

Finding Semantic Bugs in File Systems with an Extensible Fuzzing Framework

Seulbae Kim, Meng Xu^{*}, Sanidhya Kashyap^{*}, Jungyeon Yoon, Wen Xu, Taesoo Kim



Demonstration

Fuzzing F2FS in Linux v5.0-rc7 for **crash consistency**
Result at the end of the talk!

Question: Can file systems be bug-free?

Can file systems be bug-free?

- Code base is **massive**

 [torvalds / linux](#)

 Code

 Pull requests **300**

 Projects **0**

 Security

History for [linux / fs / ext4](#)

History for [linux / fs / btrfs](#)

History for [linux / fs / xfs](#)

Can file systems be bug-free? Not likely

- Code base is **massive**

torvalds / linux

<> Code

Pull requests 300

Projects 0

Security

39 KLoC

History for linux / fs / **ext4**

98 KLoC

History for linux / fs / **btrfs**

94 KLoC

History for linux / fs / **xfs**

+ common VFS layer (53 KLoC)!

Can file systems be bug-free? Not likely

- Code base is massive and **evolving**

torvalds / linux

<> Code Pull requests 300 Projects 0 Security

39 KLoC

History for linux / fs / **ext4**

Commits on Sep 29, 2019

Merge branch 'entropy' ...

 torvalds committed 23 days ago

Revert "Revert "ext4: make __ext4_..."

 torvalds committed 24 days ago

Commits on Sep 21, 2019

98 KLoC

History for linux / fs / **btrfs**

Commits on Oct 23, 2019

Merge tag 'for-5.4-rc4-tag' of git://git.k...

 torvalds committed 2 hours ago

Commits on Oct 17, 2019

Btrfs: check for the full sync flag while l...

 fdmanana authored and kdave committ

94 KLoC

History for linux / fs / **xfs**

Commits on Oct 15, 2019

xfs: change the seconds fields in xfs_bulks...

 djwong committed 8 days ago

Commits on Oct 9, 2019

xfs: move local to extent inode logging in

 Brian Foster authored and djwong committ

Can file systems be bug-free? Not likely

- Code base is massive and **evolving**

torvalds / linux

<> Code

Pull requests 300

Projects 0

Security

39 KLoC

History for linux / fs / **ext4**

98 KLoC

History for linux / fs / **btrfs**

94 KLoC

History for linux / fs / **xfs**

Commits on Sep 29, 2019

Commits on Oct 23, 2019

Commits on Oct 15, 2019

100+ ext4, Btrfs, XFS bugs were reported in 2019

Commits on Sep 21, 2019



fdmanana authored and kdave committ



Brian Foster authored and djwong commi

File system bugs are devastating

- Bugs and effects



Crash consistency bug



Data loss / corruption



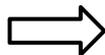
Specification violation



Unexpected runtime error



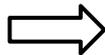
Logic bug



Incorrect result



Memory error



DoS / Privilege escalation

Previous approaches to find FS bugs

Regression Testing	Model Checking	Verified File System	Fuzzing
Linux Test Project xfstests fsck	FiSC (OSDI'04) eXplode (OSDI'06) Juxta (SOSP'15) Ferrite (ASPLOS'16) B3 (OSDI'18)	FSCQ (SOSP'15) Yggdrasil (OSDI'16) DFSCQ (SOSP'17) SFSCQ (OSDI'18)	Syzkaller (Google) kAFL (Security'17) Janus (S&P'19)

Previous approaches to find FS bugs

Regression Testing	Model Checking	Verified File System	Fuzzing
Linux Test Project xfstests fsck	FiSC (OSDI'04) eXplode (OSDI'06) Juxta (SOSP'15) Ferrite (ASPLOS'16) B3 (OSDI'18)	FSCQ (SOSP'15) Yggdrasil (OSDI'16) DFSCQ (SOSP'17) SFSCQ (OSDI'18)	Syzkaller (Google) kAFL (Security'17) Janus (S&P'19)
Only test known cases			

Previous approaches to find FS bugs

Regression Testing	Model Checking	Verified File System	Fuzzing
Linux Test Project xfstests fsck	FiSC (OSDI'04) eXplode (OSDI'06) Juxta (SOSP'15) Ferrite (ASPLOS'16) B3 (OSDI'18)	FSCQ (SOSP'15) Yggdrasil (OSDI'16) DFSCQ (SOSP'17) SFSCQ (OSDI'18)	Syzkaller (Google) kAFL (Security'17) Janus (S&P'19)
Only test known cases	High false positive Limited to known test cases		

