Actor: Action-Guided Kernel Fuzzing

Marius Fleischer[1]*, Dipanjan Das[1]*, Priyanka Bose[1], Weiheng Bai[2], Kangjie Lu[2], Mathias Payer[1,3], Christopher Kruegel[1], Giovanni Vigna[1]

[1]University of California, Santa Barbara [2]University of Minnesota [3]EPFL

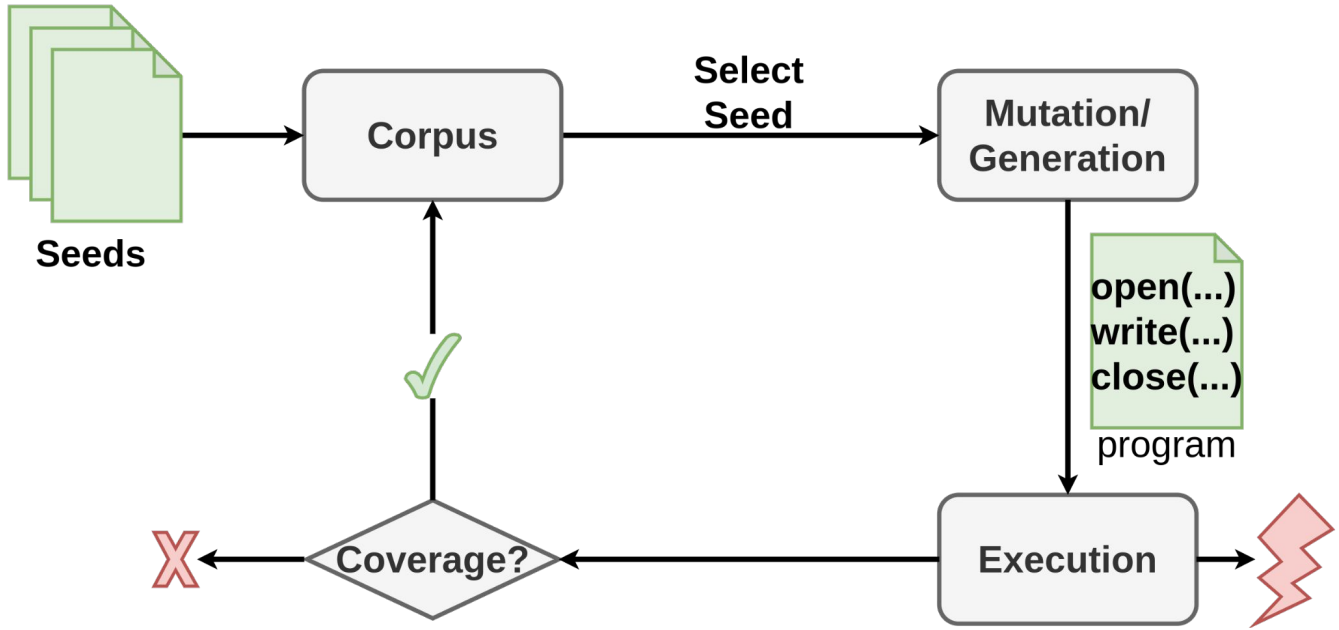
*equal contribution

Kernel Security Analysis

- OS kernel is management layer between hardware and applications
- Google continuously fuzzes several *nix kernels using syzbot
- Some bugs persist for up to ~15 years
- Two privilege escalation bugs in the Ubuntu Linux kernel

Source: https://blog.grimm-co.com/2021/03/new-old-bugs-in-linux-kernel.html
https://www.bleepingcomputer.com/news/security/almost-40-percent-of-ubuntu-users-vulnerable-to-new-privilege-elevation-flaws/

