# FUZZING FILE SYSTEM IMPLEMENTATIONS

ON BSD BASED OPERATING SYSTEMS

~~LAB~~ LOCKDOWN EDITION

## WHOAMI

- Christopher Krah
  - : @0xricksanchez
  - : 0xricksanchez
  - : ricksanchez
  - : christopher.krah@fkie.fraunhofer.de
- B.Sc - Comp. Sci. – 2017
- M.Sc Comp. Sci. - 2019
- Security Researcher @ Fraunhofer FKIE in Germany – 2019
- Interests: UNIX, IoT, RE, Exploit-Dev. & Fuzzing

## OUTLINE

## WHY...

- fuzzing?
- *BSD?
- file systems?
- not use tool X?

## FUZZING

- "Fuzzing renaissance"
- Allows for deeply inspecting a project
- Manual bug hunting not scalable
- Fun

## ck@hitb:~/why$

*BSD

PlayStation 3

NINTENDO SWITCH

NETFLIX

TOTAL RESULTS

302,951

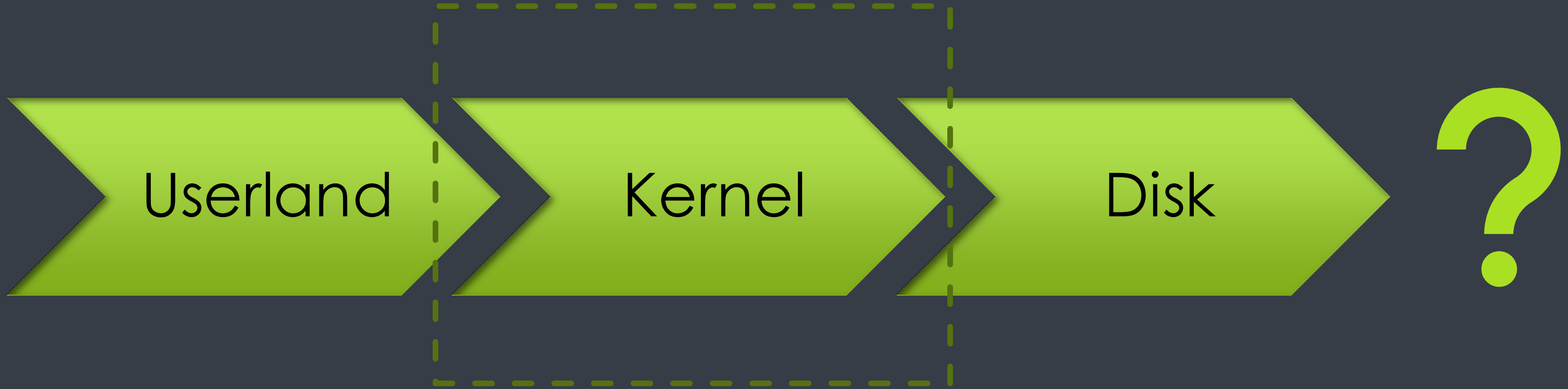
•••

- Not interested in Windows
- Also everybody does GNU/Linux…

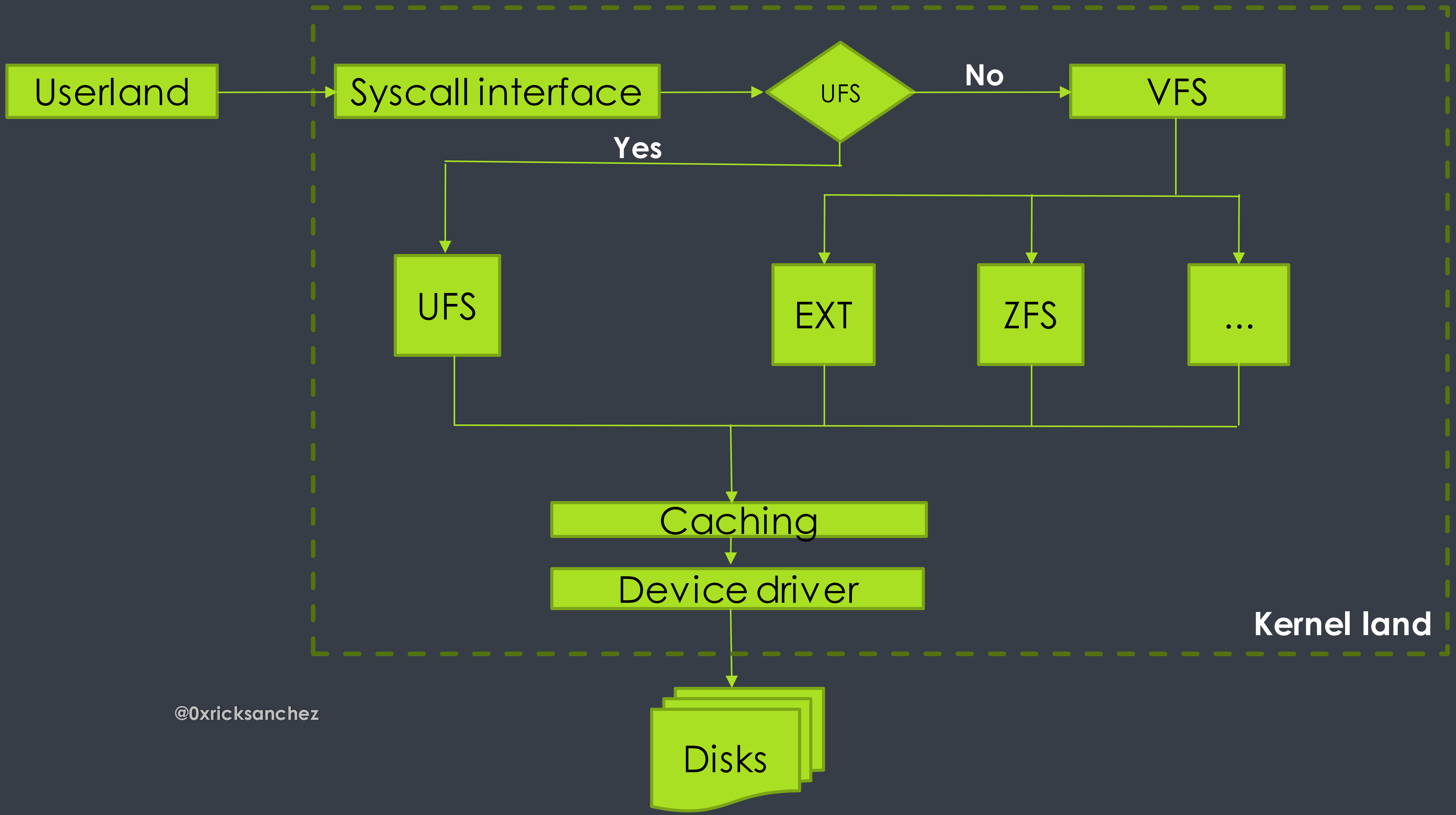
➡ Not all Systems tested equally well

➡ So Why not check out the BSDs!

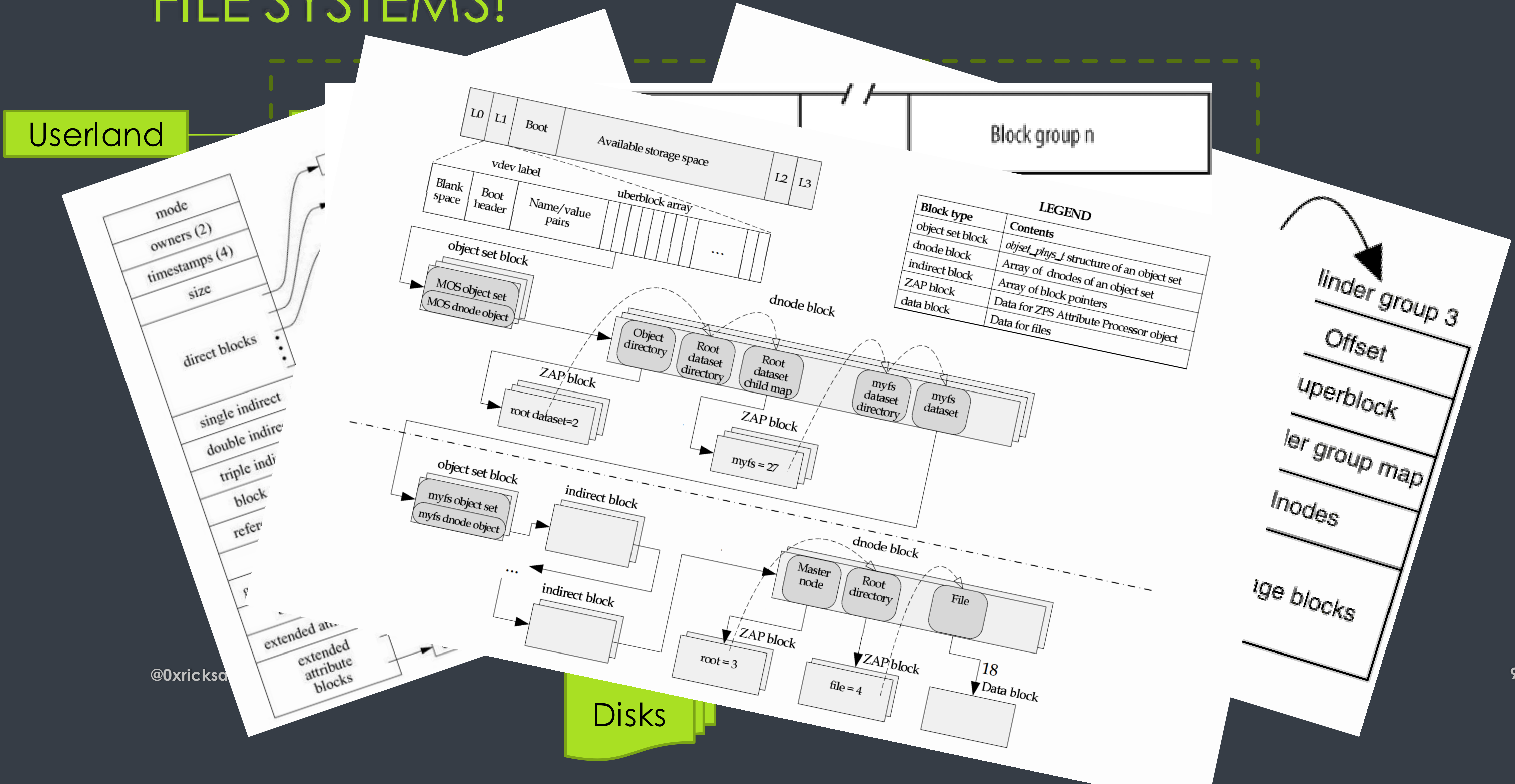
## FILE SYSTEMS?