Previous approaches to find FS bugs

Regression Testing	Model Checking	Verified File System	Fuzzing
Linux Test Project xfstests fsck	FiSC (OSDI'04) eXplode (OSDI'06) Juxta (SOSP'15) Ferrite (ASPLOS'16) B3 (OSDI'18)	FSCQ (SOSP'15) Yggdrasil (OSDI'16) DFSCQ (SOSP'17) SFSCQ (OSDI'18)	Syzkaller (Google) kAFL (Security'17) Janus (S&P'19)
Only test known cases	High false positive Limited to known test cases	Large unverified parts (buggy)	

Previous approaches to find FS bugs

Regression Testing	Model Checking	Verified File System	Fuzzing
Linux Test Project xfstests fsck	FiSC (OSDI'04) eXplode (OSDI'06) Juxta (SOSP'15) Ferrite (ASPLOS'16) B3 (OSDI'18)	FSCQ (SOSP'15) Yggdrasil (OSDI'16) DFSCQ (SOSP'17) SFSCQ (OSDI'18)	Syzkaller (Google) kAFL (Security'17) Janus (S&P'19)
Only test known cases	High false positive Limited to known test cases	Large unverified parts (buggy)	?

Our approach: Fuzzing file systems

- Feedback-driven fuzzing is a **complementary** solution
 - 😊 Produces effective test cases on-the-fly
 - 😊 Proven to be scalable in practice
- Known file system fuzzers
 - VM-based kernel fuzzers
 - kAFL (Security'17), Syzkaller (Google)
 - LibOS-based fuzzer
 - Janus (S&P'19) - our previous work!

Our approach: Fuzzing file systems

- Feedback-driven fuzzing is a **complementary** solution

**Janus discovered 90 memory-safety bugs
from file systems in 2018 😊**

- **KNOWN FILE SYSTEM FUZZERS**
 - VM-based kernel fuzzers
 - kAFL (Security'17), Syzkaller (Google)
 - LibOS-based fuzzer
 - Janus (S&P'19) - our previous work!

Our approach: Fuzzing file systems

- Feedback-driven fuzzing is a **complementary** solution

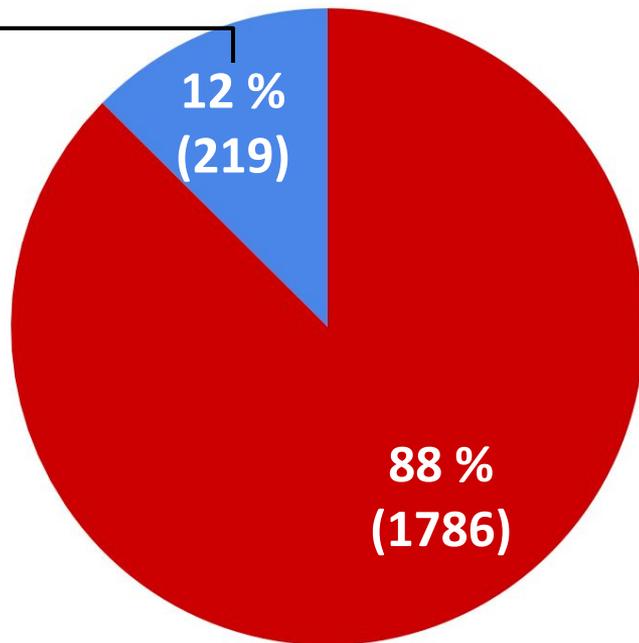
Janus discovered 90 memory-safety bugs from file systems in 2018 😊

- Known file system fuzzers
 - VM-based kernel fuzzers

However, existing file system fuzzers focus only on memory-safety bugs 😞

File system bugs in various flavors

- **Memory-safety bugs**
(focus of existing fuzzers)

