AMBER

北京大学
PEKING UNIVERSITY

Fuzzing the Latest NTFS in Linux with Papora: An Empirical Study

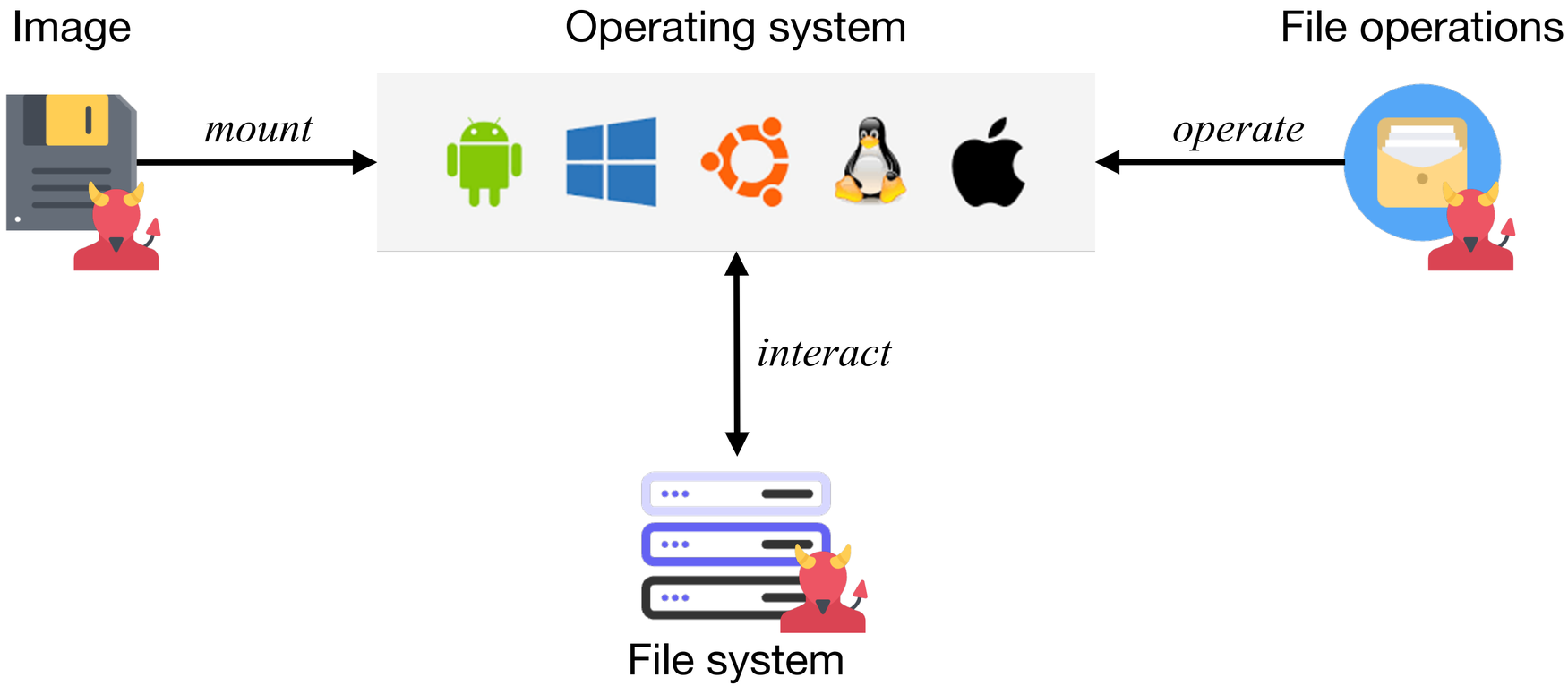
Edward Lo[1*], *Ningyu He*[2*], Yuejie Shi[1], Jiajia Xu[1], Chiachih Wu[1], Ding Li[2], and Yao Guo[2]

