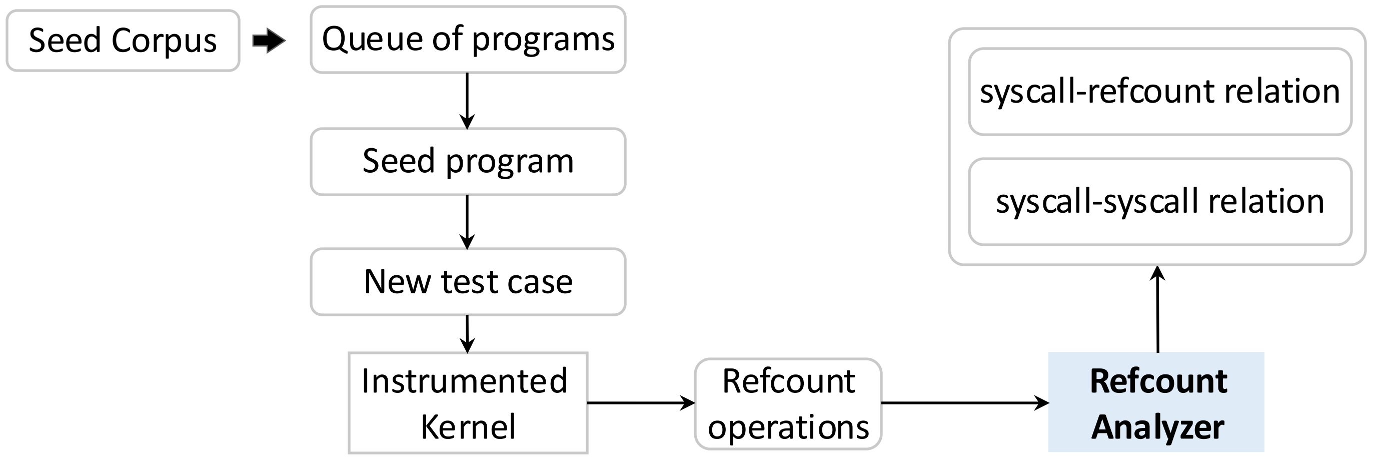


Original Syzkaller Relation

$$\downarrow$$

$$OverallRelation = \log_2 SyzRelation + k * \log_2 RefcntRelation$$

Design



- Reshape Syscall Relation

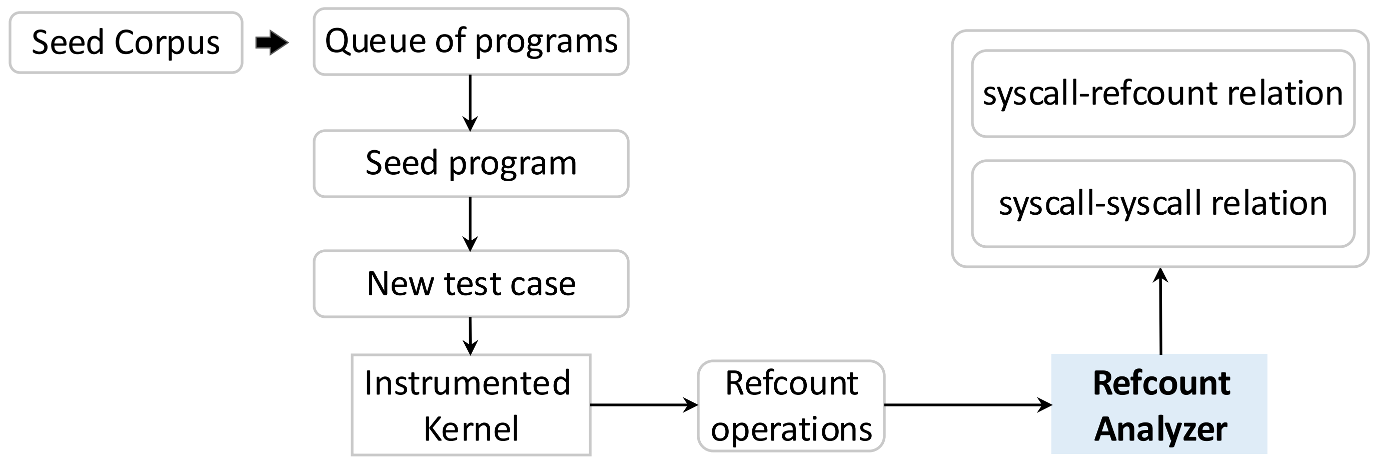
<u>Our Refcount Relation</u>

$$\downarrow$$

$$OverallRelation = \log_2 SyzRelation + k * \log_2 RefcntRelation$$

Design

- Reshape Syscall Relation



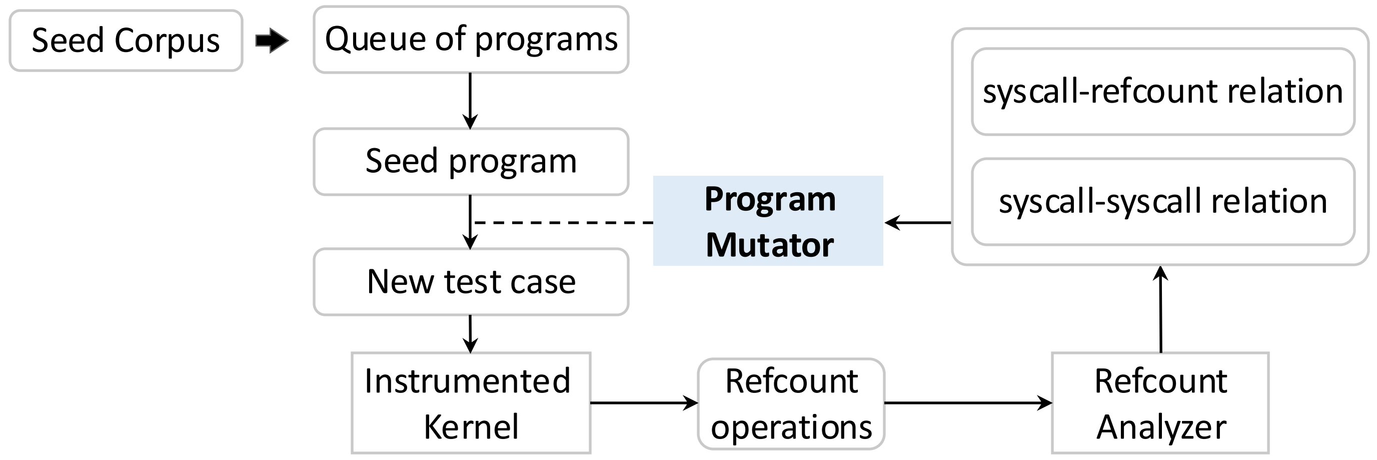