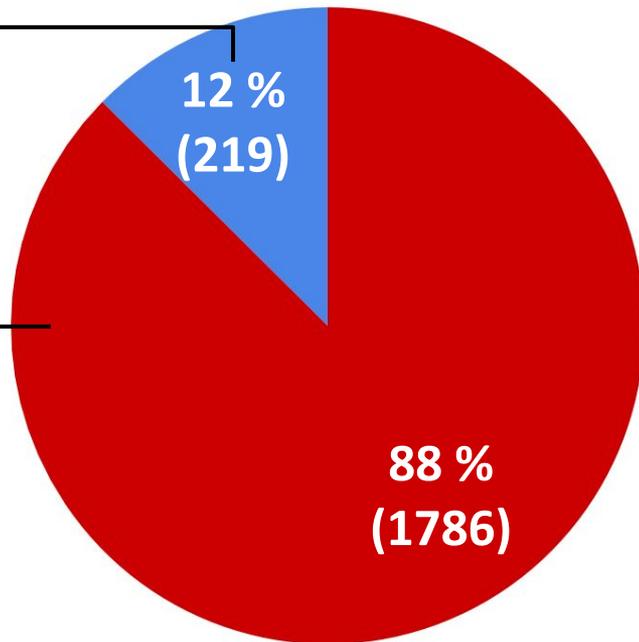


*Reference: Lu, Lanyue, et al. "A study of Linux file system evolution."
FAST'13

File system bugs in various flavors

- **Memory-safety bugs** (focus of existing fuzzers)

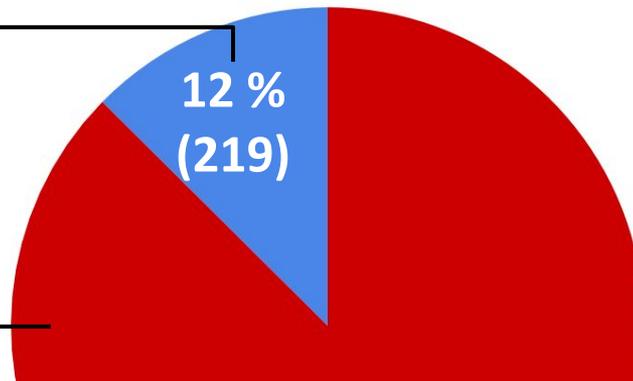
- **Semantic bugs**
 - Crash consistency bug
 - Specification violation
 - Logic bug
 - ...



*Reference: Lu, Lanyue, et al. "A study of Linux file system evolution."
FAST'13

File system bugs in various flavors

- Memory-safety bugs
(focus of existing fuzzers)



- Semantic bugs

**We'd like to take advantage of fuzzing
for finding **semantic bugs****

*Reference: Lu, Lanyue, et al. "A study of Linux file system evolution."