[1]Amber Group
[2]Peking University

2023.5.25

File System

Key components of file systems



NTFS

New Technology File System

- A proprietary journaling file system developed by Microsoft
- Default file system of the Windows NT family starting NT 3.1
- NTFS3 was firstly upstreamed to Linux kernel in late 2021
- A new file system is complicated enough to have some bugs
- Existing fuzzers cannot efficiently fuzz a new file system

Challenges

Challenges to fuzz a file system

Fuzzing image

- **Images are large**
  - Only metadata matters
  - Mutation on user data is basically a waste of time
- **Each file system has their own metadata structure design**
  - Need to develop a specific parser for the file system
- **Checksums**
  - Corrupted after mutation, which could lead to mount fails

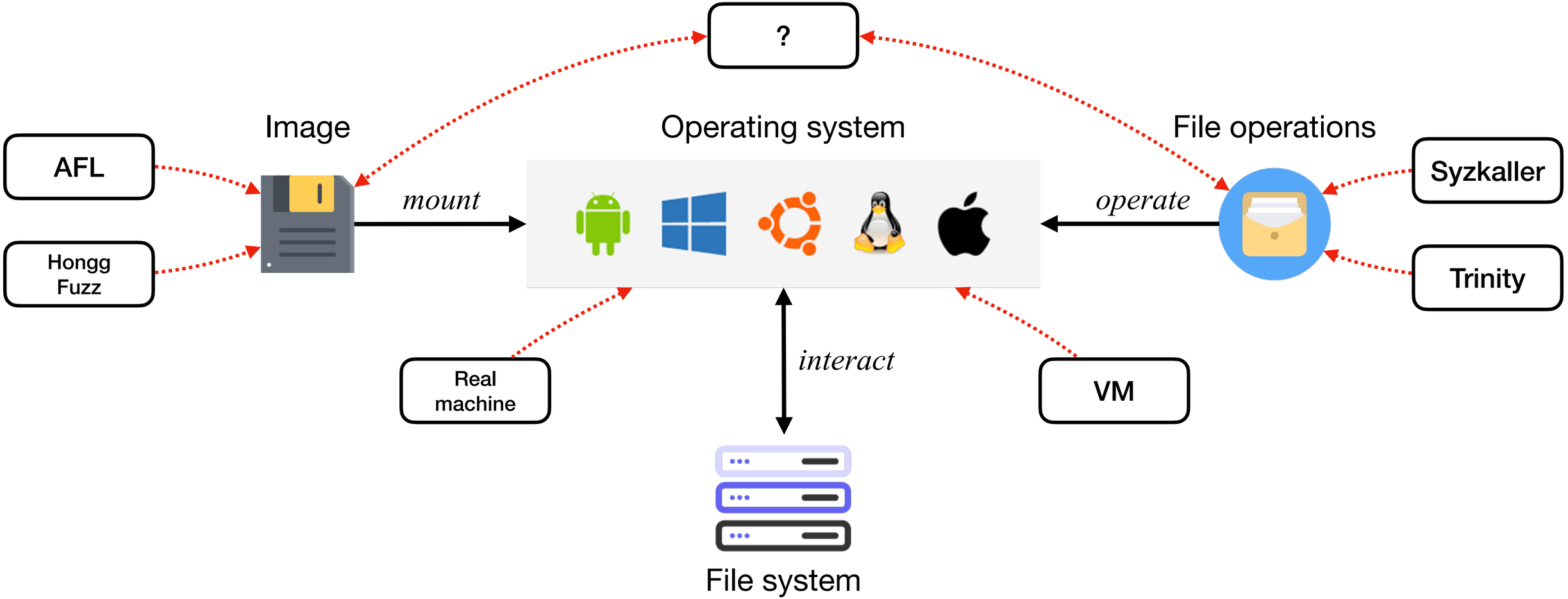
Fuzzing file operation

- **What to generate**
  - Syscalls for file operations
- **How to mutate**
  - The fuzzer should know how to mutate each arguments of the syscalls
  - A valid fd, combination of flags, an allocated buffer…
- **Context awareness**
  - The context should be maintained across each syscalls

```
int fd = open(“papora.seed”, …);
read(fd, buf, 256);
close(fd);
```