<u>New Relation for Mutation</u>

$$\downarrow$$

$$OverallRelation = \log_2 SyzRelation + k * \log_2 RefcntRelation$$

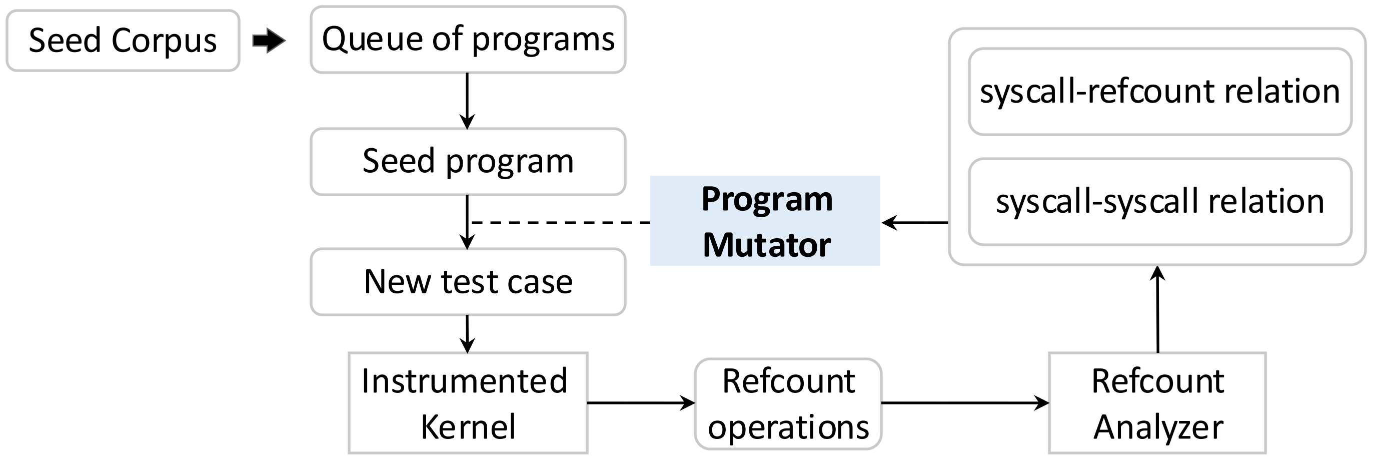
Design

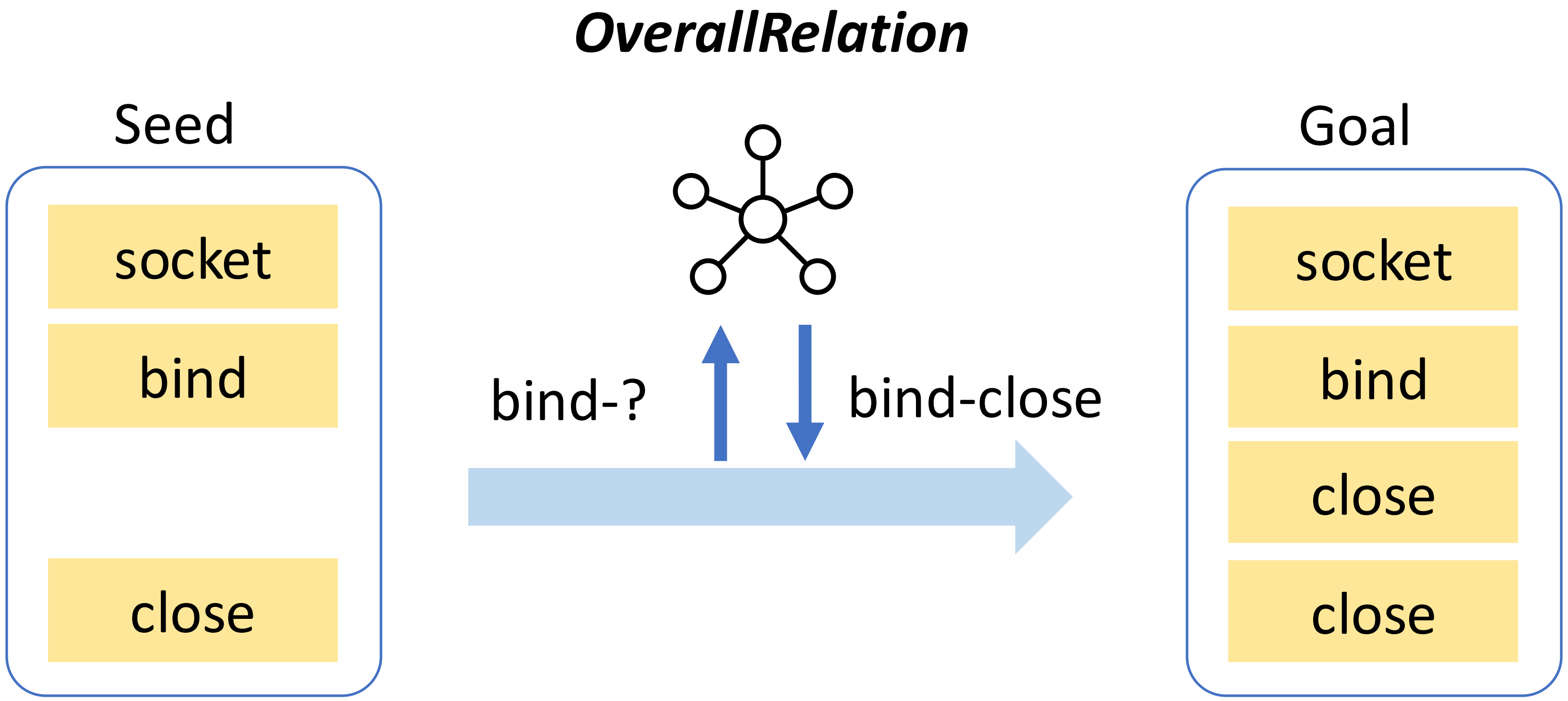
- Relation-based Mutation



Design

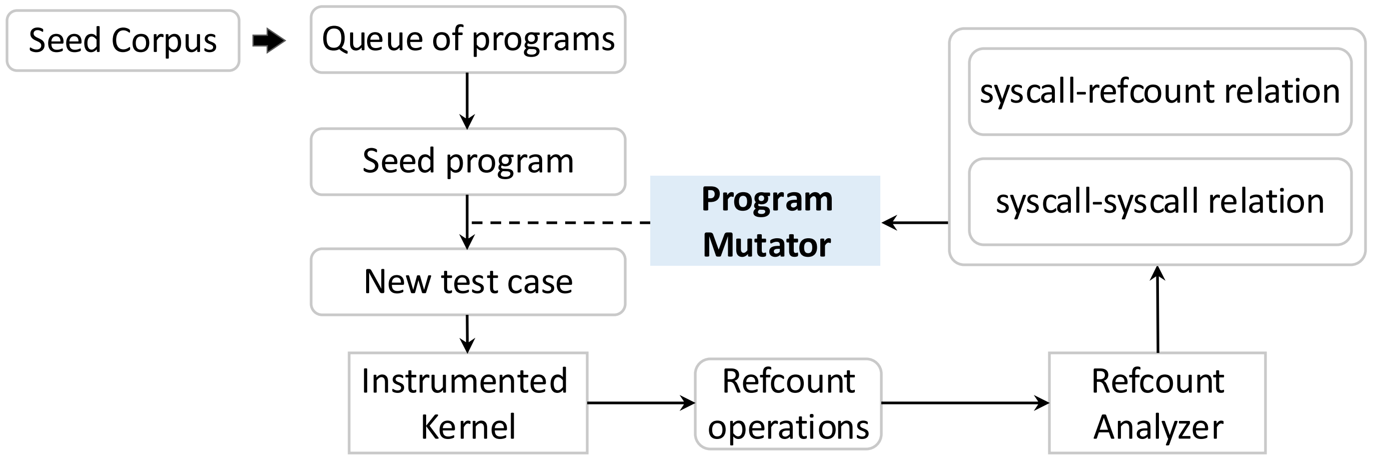
- Relation-based Mutation

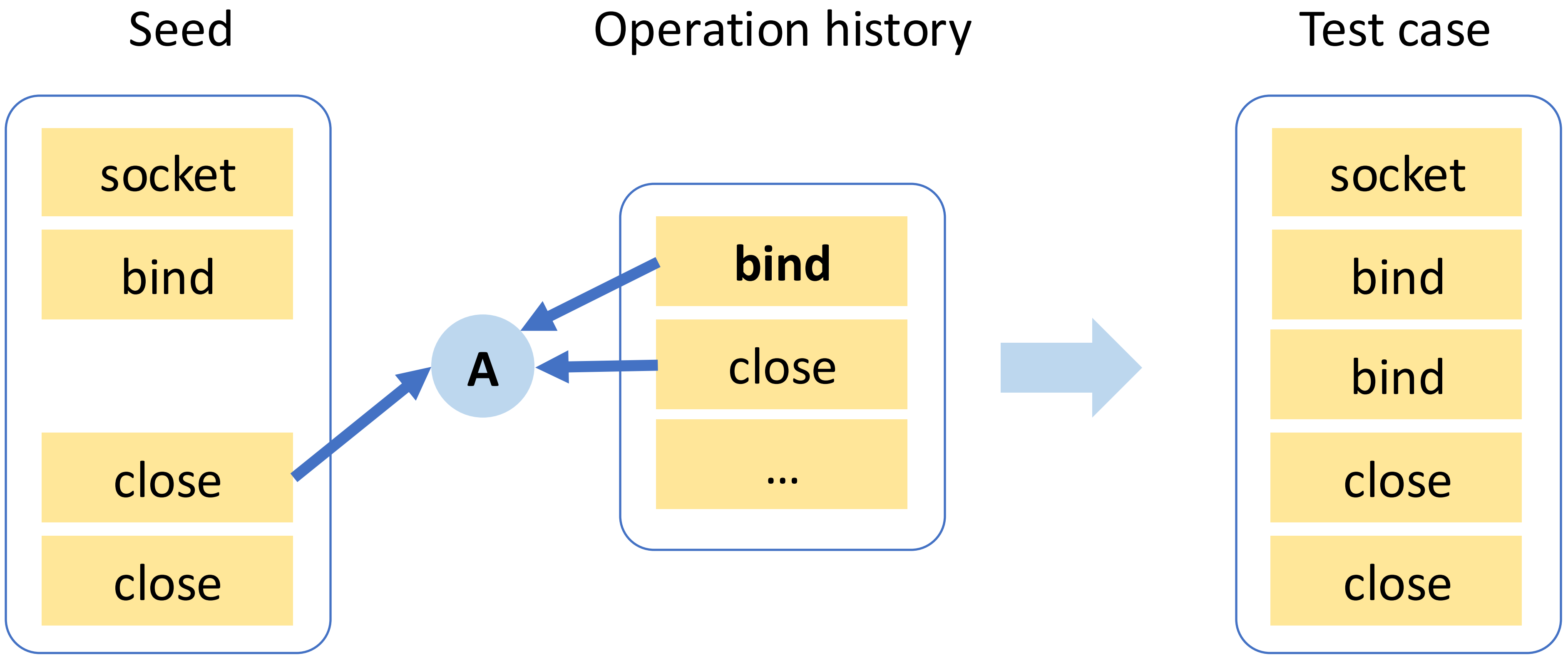




Design

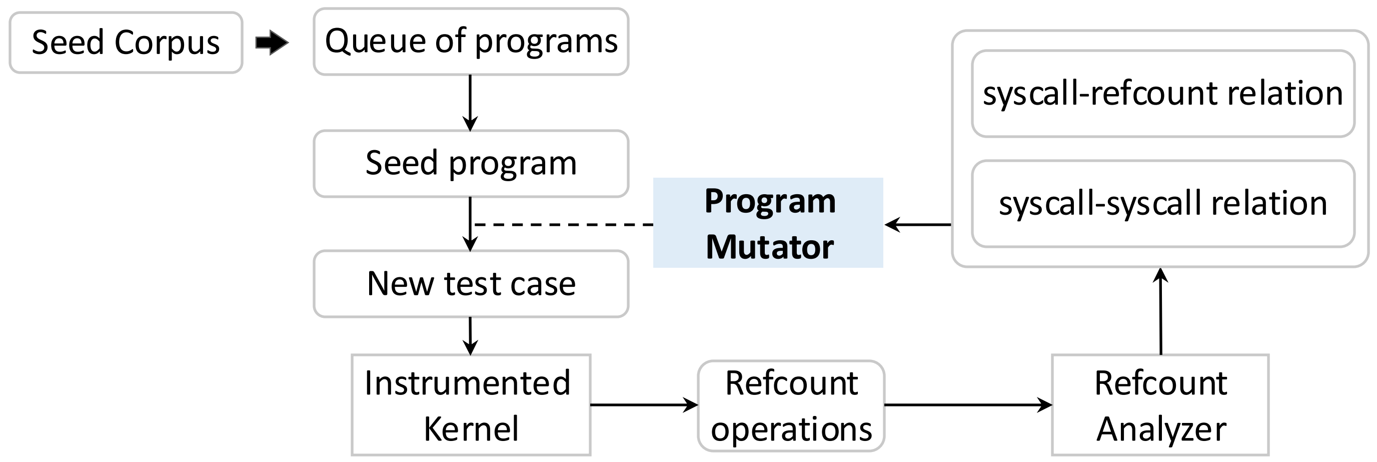
- Object-guided Mutation
  - Refcount mutator