FAST'13

Challenge: Semantic bugs are harder to detect

- Key idea in fuzzing: “Crashes” are feedback to fuzzers

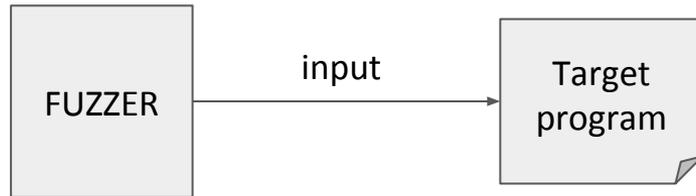
Fuzzing for memory-safety bugs



Challenge: Semantic bugs are harder to detect

- Key idea in fuzzing: “Crashes” are feedback to fuzzers

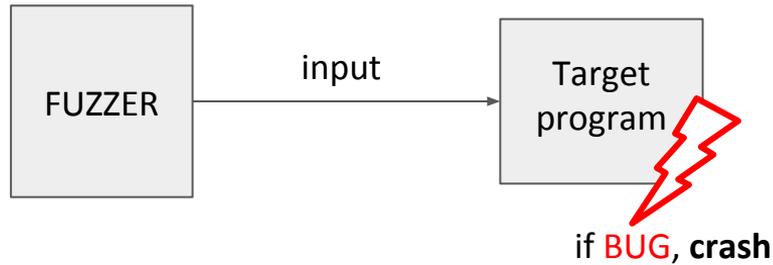
Fuzzing for memory-safety bugs



Challenge: Semantic bugs are harder to detect

- Key idea in fuzzing: “Crashes” are feedback to fuzzers

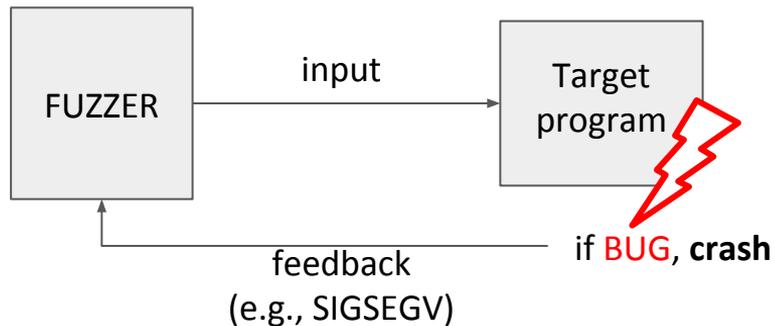
Fuzzing for memory-safety bugs



Challenge: Semantic bugs are harder to detect

- Key idea in fuzzing: “Crashes” are feedback to fuzzers

Fuzzing for memory-safety bugs

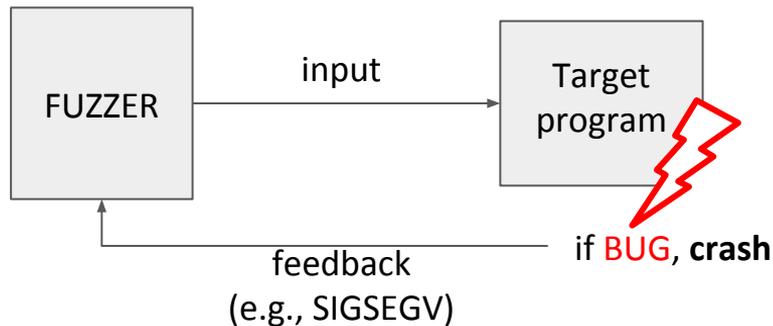


Detected!

Challenge: Semantic bugs are harder to detect

- Problem: Semantic bugs **fail SILENTLY** (i.e., no feedback)

Fuzzing for memory-safety bugs



Detected!

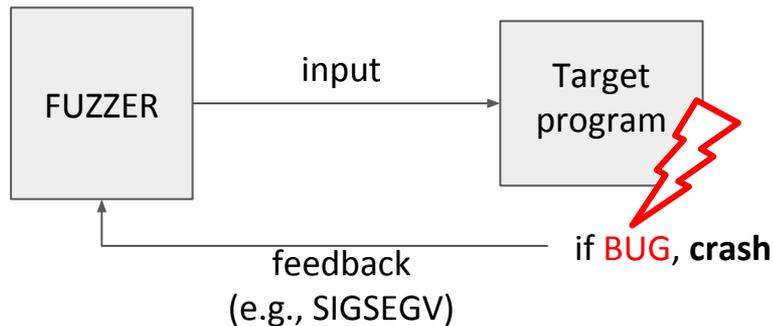
Fuzzing for semantic bugs
(e.g., spec. violation)



Challenge: Semantic bugs are harder to detect

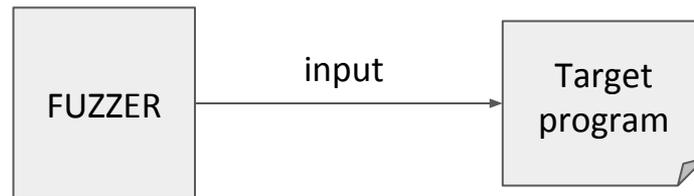
- Problem: Semantic bugs **fail SILENTLY** (i.e., no feedback)

Fuzzing for memory-safety bugs



Detected!

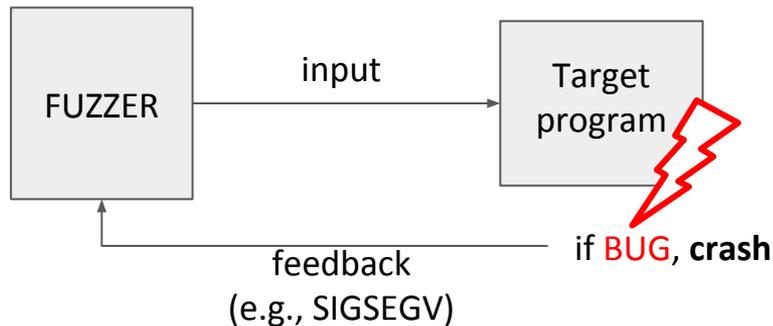
Fuzzing for semantic bugs
(e.g., spec. violation)



Challenge: Semantic bugs are harder to detect

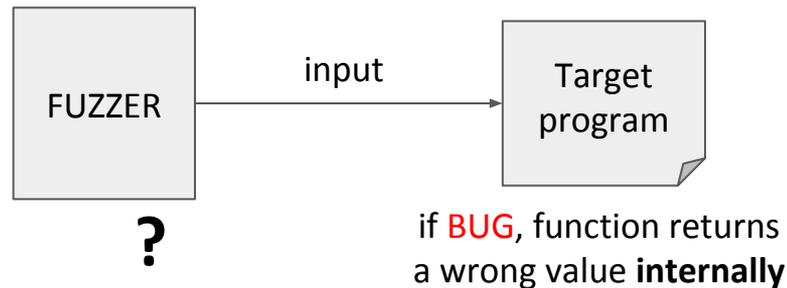
- Problem: Semantic bugs **fail SILENTLY** (i.e., no feedback)

Fuzzing for memory-safety bugs



Detected!