## FILE SYSTEMS!



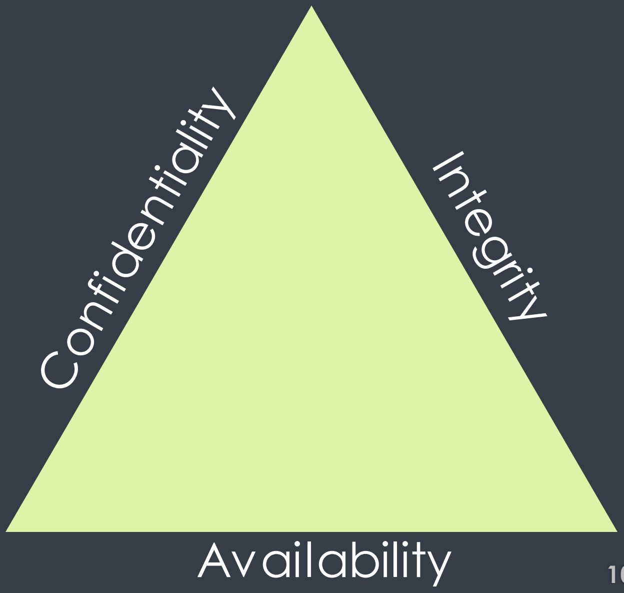
## FILE SYSTEMS!



| Block type | Contents |
|---|---|
| object set block | *objset_phys_t* structure of an object set |
| dnode block | Array of dnodes of an object set |
| indirect block | Array of block pointers |
| ZAP block | Data for ZFS Attribute Processor object |
| data block | Data for files |

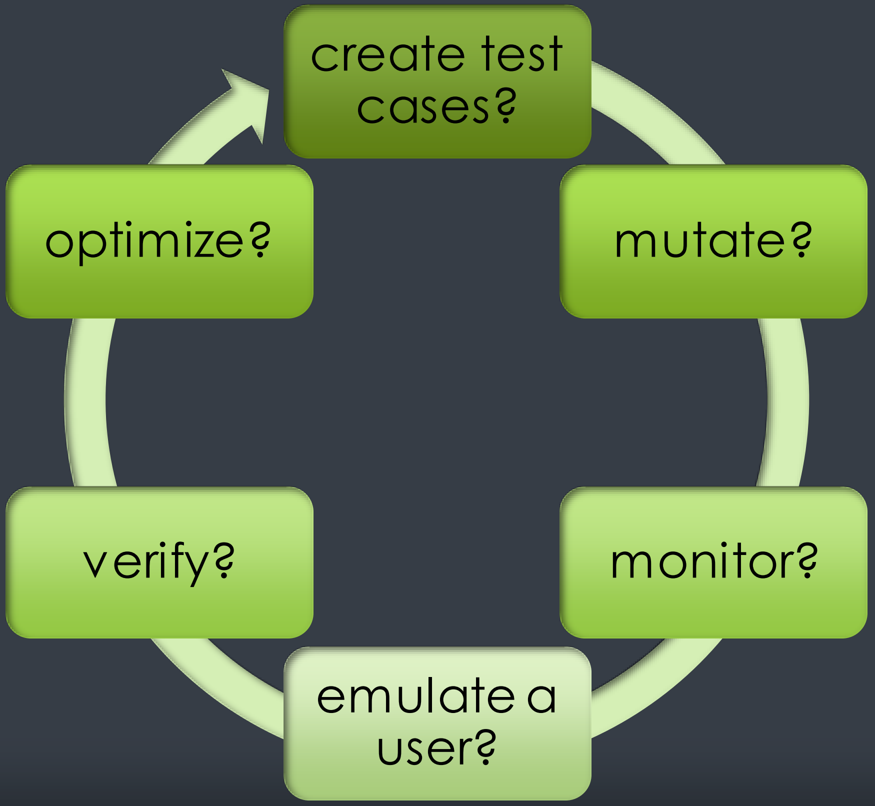
## So Why Filesystems afterall?

- Filesystems often overlooked
- However:
  - At least Availablity of data should be ensured/tested for
  - Additionally: Daily Usage of e.g. USB drives
  - Ultimately, filesystems == kernel code execution



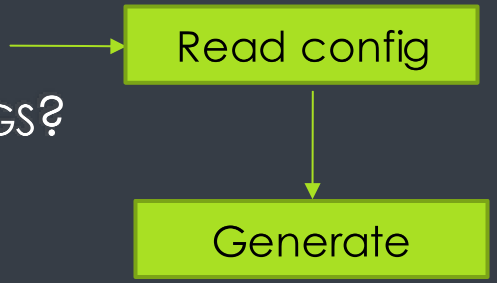
## Why not use 'X' for kernel fuzzing?

- Interested in the complete execution chain
  - Metadata parsing
  - Mounting
  - Accessing
  - Modifcation
  - Unmounting


create test cases?
mutate?
monitor?
emulate a user?
verify?
optimize?