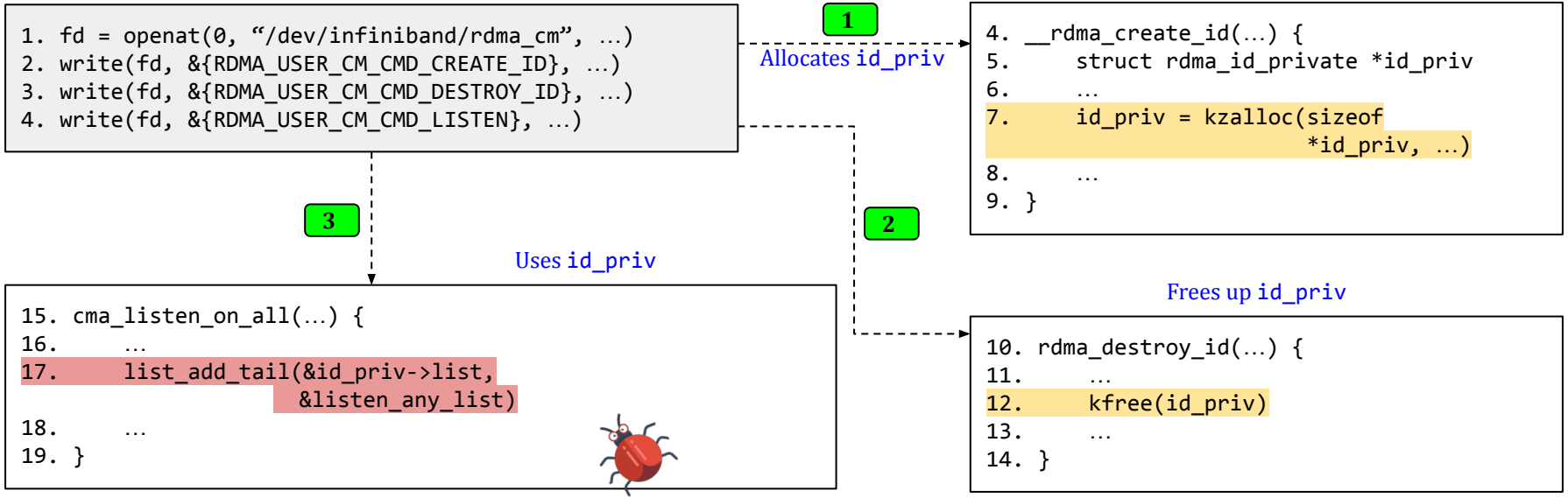
Coverage-Guided Kernel Fuzzing



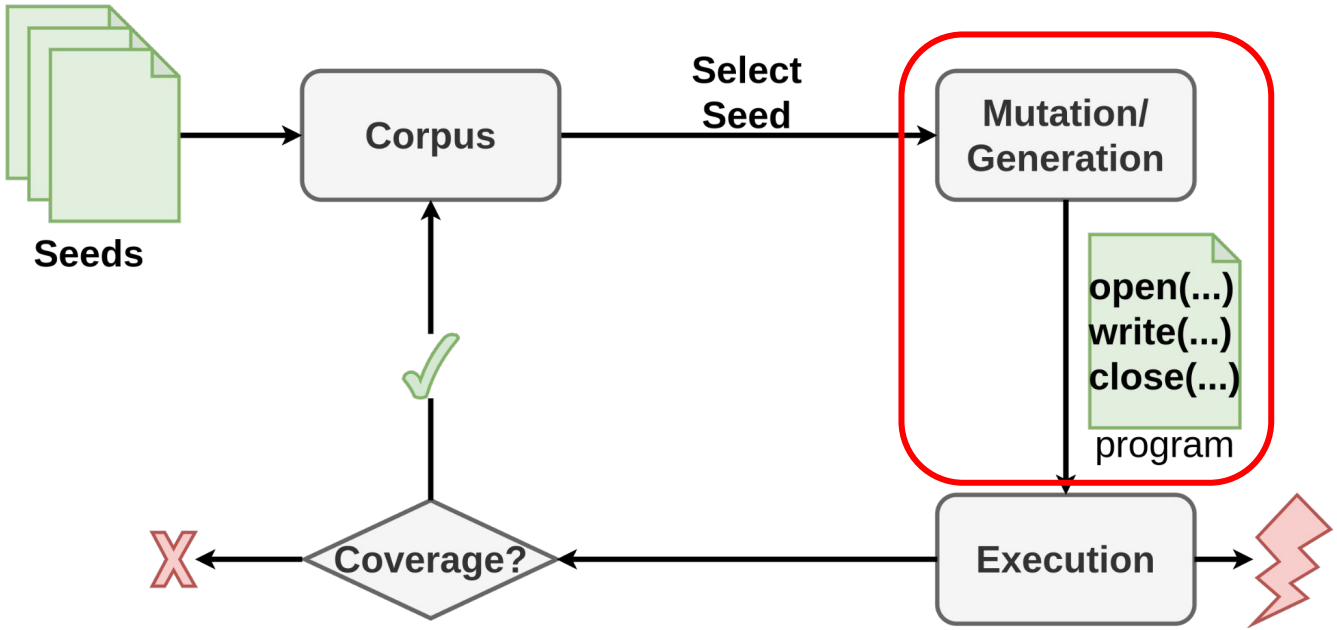
Beyond Code Coverage

- Most state-of-the-art fuzzers are coverage-maximizing
- Coverage is necessary, but not sufficient for finding bugs:
  - Memory-related bugs, e.g., Use-After-Free, Double-Free, Uninitialized Read
  - Spatial constraint: operations need to affect same memory
  - Temporal constraint: operations need to be performed in the right order