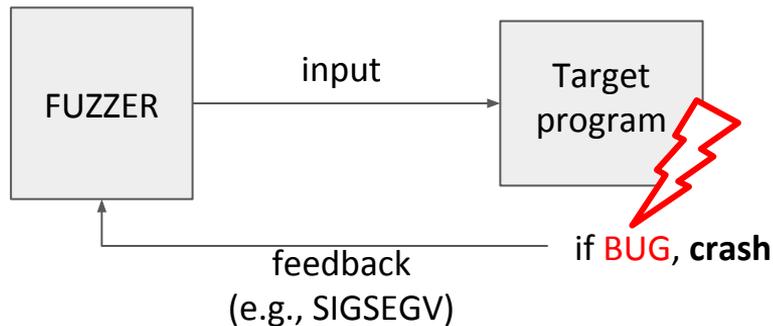
Fuzzing for semantic bugs
(e.g., spec. violation)



Challenge: Semantic bugs are harder to detect

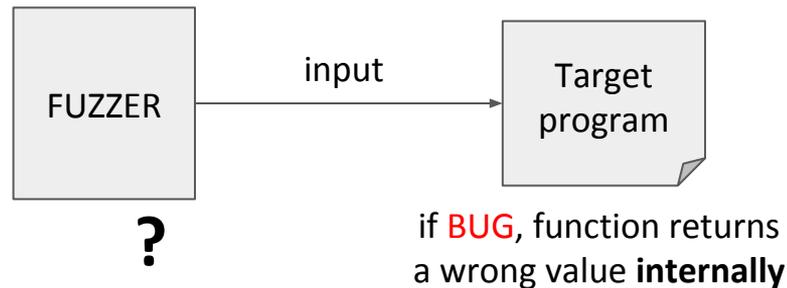
- Problem: Semantic bugs **fail SILENTLY** (i.e., no feedback)

Fuzzing for memory-safety bugs



Detected!

Fuzzing for semantic bugs
(e.g., spec. violation)

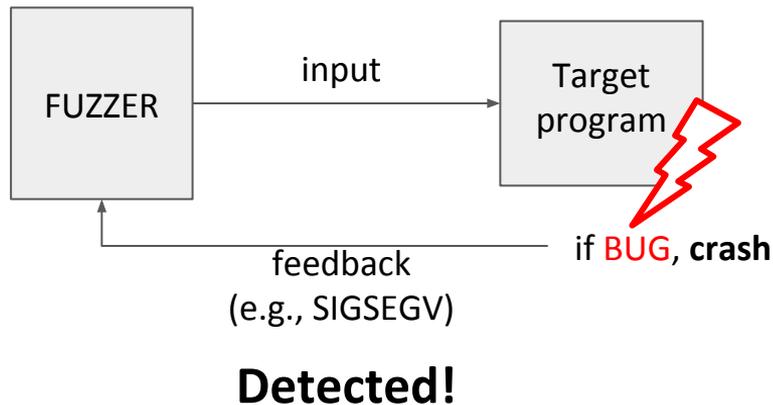


Not detected 😞

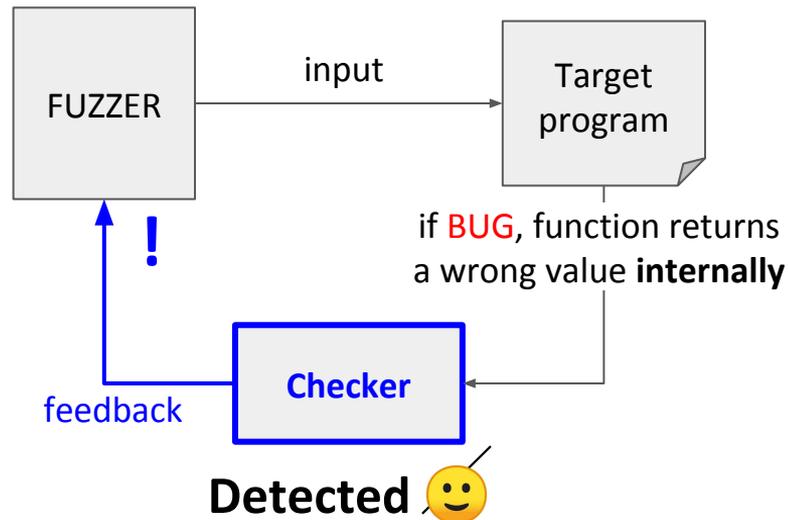
Challenge: Semantic bugs are harder to detect