## HOW TO...

## 1. Test case generator



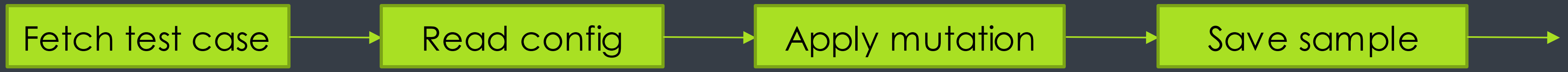
- What's a valid test case when looking for file systems bugs?
  - An actual disk image!
- Automatic generation of
  - (non-) populated FS with variable sizes
  - Currently supported: UFSv1/v2, ZFS, EXT2/3/4, APFS

Observation: Avoid headaches by using the same OS for target and host..

## 2. Mutation?

- Zero-/FF-out/Randomize superblocks, cylinder groups, single bytes
- Targeted mutations in superblock(s)
- (Deterministic) Full binary mutation via radamsa

➢ Observation: 'Dumb mutations' often enough*



## 2. Mutation?

| FS | #good_mounts | #bad_mounts |
|---|---|---|
| UFS | ~ 20% | ~ 80% |
| EXT | ~ 80% | ~ 20% |
| ZFS | ~ 7.5% | ~ 92.5% |

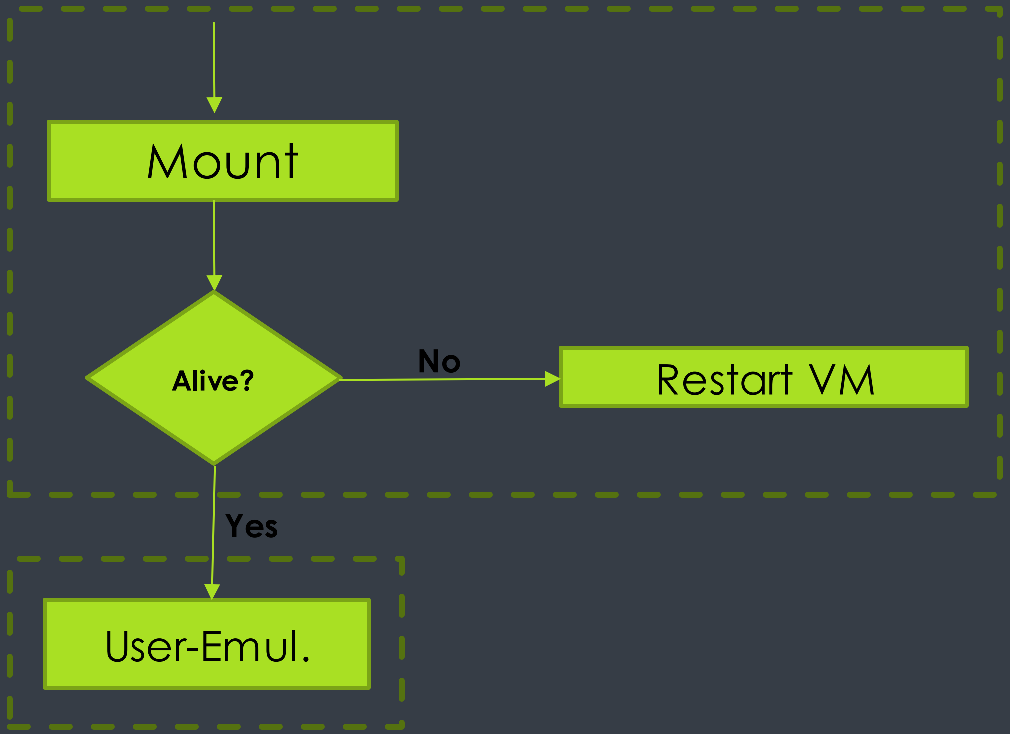
| Scenario | Result |
|---|---|
| Pool not recognized | Not importable |
| *Pool metadata corrupted* | Not importable |
| *One or more devices contain corrupted data* | Not importable |
| *Valid pool* | Importable |