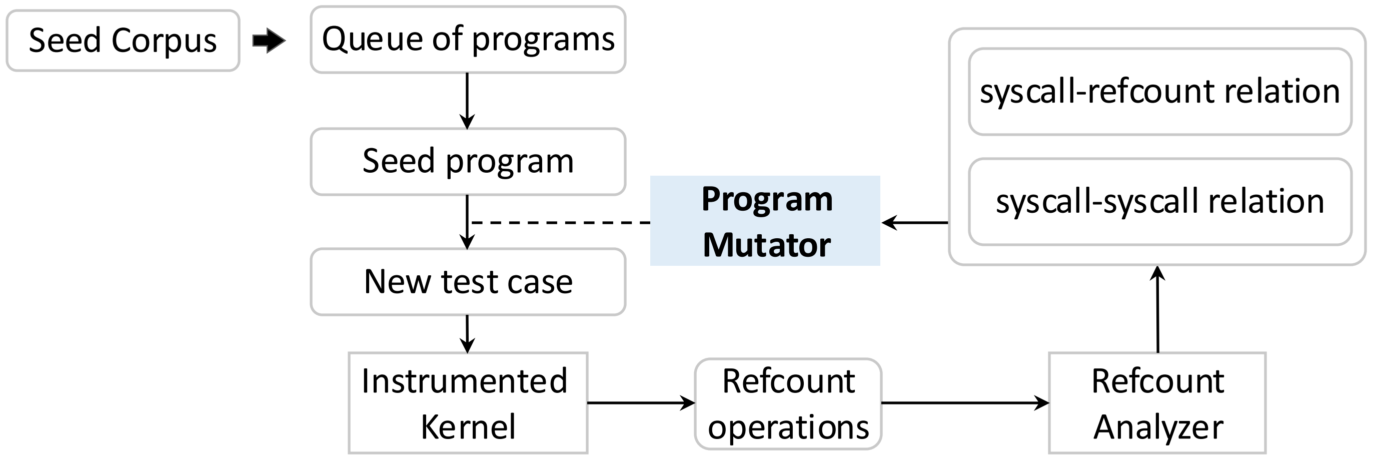
Design

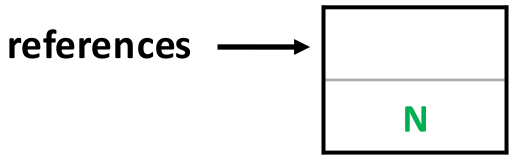
- refcount issue => use-after-free



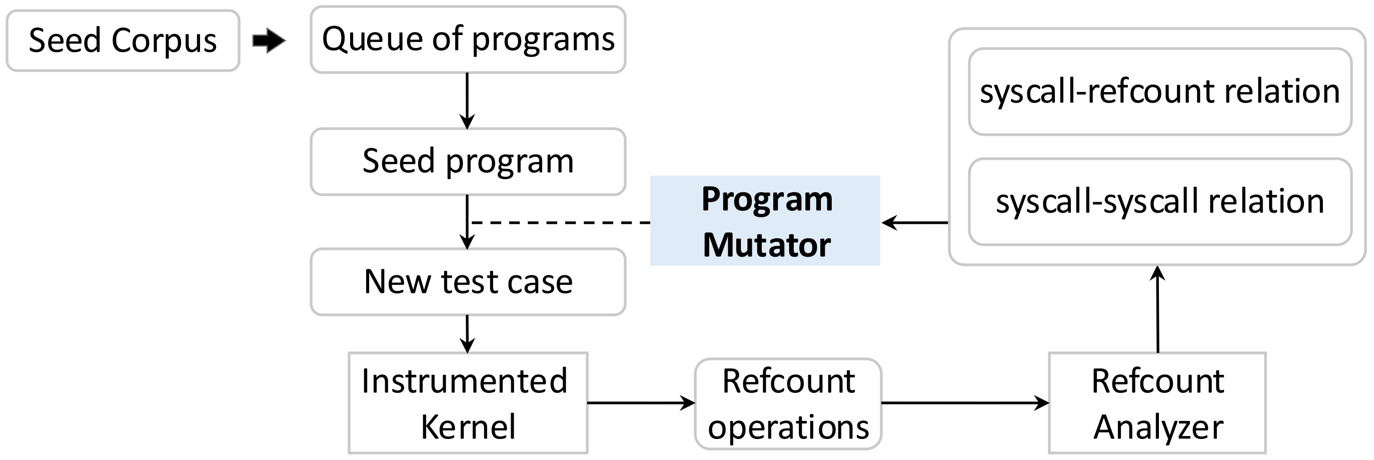
Design

- refcount issue => use-after-free



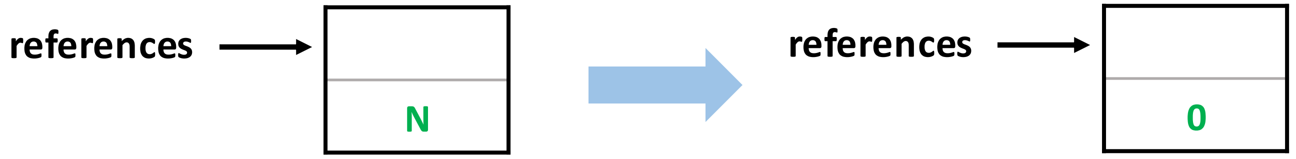


Design

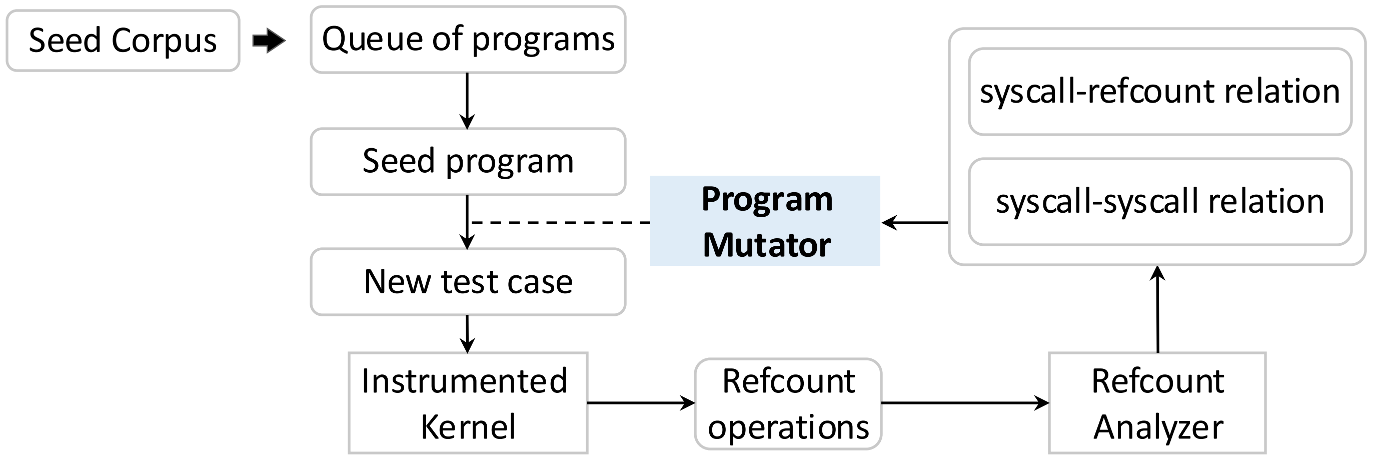


- refcount issue => use-after-free

Free object (Refcount -= N)