Proper Weapon

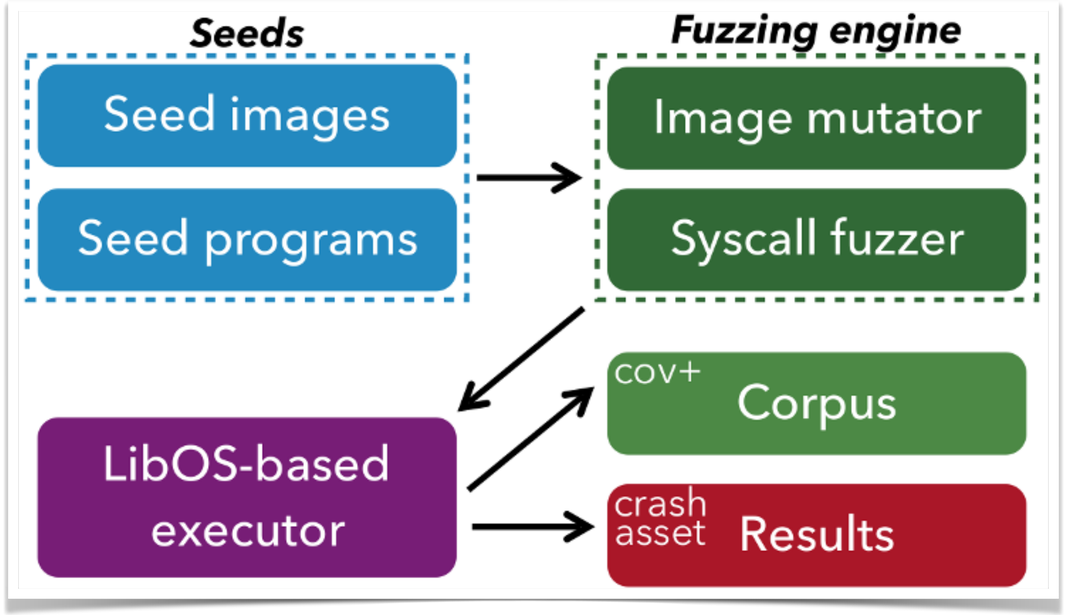
Please choose your weapon



Janus

Published on S&P'19

- In 2019, Georgia tech SSLab proposed Janus - a coverage-driven fuzzer that efficiently and effectively test images and file operations in a joint manner
- However, we can't use it for our target file system (NTFS)
  - Need a specific image parser for NTFS (more about it later)
  - The library (Linux kernel library) used by executor is obsolete (v5.3) and inactive at the surveying time
  - KASAN patch integration and modification for the evolving new kernel



The structure and workflow of Janus

NTFS Sanity Check

Lots of sanity check on metadata

```
if (memcmp(boot->system_id, "NTFS    ", sizeof("NTFS    ") - 1)) {
        ntfs_err(sb, "Boot's signature is not NTFS.");
        goto out;
}
```

OEM ID must equal “NTFS “ (4 spaces)

```
boot_sector_size = ((u32)boot->bytes_per_sector[1] << 8) |
                   boot->bytes_per_sector[0];
if (boot_sector_size < SECTOR_SIZE ||
    !is_power_of_2(boot_sector_size)) {
        ntfs_err(sb, "Invalid bytes per sector %u.", boot_sector_size);
        goto out;
}
```