## 2. Mutation?

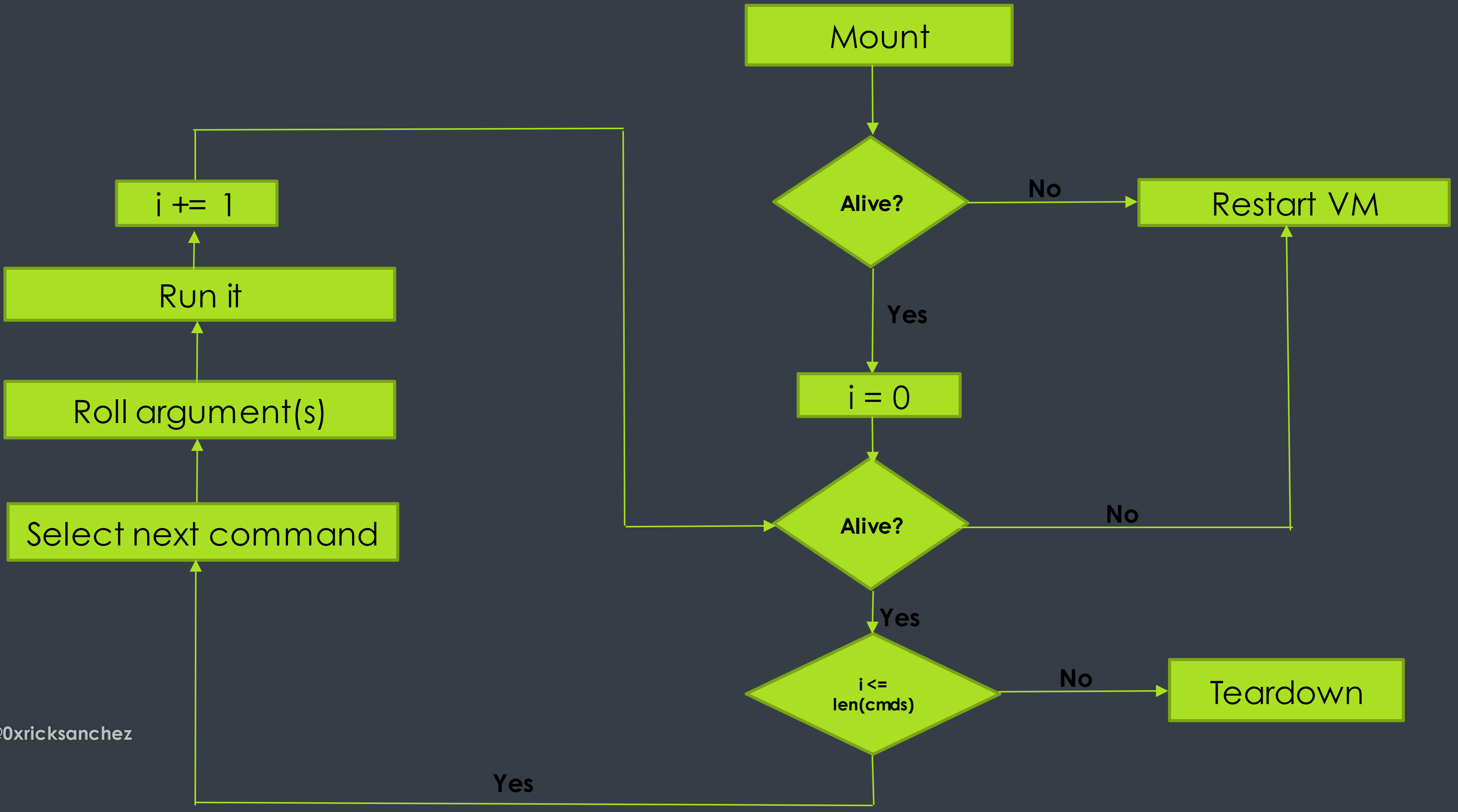
| FS | #good_mounts | #bad_mounts |
|---|---|---|
| UFS | ~ 20% | ~ 80% |
| EXT | ~ 80% | ~ 20% |
| ZFS | ~ 7.5% | ~ 92.5% |

- Observation: Inverse correlation between UFS and EXT
- Observation: Integrity checks of ZFS

## 3. User Emulation



## 3. User Emulation



## 3. User Emulation

| Category | Operation |
|---|---|
| changing geometry | chflags, chgrp, chmod, chown, mv, rm, truncate* |
| extending geometry | cp, dd, echo, ln, mkdir, mknod, split*, touch |
| parsing geometry | basename*, chdir, dirname*, du*, file, find, getfacl*, ls, readlink, stat, wc* |

➡ Static vs. Randomized order

➡ Static vs. Randomized Arguments

## 3. User Emulation

| FS | Static User-Emulation | | Random User-Emulation | |
|---|---|---|---|---|
| | #good | #bad | #good | #bad |
| UFS | ~ 27.5% | ~ 72.5% | ~ 45% | ~ 55% |
| EXT | ~ 20% | ~ 80% | ~ 40% | ~ 60% |
| ZFS | ~ 98 % | ~ 2% | ~ 98 % | ~ 2% |

- Observation: RNG matters!
- Observation: Crashes happen during Mount, User-Emul. & Teardown!

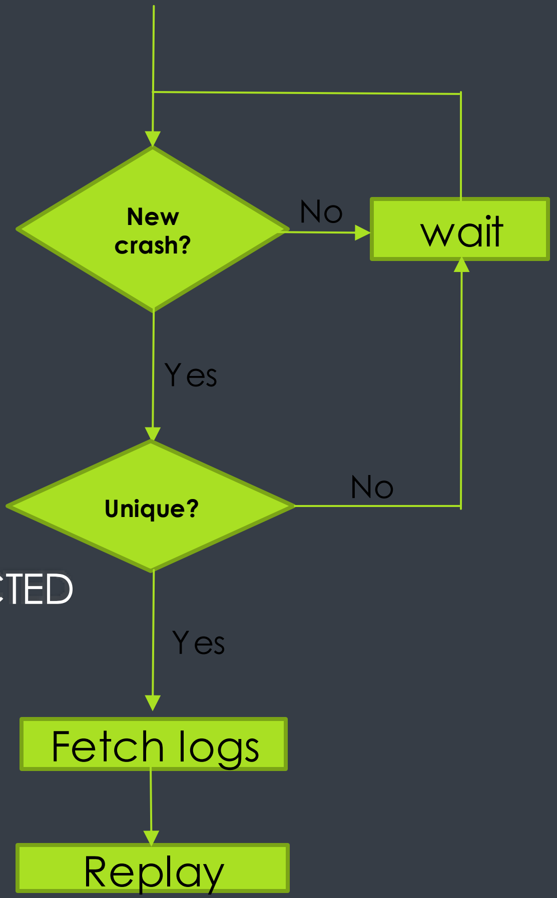
## 4. Monitoring

- Permanent alive checks for fuzzers
- Tracking of samples, mutations, seeds, crashes
- Logging of FS structure
- Logging of user emulation