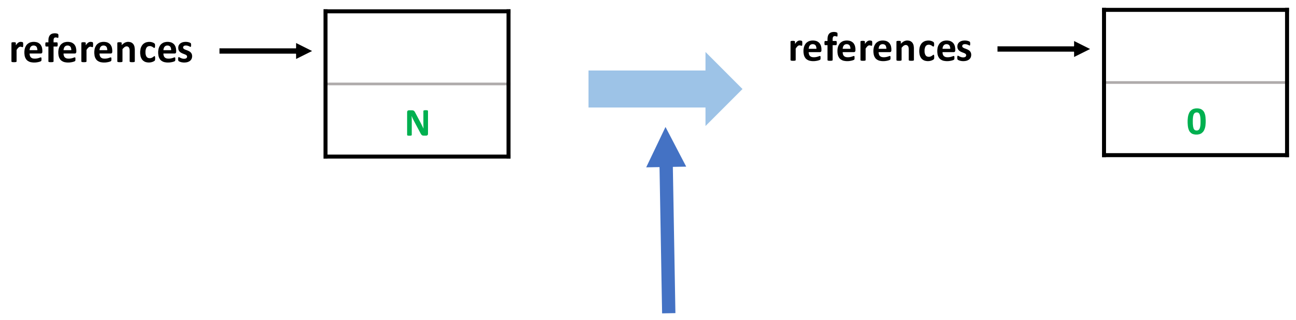


Design



- refcount issue => use-after-free

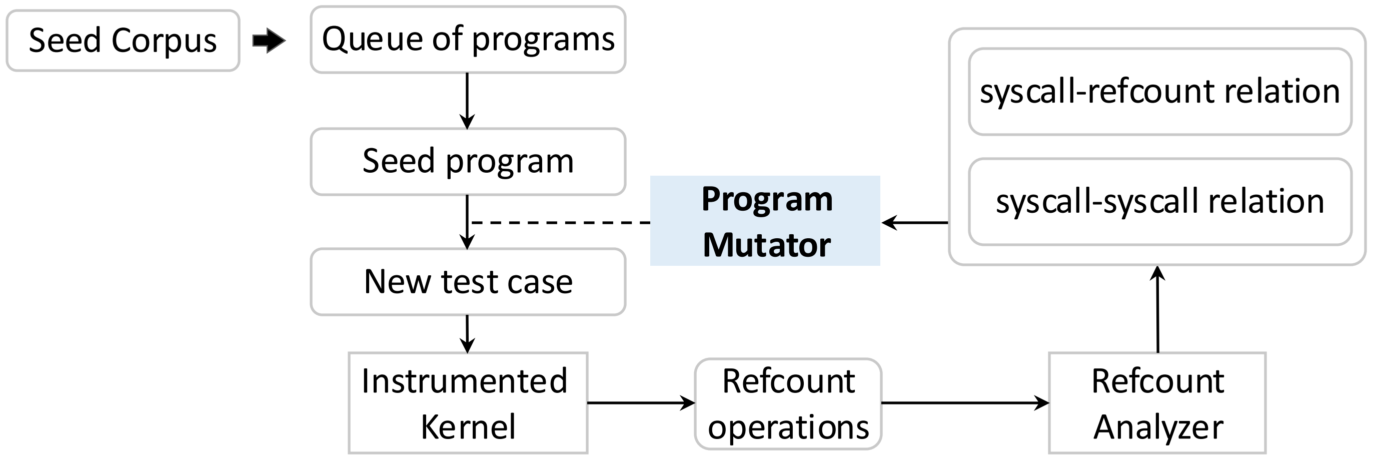
Free object (Refcount -= N)

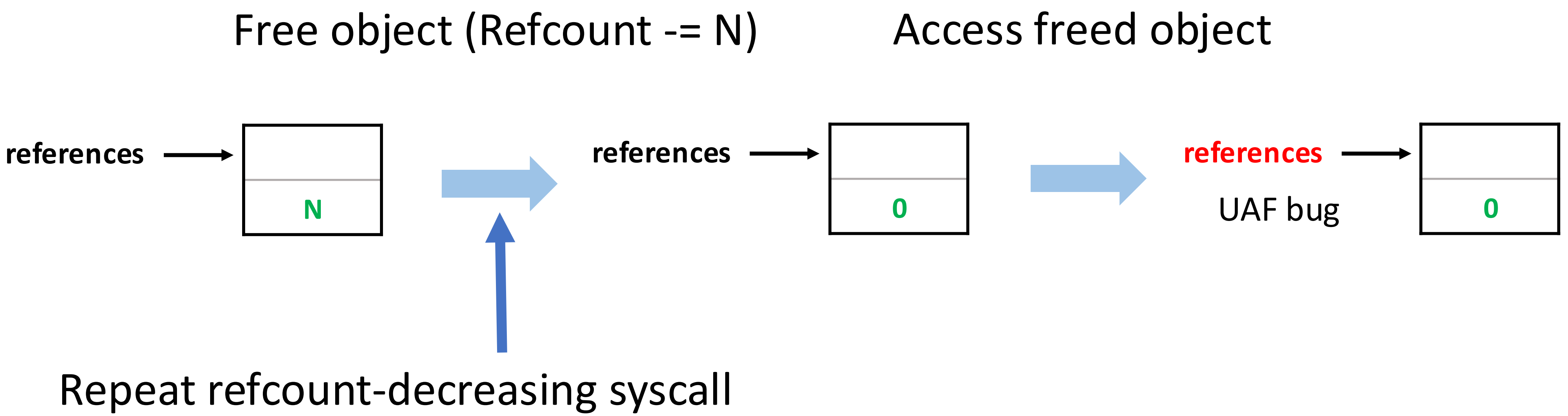


Repeat refcount-decreasing syscall

Design

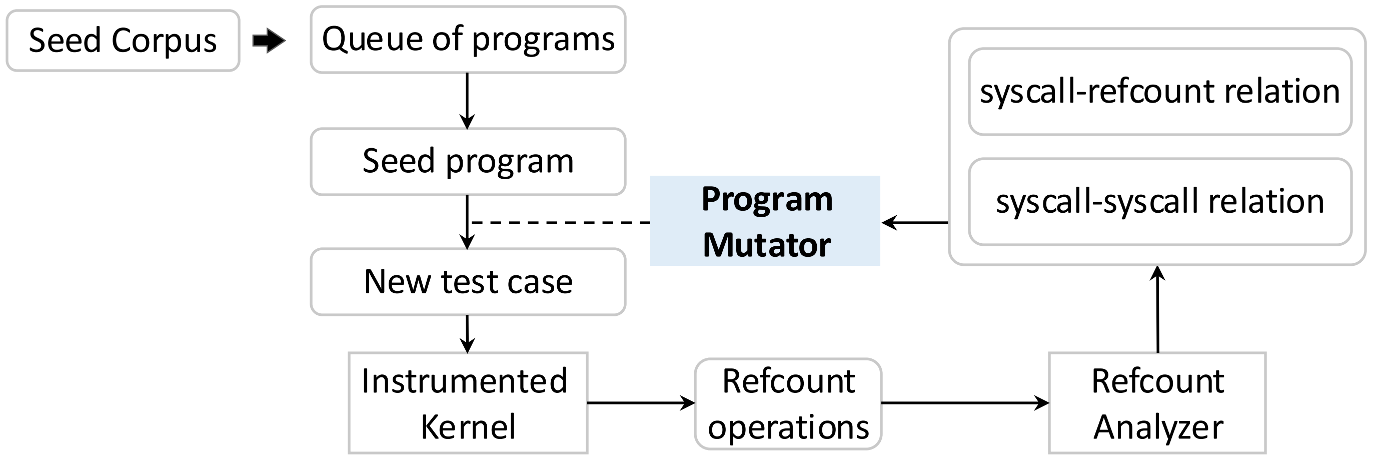
- refcount issue => use-after-free

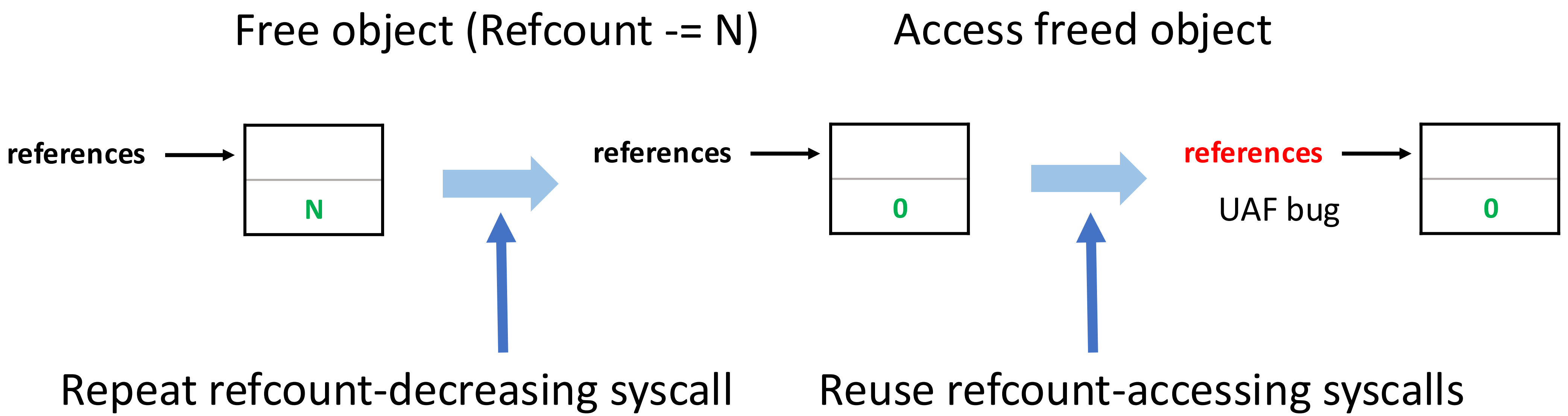



Free object (Refcount -= N)
Access freed object
references
N
references
0
references
UAF bug
0
Repeat refcount-decreasing syscall