- Problem: Semantic bugs **fail SILENTLY** (i.e., no feedback)

Fuzzing for memory-safety bugs



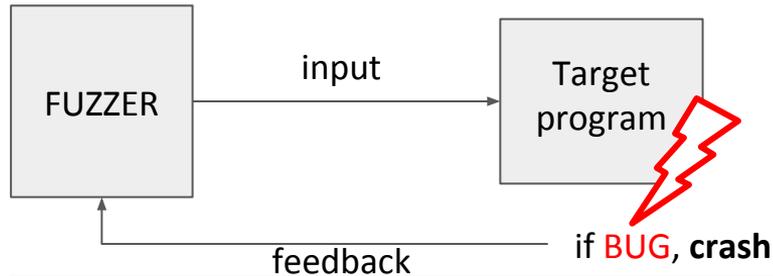
Fuzzing for semantic bugs (e.g., spec. violation)



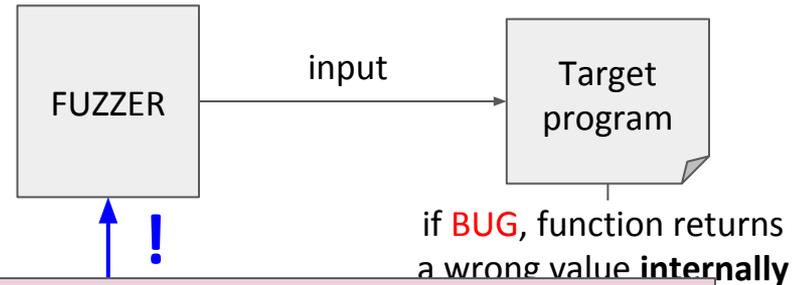
Challenge: Semantic bugs are harder to detect

- Problem: Semantic bugs **fail SILENTLY** (i.e., no feedback)

Fuzzing for memory-safety bugs



Fuzzing for semantic bugs
(e.g., spec. violation)

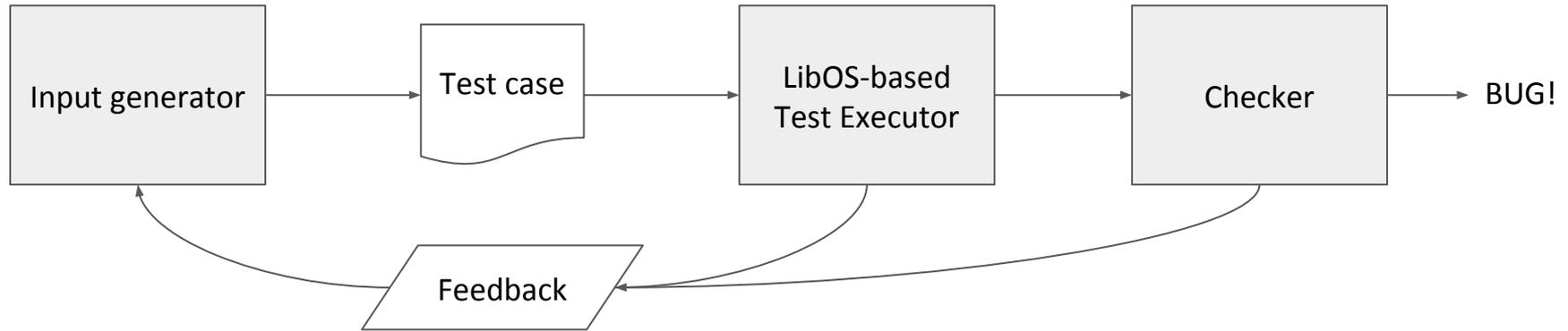


Accurate **checker** for each **bug type**
needs to be integrated to fuzzing!