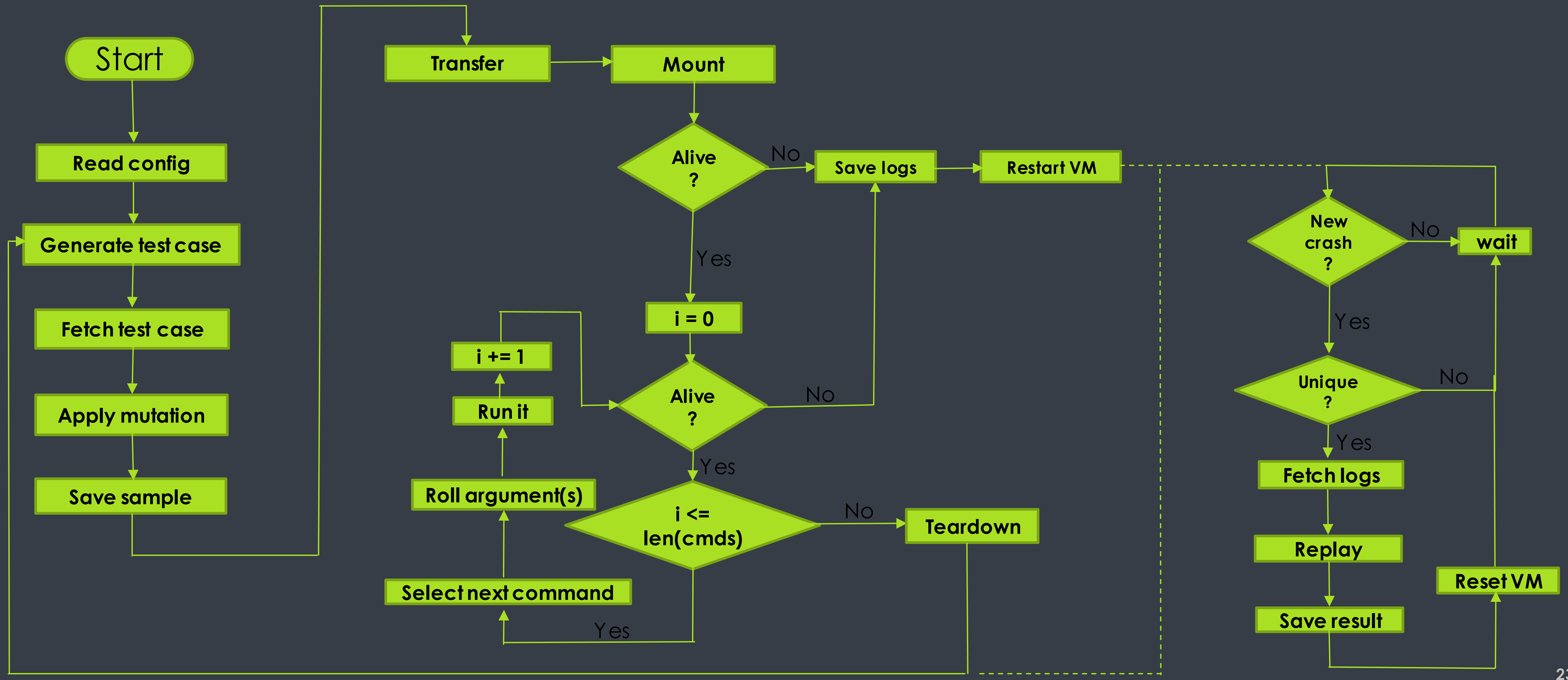
## 5. Verification

- Automatic continuous checks for new crashes
  - Based on hash identifier
- Replay on separate, 'always fresh' instance