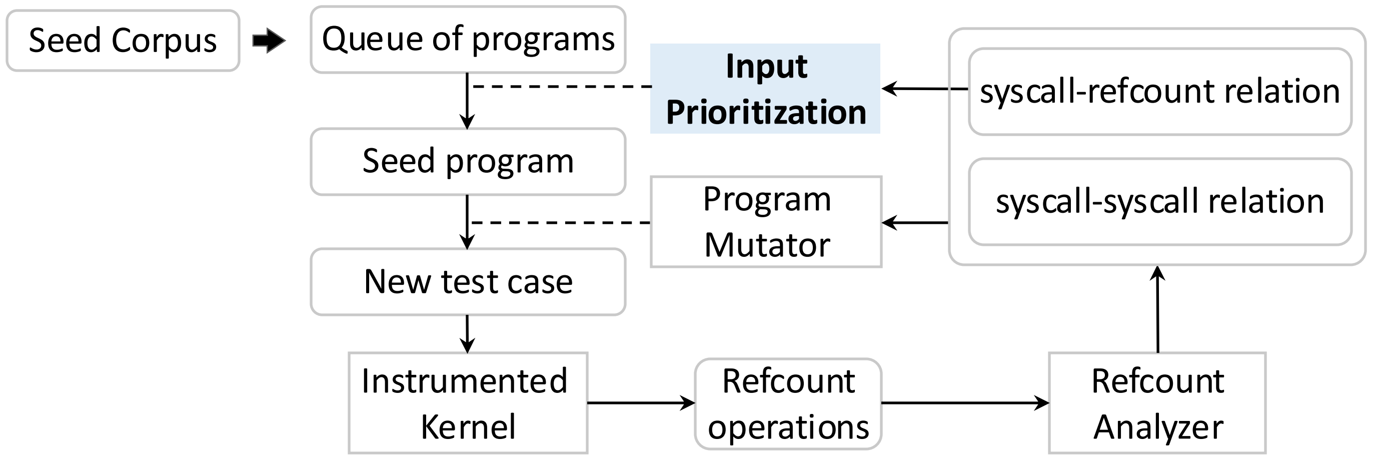
Design

- refcount issue => use-after-free



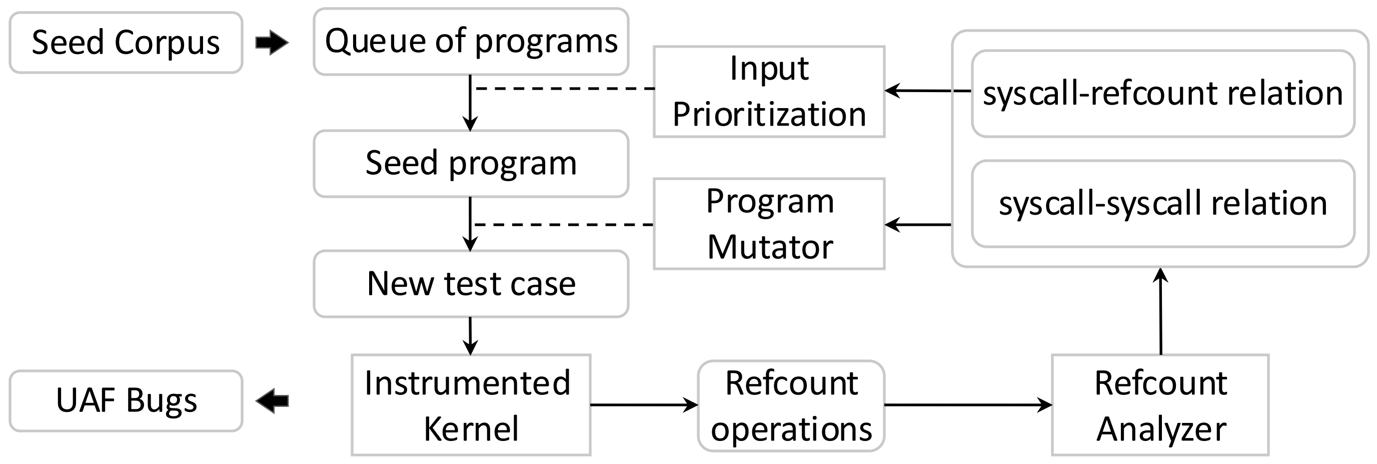

Free object (Refcount -= N)
Access freed object
references
N
references
0
references
UAF bug
0
Repeat refcount-decreasing syscall
Reuse refcount-accessing syscalls

Design



- Input Prioritization
  1. New code coverage
  2. New refcount operation
     - (syscall, object)

Design

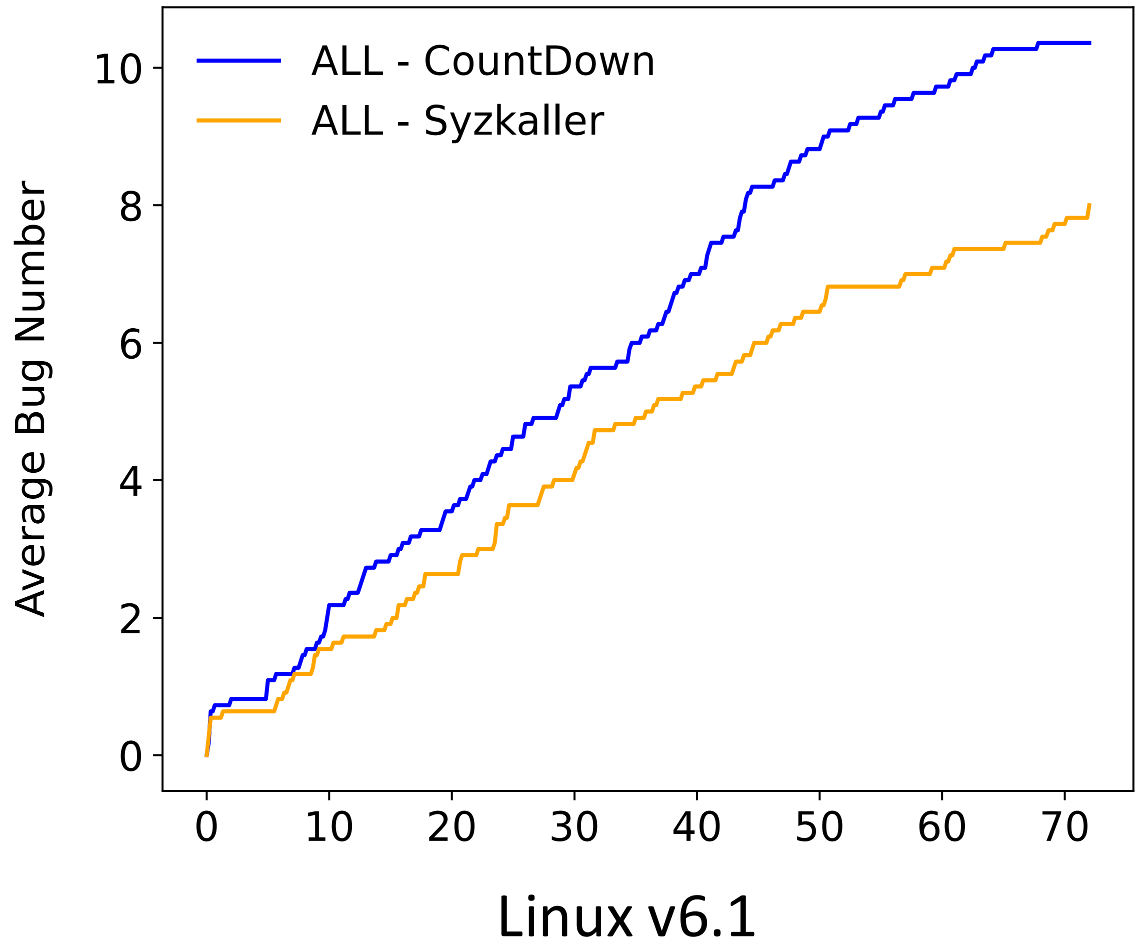

Seed Corpus
Queue of programs
Input Prioritization
syscall-refcount relation
syscall-syscall relation
Seed program
Program Mutator
New test case
UAF Bugs
Instrumented Kernel
Refcount operations
Refcount Analyzer

Evaluation – Setup

- Comparison with Syzkaller
  - Kernel versions: v5.15, v6.1, v6.6
  - Corpus: Syzbot corpus

- Comparison with other advanced tools (Moonshine, Actor)
  - Kernel version: v6.2-rc5 (supported by Actor)
  - Corpus: refer to paper