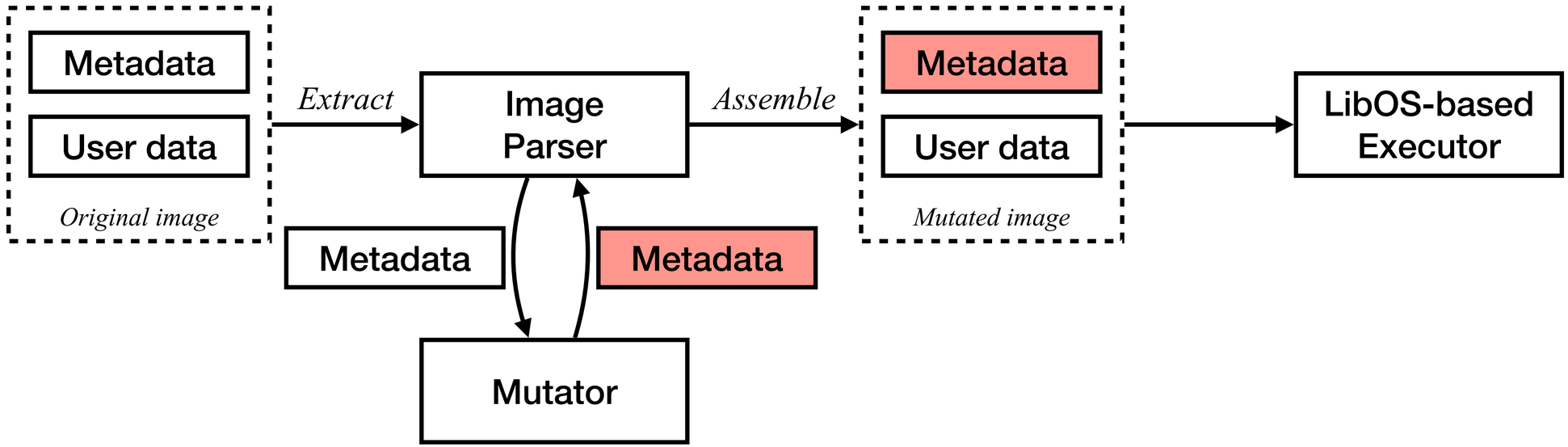
Sector size >= 512 and must be a power of 2

```
sct_per_clst = true_sectors_per_clst(boot);
if ((int)sct_per_clst < 0 || !is_power_of_2(sct_per_clst)) {
        ntfs_err(sb, "Invalid sectors per cluster %u.", sct_per_clst);
        goto out;
}
```