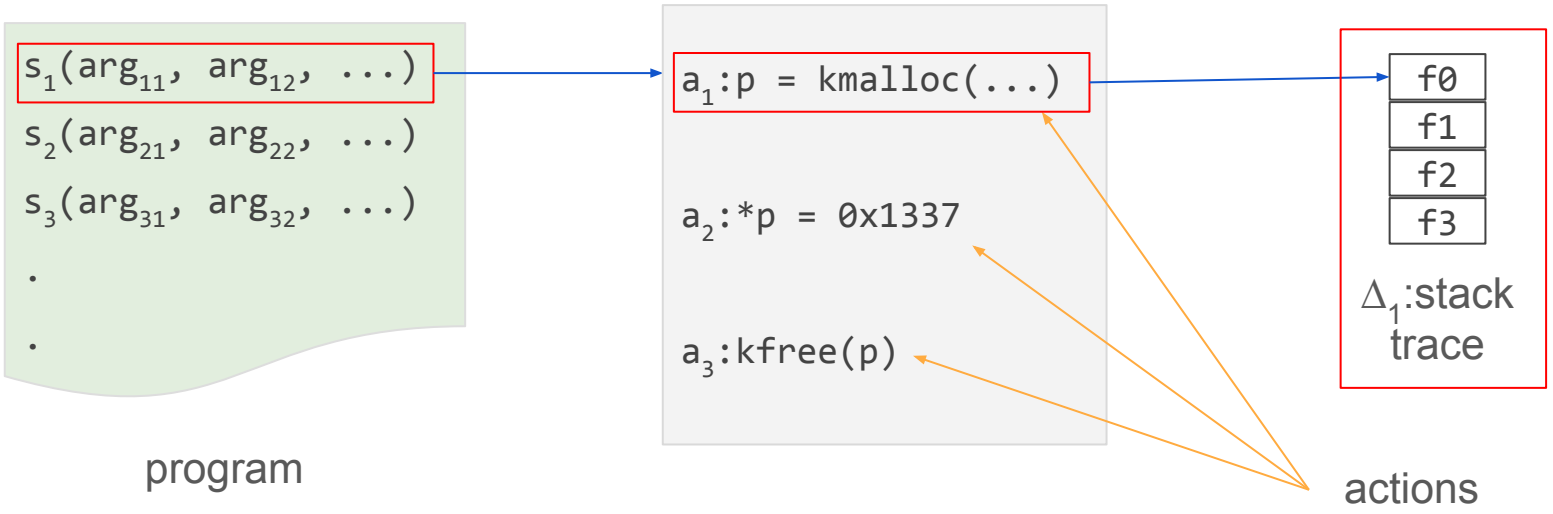
Example: Use-After-Free Linux Bug



Beyond Code Coverage



Darts: connecting system calls to actions and contexts



➔ Dart captures which system call triggers what action with its context

Actions

- Action is defined by its type, address and size
- Current prototype supports only heap-related action types
  - Allocation
  - Value read
  - Pointer read
  - Index read
  - Deallocation
  - Value write
  - Pointer write
  - Index write