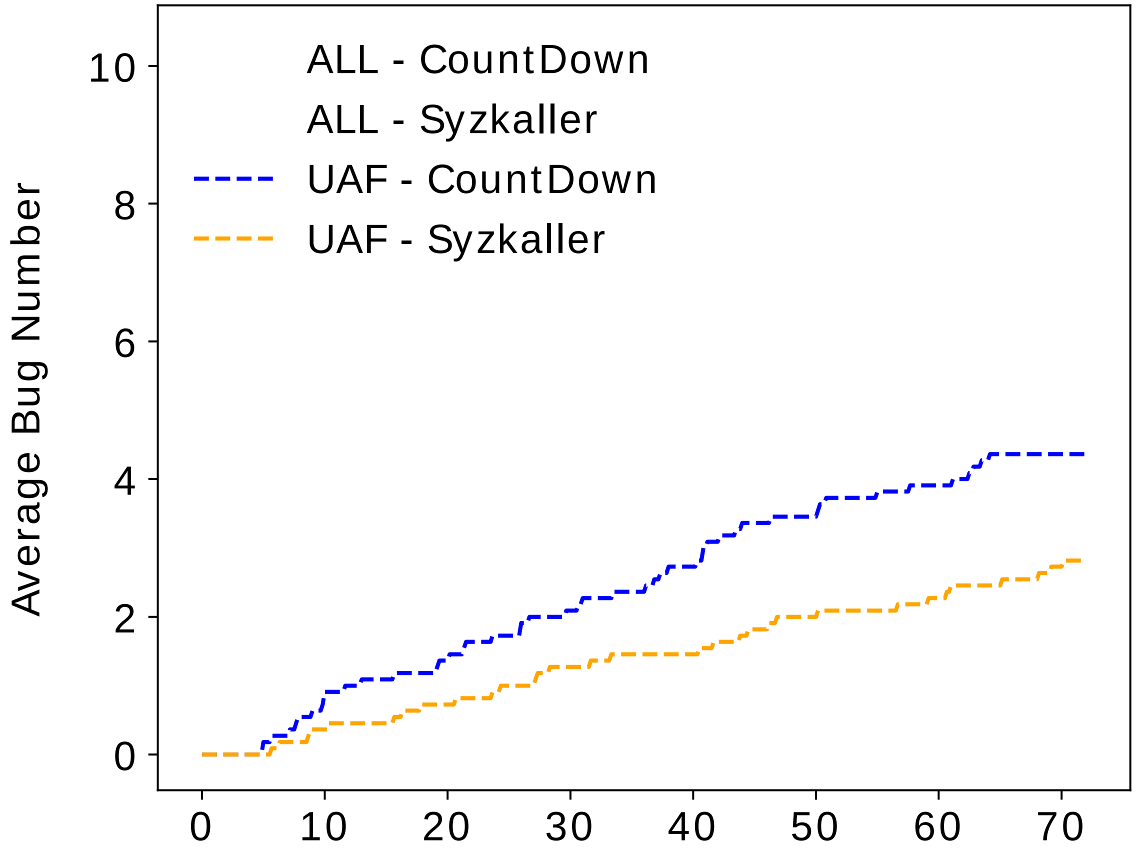
Evaluation – Bug Finding



CountDown v.s. Syzkaller (v6.1)

- **30.0%** more KASAN reports

Evaluation – Bug Finding

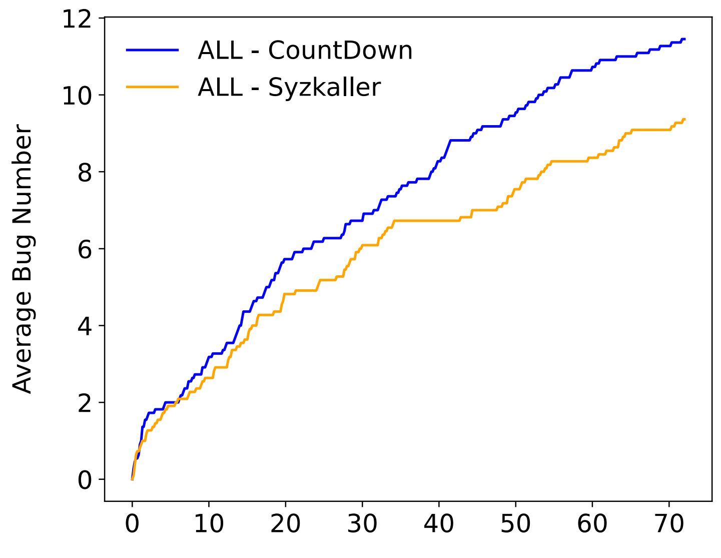


Linux v6.1

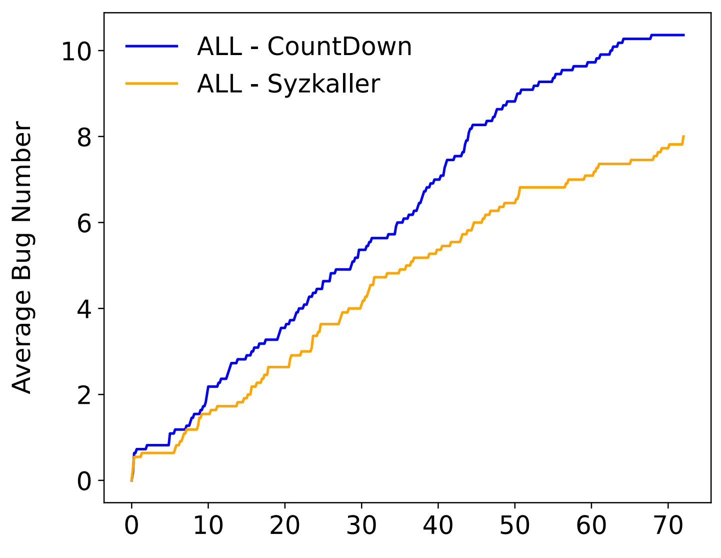
CountDown v.s. Syzkaller (v6.1)

- **30.0%** more KASAN reports
- **57.1%** more UAF bugs

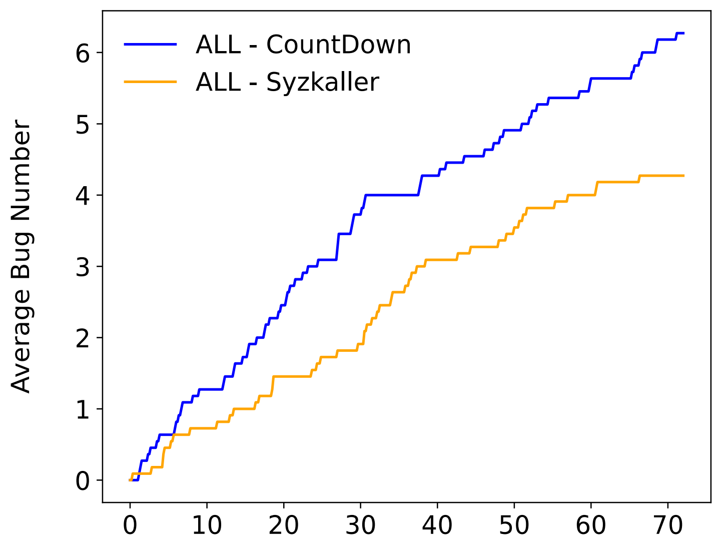
Evaluation – Bug Finding



Linux v5.15



Linux v6.1

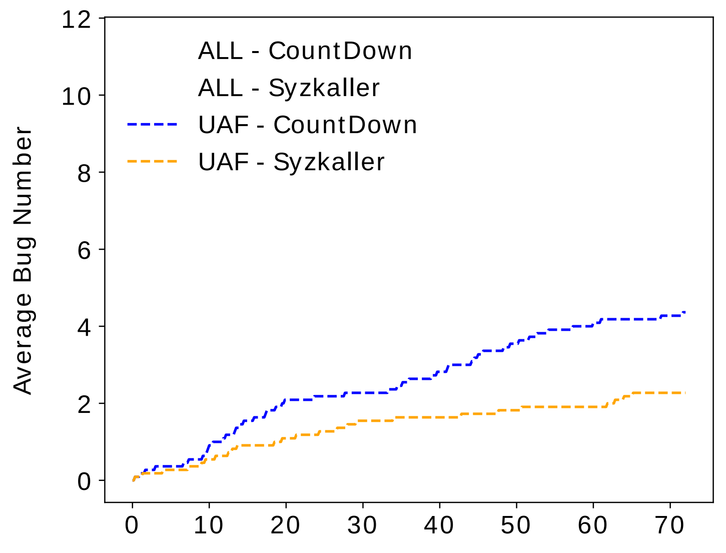


Linux v6.6

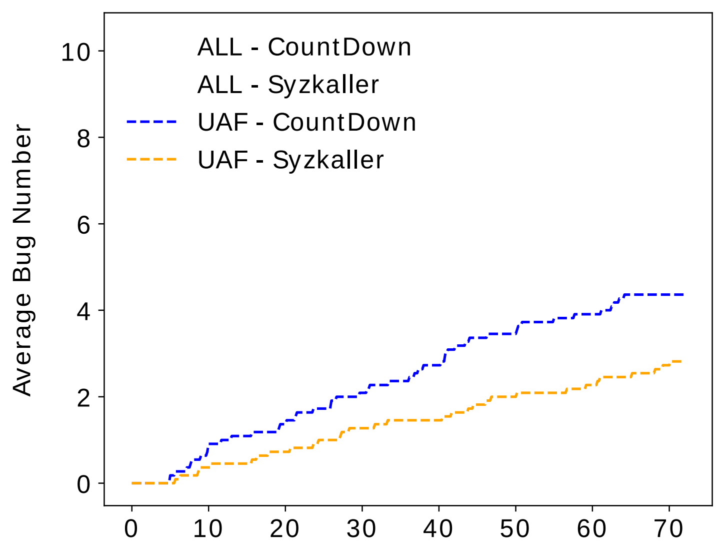
Similar results on three kernel versions