Observation : OS side effects not as bad as expected

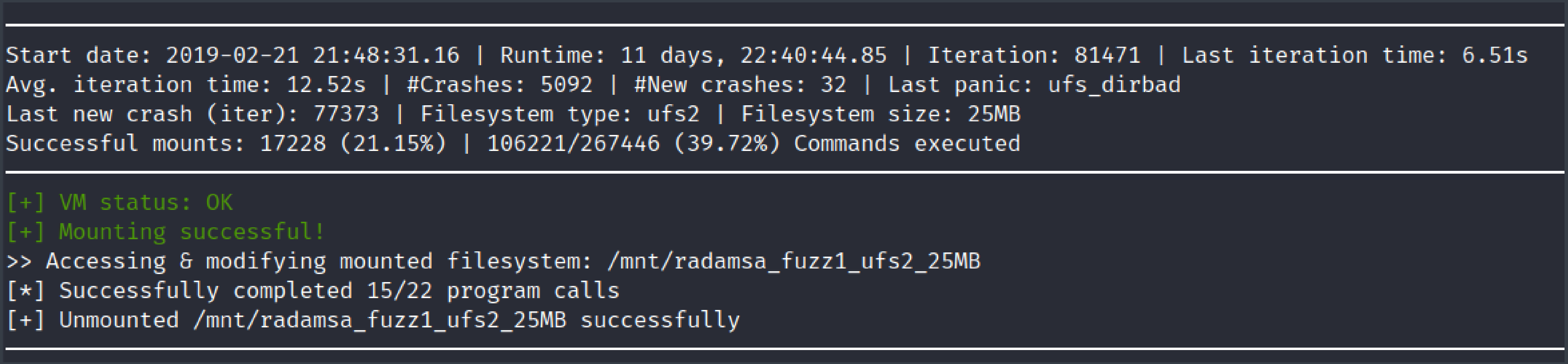


## PUTTING THINGS TOGETHER



## BUT CAN IT PRODUCE CRASHES?

## Putting things together



```
Start date: 2019-02-21 21:48:31.16 | Runtime: 11 days, 22:40:44.85 | Iteration: 81471 | Last iteration time: 6.51s
Avg. iteration time: 12.52s | #Crashes: 5092 | #New crashes: 32 | Last panic: ufs_dirbad
Last new crash (iter): 77373 | Filesystem type: ufs2 | Filesystem size: 25MB
Successful mounts: 17228 (21.15%) | 106221/267446 (39.72%) Commands executed

[+] VM status: OK
[+] Mounting successful!
>> Accessing & modifying mounted filesystem: /mnt/radamsa_fuzz1_ufs2_25MB
[*] Successfully completed 15/22 program calls
[+] Unmounted /mnt/radamsa_fuzz1_ufs2_25MB successfully
```

## Findings

- >100 unique crashes in UFS/EXT
  - Multiple OOB-R/OOB-w
  - Triple Fault in UFS
  - Double Fault in EXT
  - Bonus: Non-Deterministic crash in UFS with 6 unique core dumps so far
- Overall >82% reproducibility rate
  - Additionally another 5% produced a different crash on verification
- 17 syscalls covered

## SYSCALLS

- 26 USERLAND PROGRAMS
- 17 SYSTEM CALLS
  - 2 NEW VIA RANDOMIZING
  - 3 NEW VIA EXTENDED EMULATION

| | | | |
|---|---|---|---|
| sys_unmount | sys_linkat | sys_rmdir | sys_open_rwtc |
| sys_symlink | sys_access | sys_openat_rwtc | sys_rename |
| sys_mknodat | sys_unlink | sys_write | sys_mkdir |

| Category | Operation |
|---|---|
| changing geometry | chflags, chgrp, chmod, chown, mv, rm, truncate* |
| extending geometry | cp, dd, echo, ln, mkdir, mknod, split*, touch |
| parsing geometry | basename*, chdir, dirname*, du*, file, find, getfacl*, ls, readlink, stat, wc* |

## Syscalls

- 17 system calls
  - 2 new via randomizing
  - 3 new via extended emulation
- Observation: RNG matters!
- Observation: Fine tuning matters!

| | | | |
|---|---|---|---|
| sys_unmount | sys_linkat | sys_rmdir | sys_open_rwtc |
| sys_symlink | sys_access | sys_openat_rwtc | sys_rename |
| sys_mknodat | sys_unlink | sys_write | sys_mkdir |
| sys_nmount | sys_fstatat | sys_writev | sys_read |
| | sys_ftruncate | | |

## Resp.Disclosure - FreeBSD

- ~50/>100 disclosed via responsible disclosure
- A bunch of Mails later:
  - 21 confirmed bug tracker numbers
  - 10 confirmed fixes
  - However, no feedback/replies for months now.. 🤔

## Resp.Disclosure - Net-/OpenBSD

- Short evaluation in both of these show similar results (FFS/UFS).
  - NetBSD: "not interested"
    - #Fixes: 0 🤷
  - OpenBSD: "FFS/UFS filesystem has made these design decisions, Kernel has no logic to handle inconsistencies, …"
    - #Fixes: 0 🤷