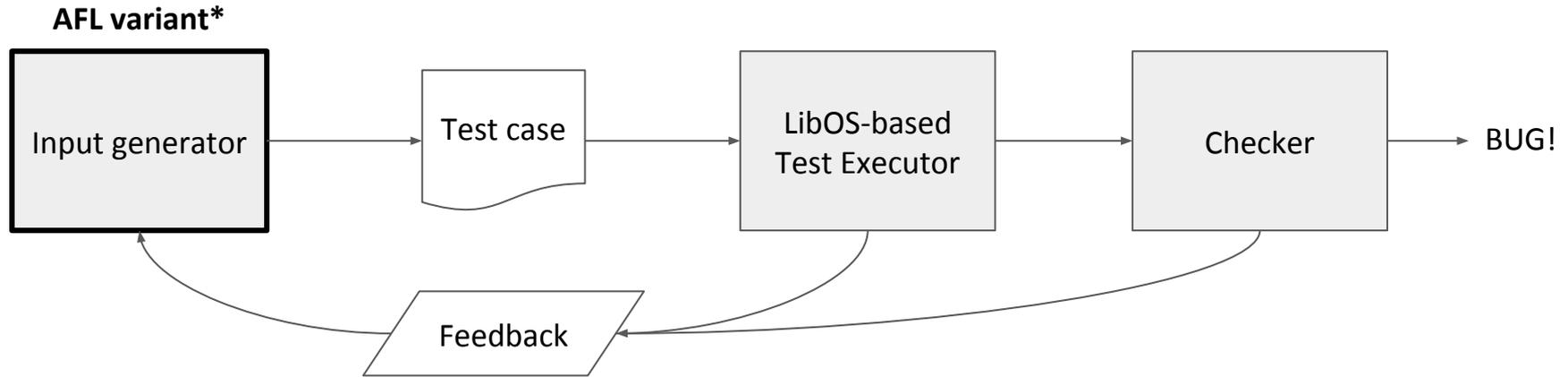
Proposed solution: Hydra

A turnkey solution for
file system fuzzing

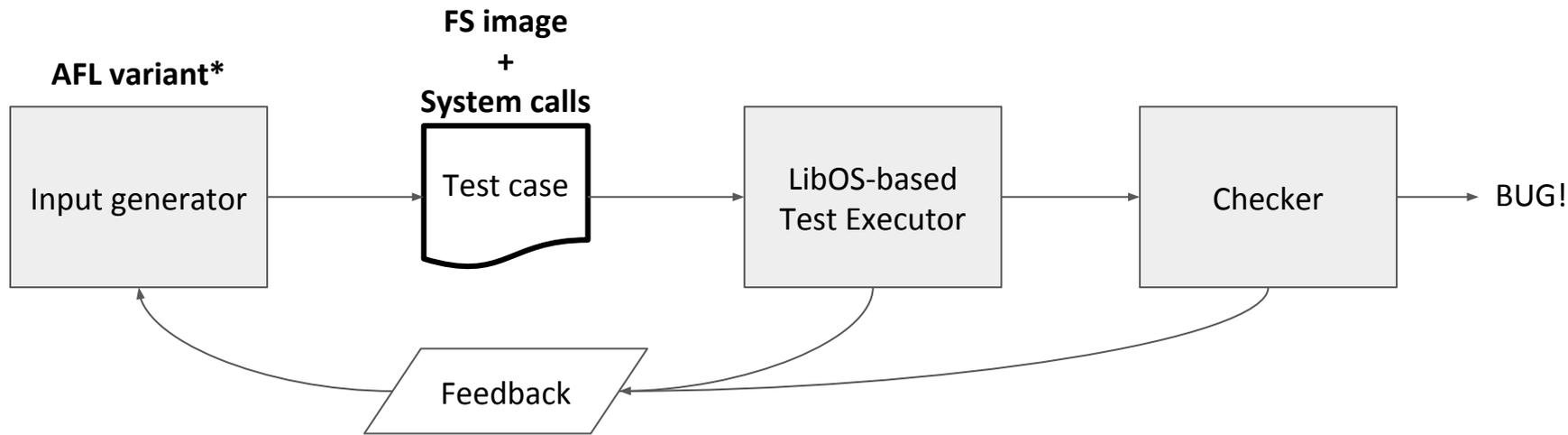
HYDRA overview (high-level)