Cluster size must be a power of 2

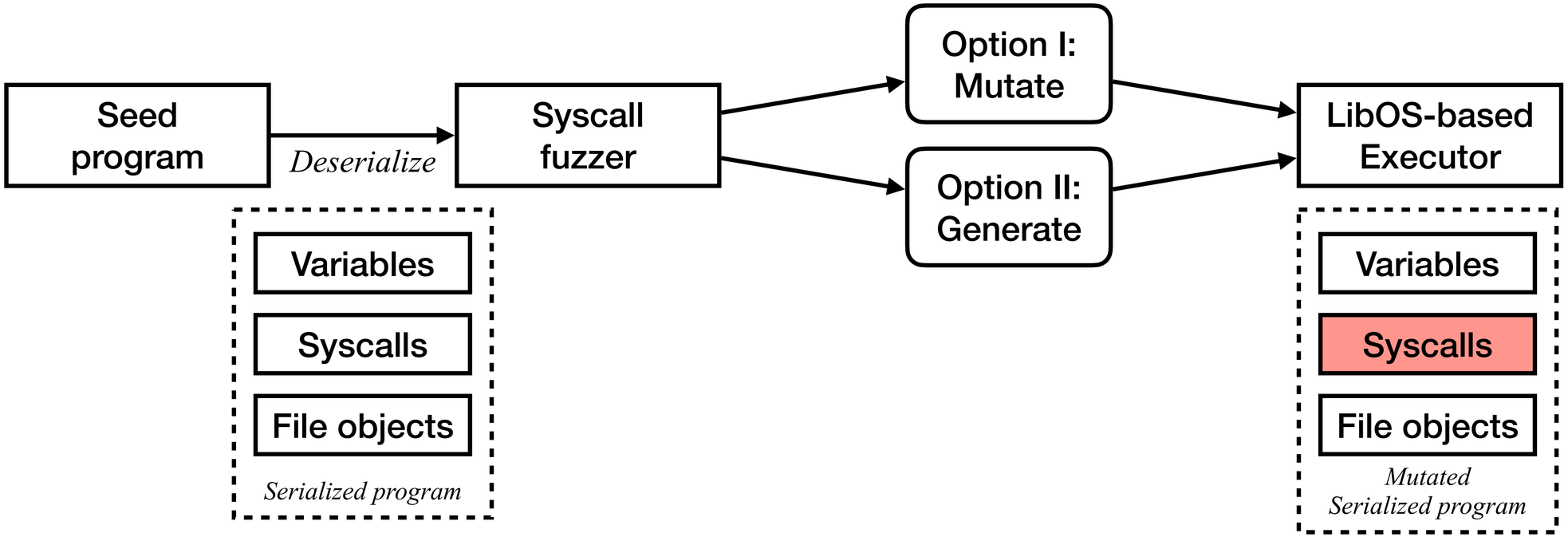
Papora Image Parser

The workflow of the Papora's image parser



Papora Syscall Fuzzer

The workflow of the Papora's syscall fuzzer



Executor

Choose an executor: which one?

	Real machine	**VM**
Speed	Fast	Slow
Scalability	Spend money	Spawn more VMs
Management (reboot / debug / etc)	Hard	Easy
Aging OS issue	No	Yes

Executor

Choose an executor: LibOS!

Pros

- Fast execution (~device)
- Easy management (reboot / debug / etc)
- Easy to scale
- Easy to reproduce (non-aging kernel)

Cons

- Since LKL is an arch of Linux, there are some limitations of current implementation, e.g., ! MMU / !SMP / etc
- Kernel upgrading effort

Executor

Choose an executor: LibOS!

Cons

- Since LKL is an arch of Linux, there are some limitations of current implementation, e.g., ! MMU / !SMP / etc
- Kernel upgrading effort

Add KASAN support

Upgrade it!

Evaluation

Syzcaller vs. Papora

- Run Syzkaller for 1 month with the customized syz-lang description
  - Run Syzkaller for 1 month with the customized syz-lang description
  - No interesting outcome 🥲
- Run Papora for 3 months intermittently
  - Upgrade LKL whenever new kernel is available (v5.15 -> v6.0)
  - Identified 12 issues: 3 vulnerabilities (assigned CVEs), 9 bugs

Commit	Bug Type	Root Cause	Upstreamed
Type I			
0b66046	NPD	Sanity check miss	✓
e19c627	OOB Read	Arithmetic overflow	✓
6db6208	OOB Read	Sanity check miss	✓
2681631	NPD	Sanity check miss	✓
c1ca8ef	NPD	Implementation flaw	✓
4f1dc7d (**CVE-2022-48424**)	Heap Corruption	Sanity check miss	✓
bfcdbae	OOB Read	Sanity check miss	✓
e6ffad3	OOB Read	Sanity check miss	
467333a (**CVE-2022-48425**)	Heap Corruption	Type confusion	
f64633f	OOB Read	Sanity check miss	
Type II			
4d42ecd	OOB Read	Sanity check miss	✓
54e4570 (**CVE-2022-48423**)	OOB Write	Sanity check miss	✓

All identified bugs.
Type I: crashed once the image is mounted
Type II: crashed once the program is executed

Takeaways

Some sound bytes

- Complicated and hard-to-fuzz software are good targets for security researchers
- File system maintainers should pay more attention on metadata integrity, and users should be cautious on mounting an disk image
- We have open-sourced both **Papora** and **the upgraded LibOS with KASAN integrated** at: https://github.com/ambergroup-labs/papora

AMBER

北京大学
PEKING UNIVERSITY

Q&A

Contact us: chiachih.wu@ambergroup.io / ningyu.he@pku.edu.cn