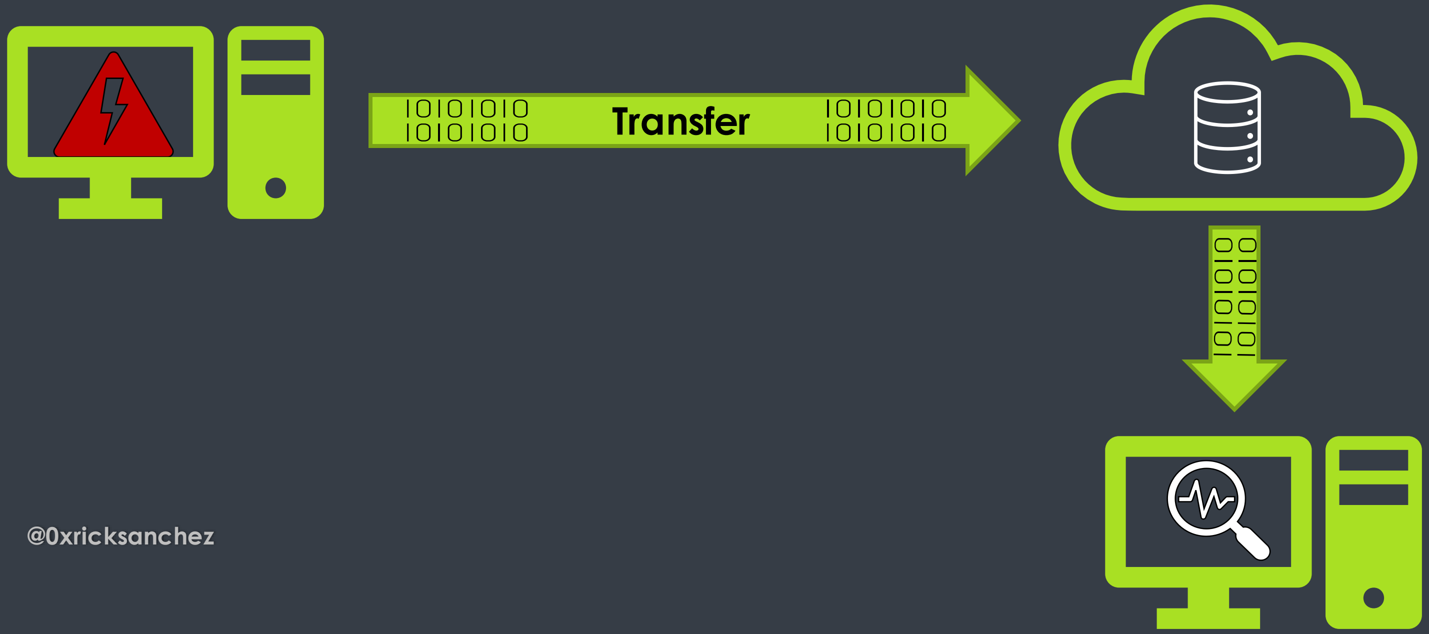
## CAVEATS

## Boot times

```
-> % cat result.txt
        | FAT     | FAT_DIAG          | SMALL_DIAG         | SMALL_DIAG_BOOT_DELAY  |
--------|---------|-------------------|--------------------|------------------------|
Run 1:  | 39.23s  | 41.97s (+7.0%)    | 37.79s (-10.0%)    | 26.10s (-38.2%)        |
Run 2:  | 39.77s  | 40.80s (+2.6%)    | 36.24s (-12.1%)    | 27.68s (-32.2%)        |
Run 3:  | 38.15s  | 40.79s (+6.9%)    | 37.26s (-8.7%)     | 27.11s (-34.5%)        |
Run 4:  | 39.12s  | 38.82s (-0.1%)    | 36.73s (-5.4%)     | 26.01s (-33.0%)        |
Run 5:  | 39.76s  | 41.45s (+4.3%)    | 36.71s (-11.5%)    | 25.58s (-38.3%)        |
====    |======   |============       |============        |==================      |
AVG:    | 39.21s  | 40.77s (+4%)      | 36.95s (-9.4%)     | 26,50s (-35.0%)        |
```

## ~~Boot times~~ man netdump

- netdump - protocol for transmitting kernel dumps to a remote server
    - Would eliminate need to reboot to fetch core details
    - However: unreliable in my setup



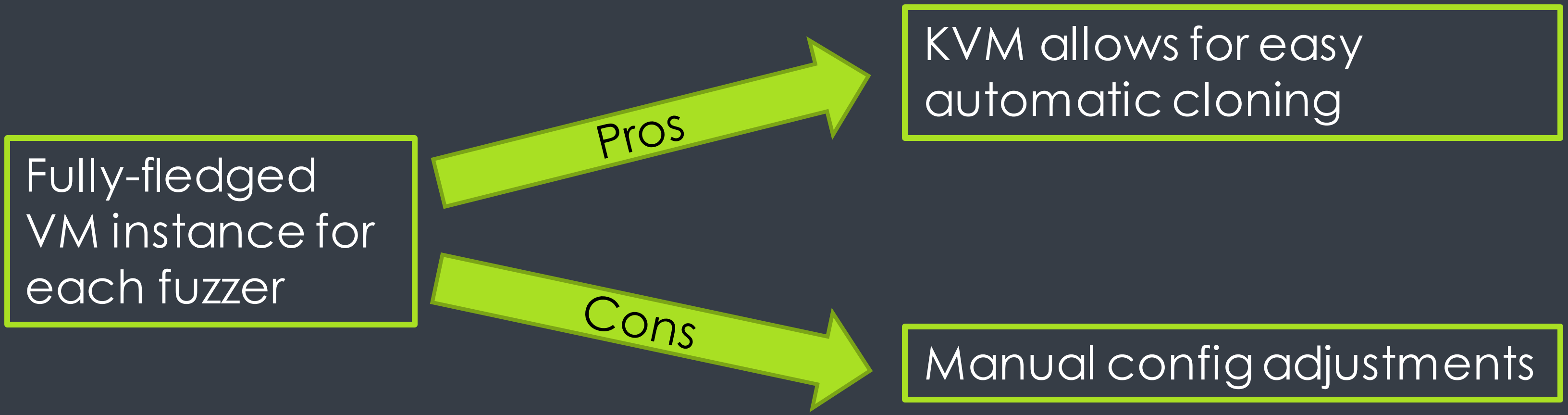
## ~~Boot times~~ LibOS

- Instead of full fletched KVM VM with '*optimized*' kernel
  - Only plug necessary parts together...

## Smart(er) Mutation?

- Right now:
  - No restoring or re-calculation of checksums/integrity checks
    - Important for EXT4, ZFS
  - Kernel feedback, KASAN
  - Automatic deduction of metadata field types/size

## Scalability?



Fully-fledged VM instance for each fuzzer

Pros

KVM allows for easy automatic cloning

Cons

Manual config adjustments

EOF

## Conclusion

- Write your own fuzzing tools!
  - Kernels still offer lots of bugs that wait to be uncovered
  - Modern FS implementations will need some more considerations
- Responsible Disclosure sometimes frustrating
- File systems allow for deep introspection of userland to kernel land behavior

## Fin.

- Questions/Suggesstions? Please reach out!
  - : https://twitter.com/0xricksanchez
  - : https://github.com/0xricksanchez
  - 0x00 : https://0x00sec.org/u/ricksanchez
  - : christopher.krah@fkie.fraunhofer.de
- Slides/Scripts?
  - Will be here: https://github.com/0xricksanchez/fs-fuzzer