Action-Guided Fuzzing

- We propose **action-guided** fuzzing, a novel technique to synthesize potentially bug-inducing programs guided by actions

  - Action-mining: Collect darts (actions triggered by system calls)
  - Program synthesis: Generate bug-inducing system call sequences based on predefined templates

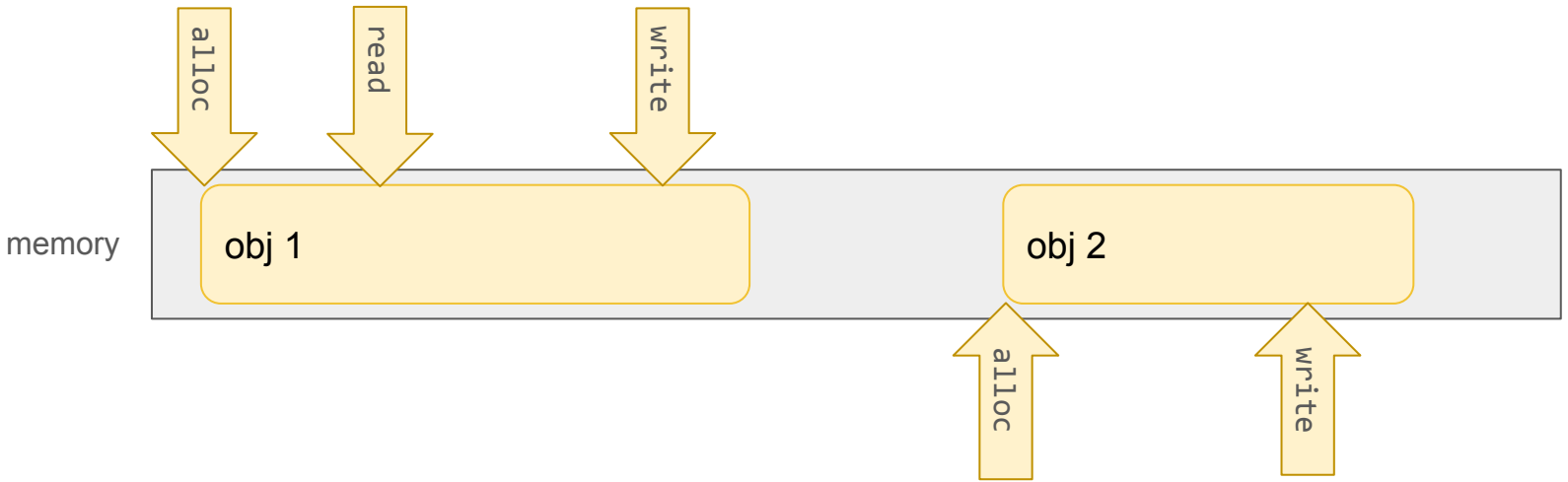
    - Use-After-Free
    - Memory Leak
    - Double-Free
    - Invalid Free
    - Uninitialized Read
    - Out-of-Bounds
    - Null-Pointer Dereference

Action Mining

- Infer relationships between system calls and the actions they trigger
- Find system calls that access shared memory
    - ➜ Spatial constraint
- Result: Groups of darts operating on shared memory