- **32.9%** more KASAN reports on average

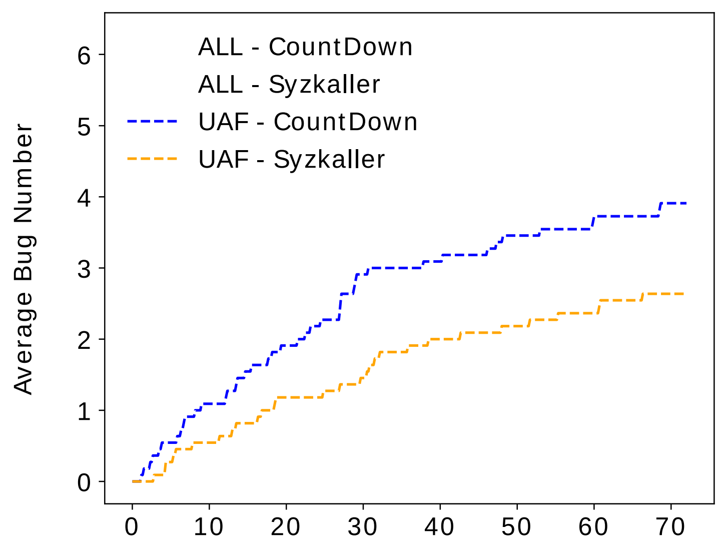
Evaluation – Bug Finding



Linux v5.15



Linux v6.1



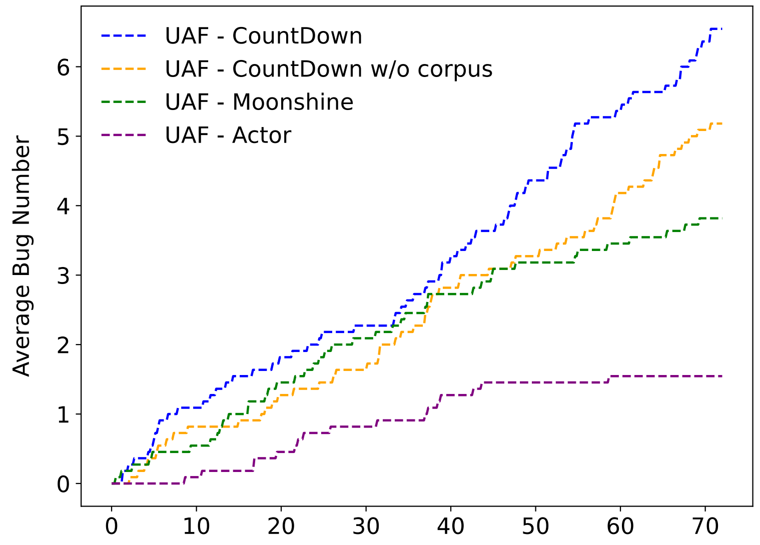
Linux v6.6

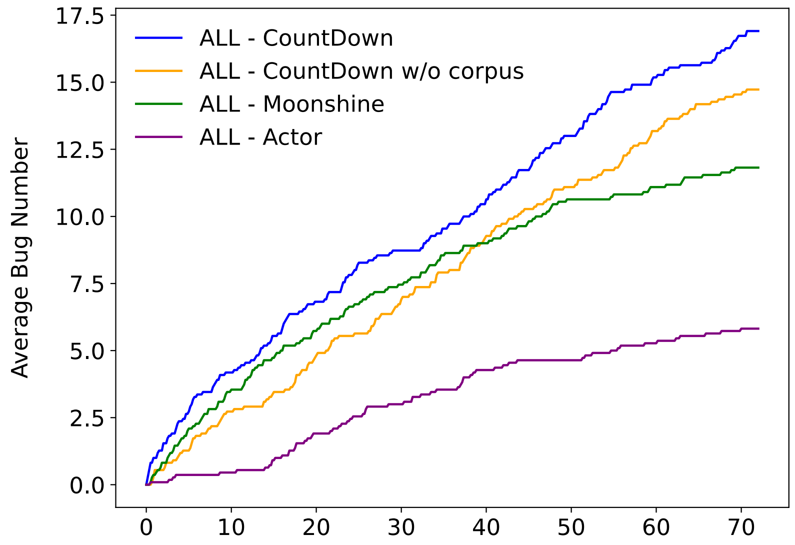
Similar results on three kernel versions

- **32.9%** more KASAN reports on average
- **66.1%** more UAF bugs on average

Evaluation – Bug Finding

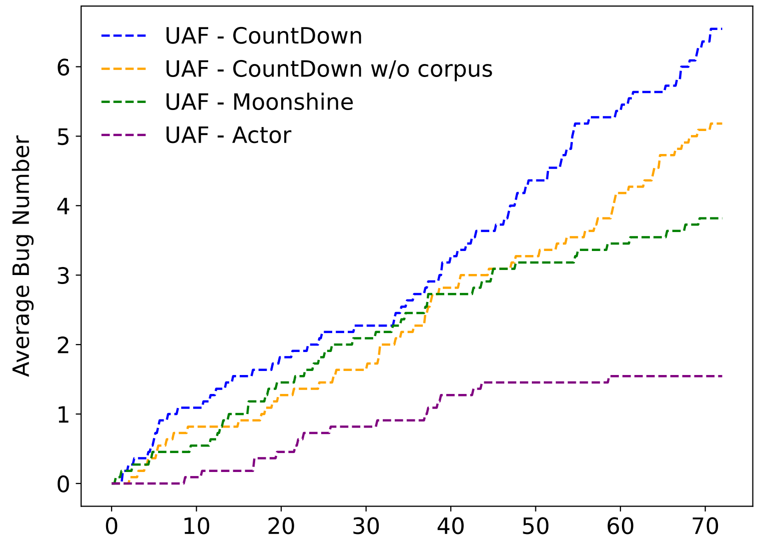
CountDown w/ syzbot corpus: the best result

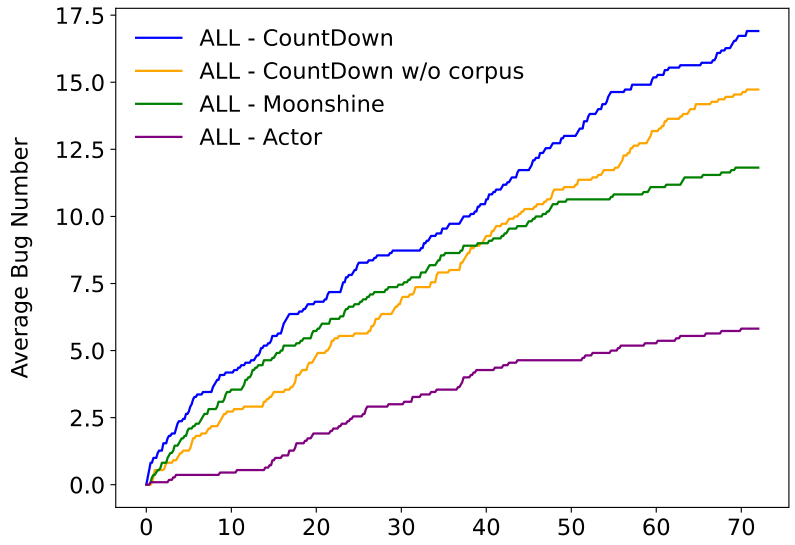




Evaluation – Bug Finding

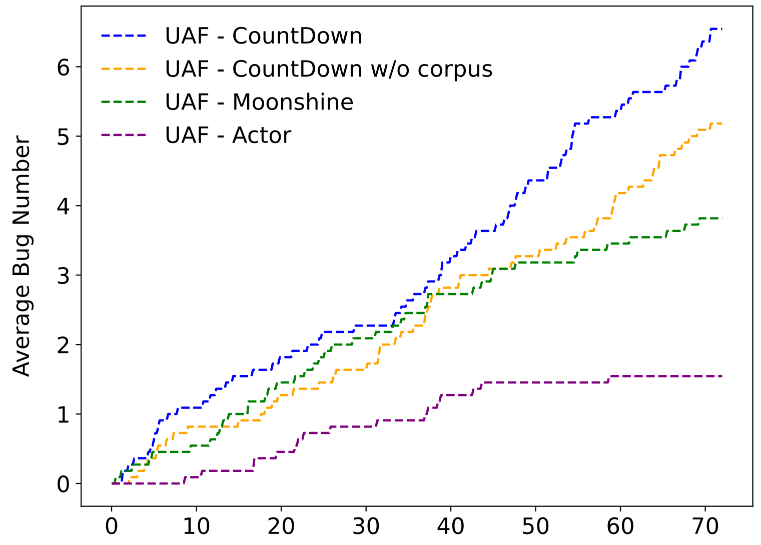
CountDown w/o corpus outperforms Moonshine and Actor