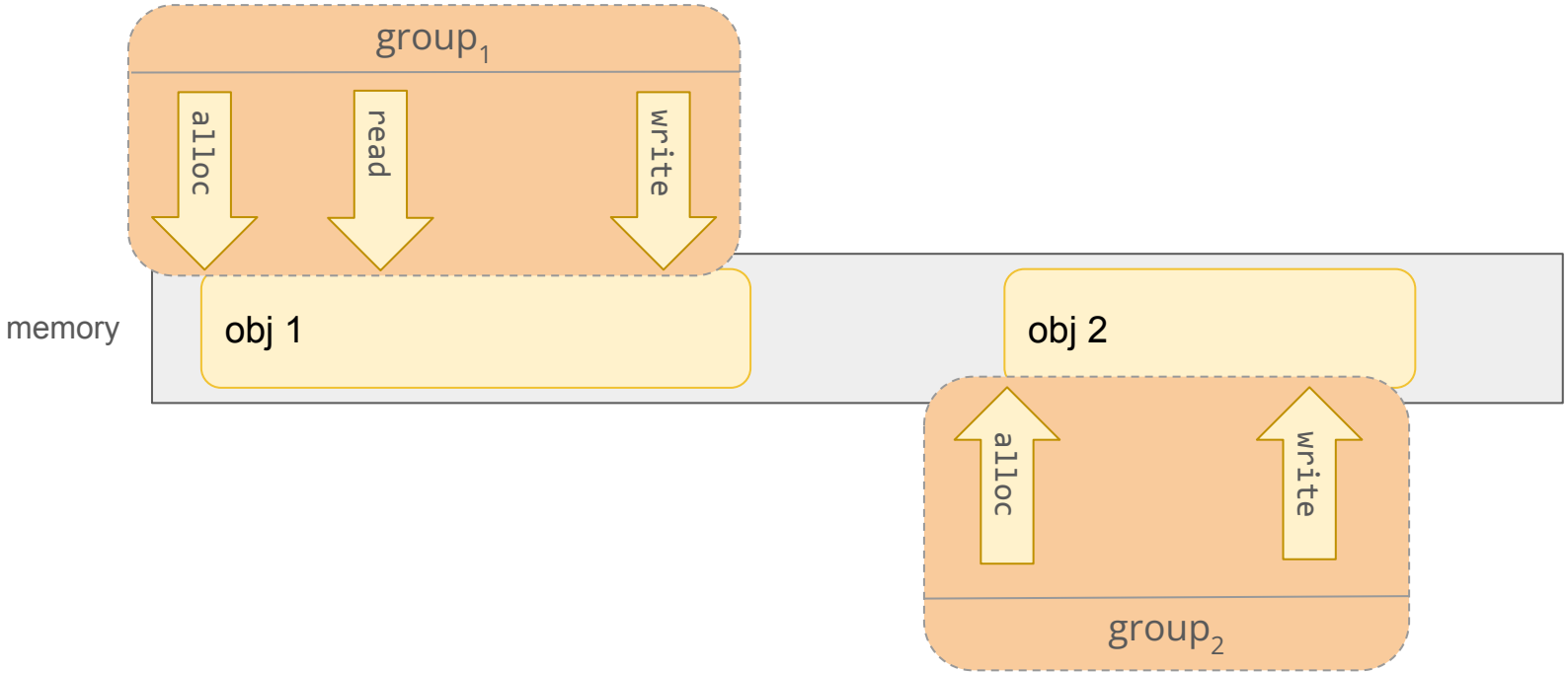
Action Mining – Grouping

Combine related darts from the **one** program together



Action Mining – Grouping

Combine related darts from the **one** program together

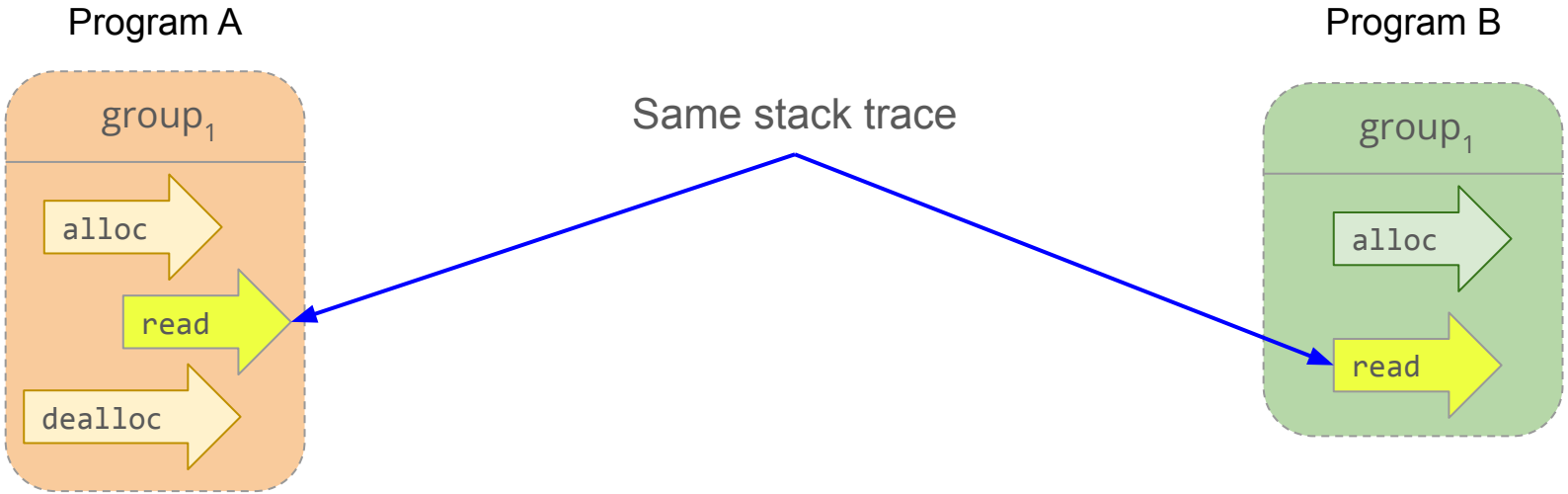


Action Mining – Merging

Learn relation between groups generated from **different** programs

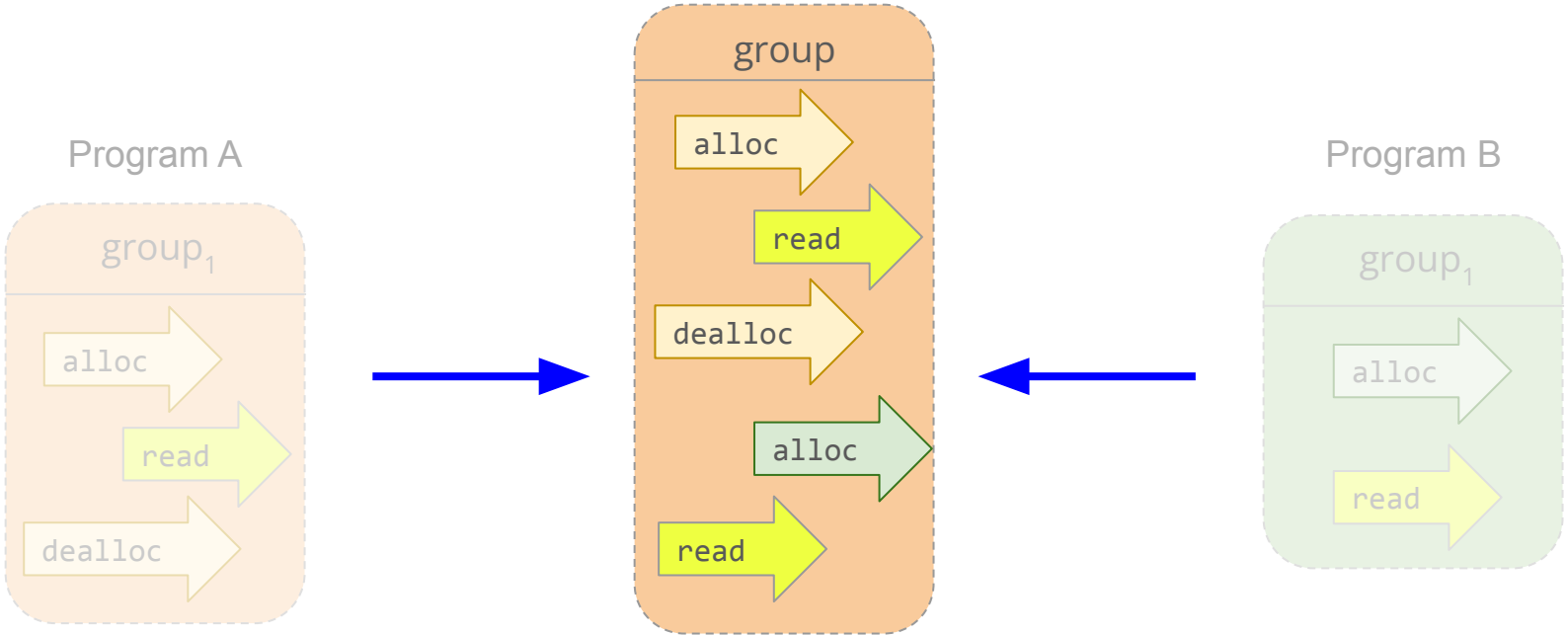
Action Mining – Merging

Learn relation between groups generated from **different** programs

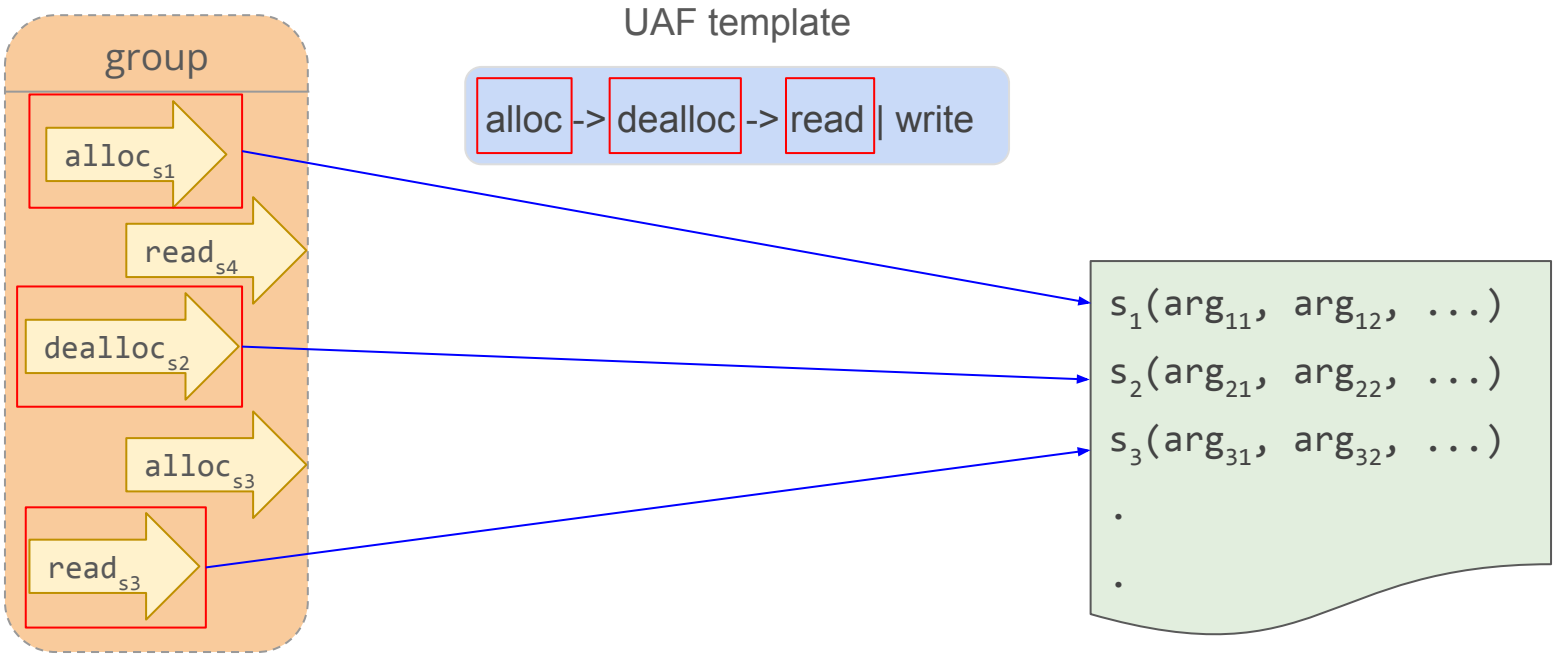


Action Mining – Merging

Learn relation between groups generated from **different** programs



Program Synthesis



Evaluation

- RQ1: Quality of darts?
- RQ2: Bug-inducing programs – action-guided vs coverage-guided
- RQ3: New bugs?

- Target Kernel
  - RQ1 & RQ2: v5.17 (kernel used during development)
  - RQ3: v5.4.206 (LTS), v5.10.131 (LTS), v5.19 (stable), v6.2-rc5 (latest)

RQ1: Quality of Darts

- Darts are executed on a **different** kernel state
  - **May not** reproduce the intended action

Action	Success (%)	Action	Success (%)
Alloc	68.07	Val Write	32.91
Val Read	38.91	Ptr Write	56.27
Ptr Read	38.18	Idx Write	18.49
Idx Read	29.37	Dealloc	42.92
Overall	54.68		

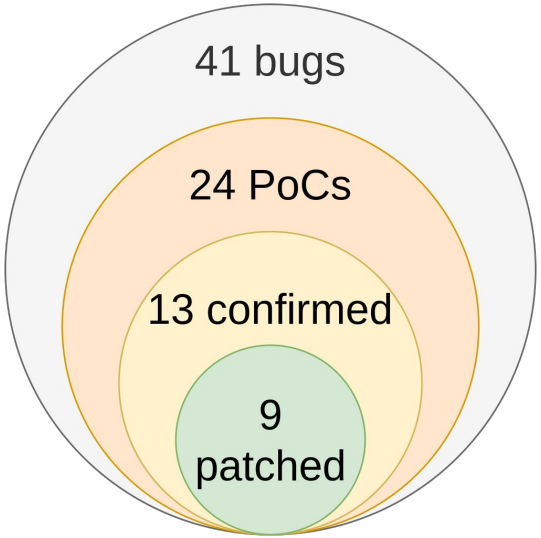
RQ2: Program Diversity

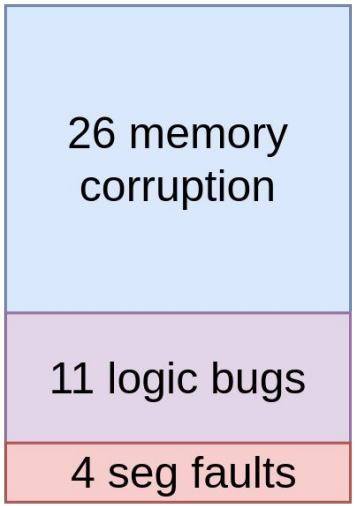
- Does Actor generate more bug-inducing programs than syzkaller?

Bug Class Templates	Improvement	Bug Class Templates	Improvement
Use-After-Free	22.46x	Null Pointer Deref	11.44x
Double Free	28.53x	Invalid Free	1.16x
Out of Bounds 1	20.93x	Memory Leak 1	1.20x
Out of Bounds 2	37.84x	Memory Leak 2	21.70x
Uninitialized Read	3.01x		

RQ3: New bugs

41 new bugs in four versions of the Linux kernel





Conclusion

- **Action-guided fuzzing** is a novel input generation strategy for kernel fuzzing
- Action-guidance **complements**, but **does not competes** with coverage
- Actor, our prototype implementation of action-guided kernel fuzzing
- Discovered 41 previously unknown vulnerabilities in well-tested and actively-patched LTS and stable Linux kernel versions

mariusfleischer@ucsb.edu

https://github.com/ucsb-seclab/actor