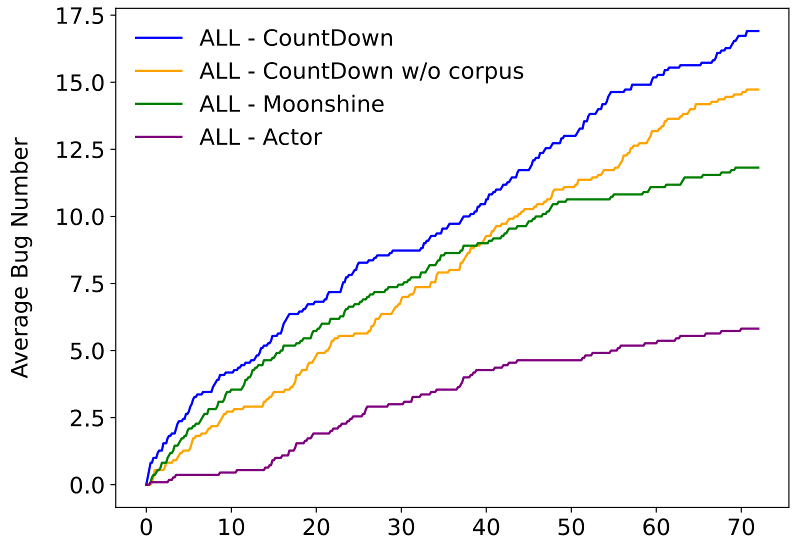




Evaluation – Bug Finding

CountDown w/o corpus outperforms Moonshine and Actor



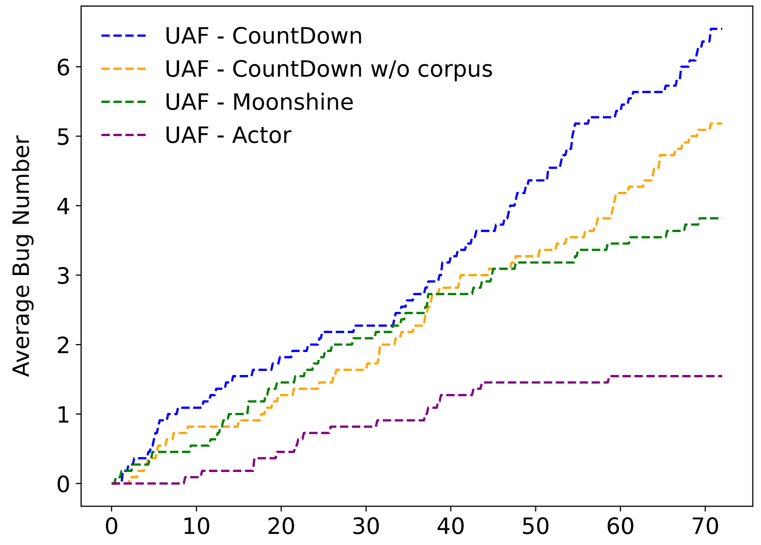


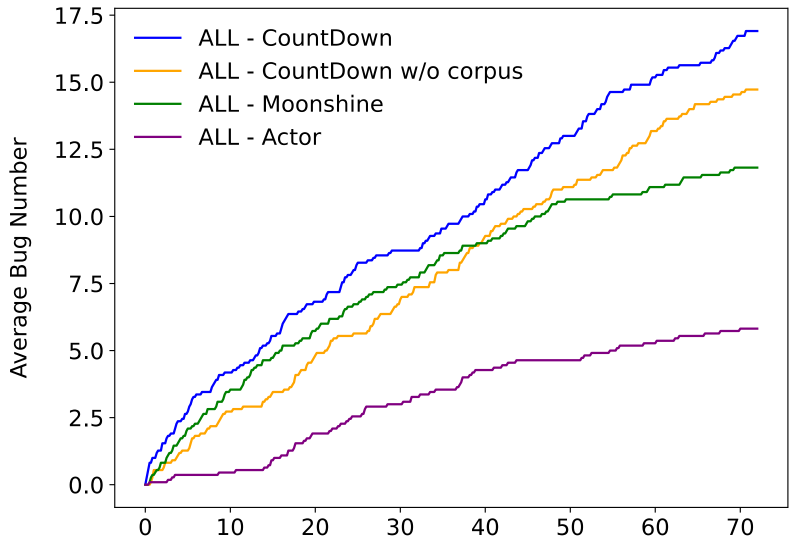
UAF bugs

- **36.8%** more than Moonshine
- **2.47x** more than Actor

Evaluation – Bug Finding

CountDown w/o corpus outperforms Moonshine and Actor





UAF bugs

- **36.8%** more than Moonshine
- **2.47x** more than Actor

KASAN reports

- **24.6%** more than Moonshine
- **1.53x** more than Actor

Evaluation – New bugs

	Bug Name	Version
1	KASAN: slab-use-after-free in __lock_acquire	v6.9
2	KASAN: slab-use-after-free in hfsplus_bnode_read	v6.9
3	KASAN: slab-use-after-free in __discard_prealloc	v6.9
4	KASAN: slab-use-after-free in jfs_readdir	v6.9
5	KASAN: use-after-free in leaf_move_items	v6.9
6	KASAN: slab-use-after-free in nfc_alloc_send_skb	v6.3
7	KASAN: use-after-free in gfs2_evict_inode	v4.19
8	KASAN: slab-out-of-bounds in gfs2_check_blk_type	v6.9
9	KASAN: slab-out-of-bounds in gfs2_invalidate_folio	v6.8
10	KASAN: slab-out-of-bounds in sock_sendmsg	v6.1
11	KASAN: slab-out-of-bounds in __crypto_xor	v4.19
12	KASAN: slab-out-of-bounds in ext4_search_dir	v4.19
13	KASAN: slab-out-of-bounds in xfs_iext_get_extent	v4.19
14	KASAN: null-ptr-deref in txBeginAnon	v6.9
15	KASAN: null-ptr-deref in mutex_lock	v4.19

15 new kernel memory bugs

(reported with reproducers)

- 7 use-after-free

Evaluation – New bugs

	Bug Name	Version
1	KASAN: slab-use-after-free in __lock_acquire	v6.9
2	KASAN: slab-use-after-free in hfsplus_bnode_read	v6.9
3	KASAN: slab-use-after-free in __discard_prealloc	v6.9
4	KASAN: slab-use-after-free in jfs_readdir	v6.9
5	KASAN: use-after-free in leaf_move_items	v6.9
6	KASAN: slab-use-after-free in nfc_alloc_send_skb	v6.3
7	KASAN: use-after-free in gfs2_evict_inode	v4.19
8	KASAN: slab-out-of-bounds in gfs2_check_blk_type	v6.9
9	KASAN: slab-out-of-bounds in gfs2_invalidate_folio	v6.8
10	KASAN: slab-out-of-bounds in sock_sendmsg	v6.1
11	KASAN: slab-out-of-bounds in __crypto_xor	v4.19
12	KASAN: slab-out-of-bounds in ext4_search_dir	v4.19
13	KASAN: slab-out-of-bounds in xfs_iext_get_extent	v4.19
14	KASAN: null-ptr-deref in txBeginAnon	v6.9
15	KASAN: null-ptr-deref in mutex_lock	v4.19

15 new kernel memory bugs

(reported with reproducers)

- 7 use-after-free
- 6 out-of-bounds

Evaluation – New bugs

	Bug Name	Version
1	KASAN: slab-use-after-free in __lock_acquire	v6.9
2	KASAN: slab-use-after-free in hfsplus_bnode_read	v6.9
3	KASAN: slab-use-after-free in __discard_prealloc	v6.9
4	KASAN: slab-use-after-free in jfs_readdir	v6.9
5	KASAN: use-after-free in leaf_move_items	v6.9
6	KASAN: slab-use-after-free in nfc_alloc_send_skb	v6.3
7	KASAN: use-after-free in gfs2_evict_inode	v4.19
8	KASAN: slab-out-of-bounds in gfs2_check_blk_type	v6.9
9	KASAN: slab-out-of-bounds in gfs2_invalidate_folio	v6.8
10	KASAN: slab-out-of-bounds in sock_sendmsg	v6.1
11	KASAN: slab-out-of-bounds in __crypto_xor	v4.19
12	KASAN: slab-out-of-bounds in ext4_search_dir	v4.19
13	KASAN: slab-out-of-bounds in xfs_iext_get_extent	v4.19
14	KASAN: null-ptr-deref in txBeginAnon	v6.9
15	KASAN: null-ptr-deref in mutex_lock	v4.19

15 new kernel memory bugs

(reported with reproducers)

- 7 use-after-free
- 6 out-of-bounds
- 2 null-ptr-deref

Conclusion

- CountDown - Refcount-guided kernel fuzzer
  - Refcount-guided mutation
  - Refcount-aware input prioritization
- Results
  - 15 new kernel bugs, including 7 UAF bugs
- Open source
  - https://github.com/psu-security-universe/countdown

References

[1] Dmitry Vyukov, and Andrey Konovalov. "Syzkaller: An Unsupervised, Coverage-guided Kernel Fuzzer." https://github.com/google/syzkaller, 2019.

[2] Shankara Pailoor, Andrew Aday, and Suman Jana. "MoonShine: Optimizing OS Fuzzer Seed Selection with Trace Distillation." In *Proceedings of the 27th USENIX Security Symposium (USENIX Security)*, pp. 729-743. 2018.

[3] Hao Sun, Yuheng Shen, Cong Wang, Jianzhong Liu, Yu Jiang, Ting Chen, and Aiguo Cui. "Healer: Relation Learning Guided Kernel Fuzzing." In *Proceedings of the 28th ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, pp. 344-358. 2021.

[4] Marius Fleischer, Dipanjan Das, Priyanka Bose, Weiheng Bai, Kangjie Lu, Mathias Payer, Christopher Kruegel, and Giovanni Vigna. "ACTOR: Action-Guided Kernel Fuzzing." In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*, pp. 5003-5020. 2023.

[5] Siliang Li, and Gang Tan. “Finding Reference-counting Errors in Python/C Programs with Affine Analysis.” In *Proceedings of the 28th Annual European Conference on Object-Oriented Programming (ECOOP),* pp. 80-104. 2014.

[6] Junjie Mao, Yu Chen, Qixue Xiao, and Yuanchun Shi. "RID: Finding Reference Count Bugs with Inconsistent Path Pair Checking.” In *Proceedings of the 21st ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 531-544. 2016

[7] Xin Tan, Yuan Zhang, Xiyu Yang, Kangjie Lu, and Min Yang. "Detecting Kernel Refcount Bugs with Two-Dimensional Consistency Checking." In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*, pp. 2471-2488. 2021.

[8] Jian Liu, Lin Yi, Weiteng Chen, Chengyu Song, Zhiyun Qian, and Qiuping Yi. "LinKRID: Vetting Imbalance Reference Counting in Linux kernel with Symbolic Execution." In *Proceedings of the 31st USENIX Security Symposium (USENIX Security)*, pp. 125-142. 2022.

Thank You

Question?

shuangpengbai@psu.edu