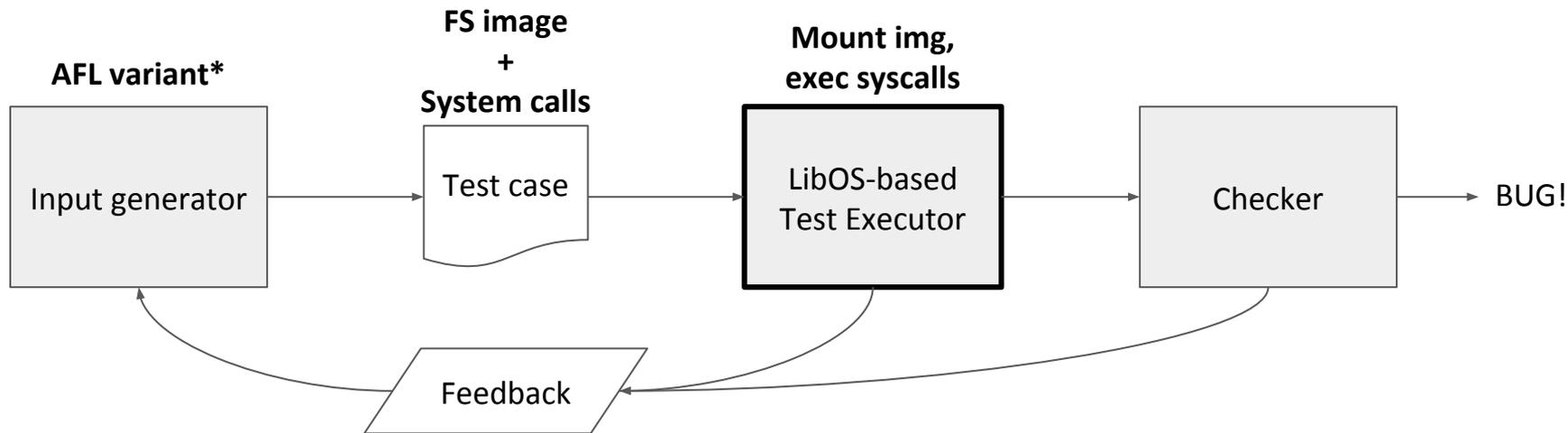
HYDRA overview - Input generator



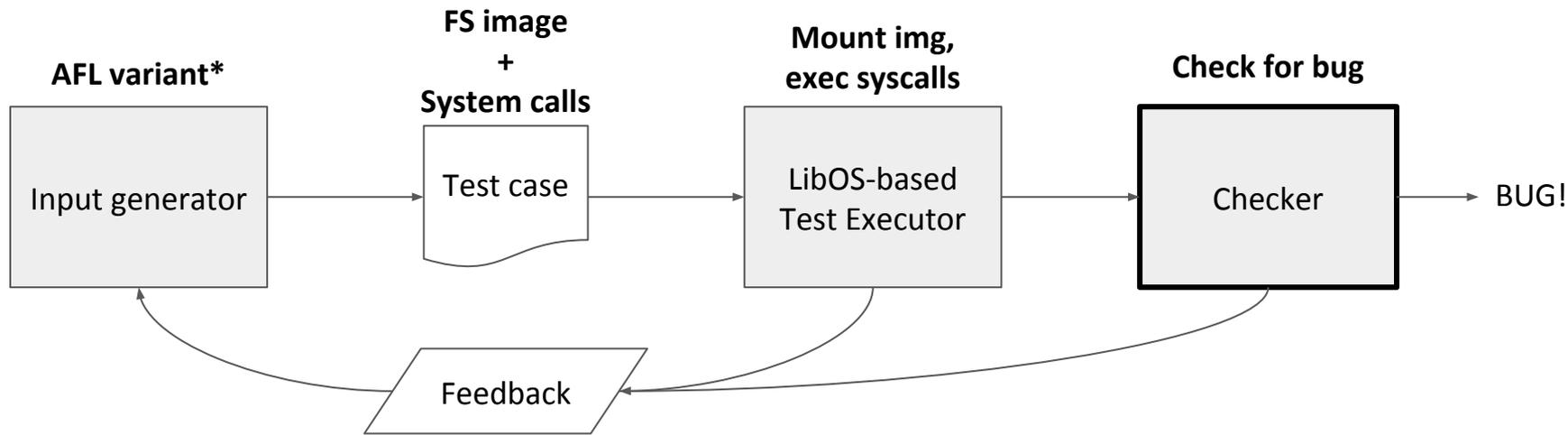
HYDRA overview - Test case



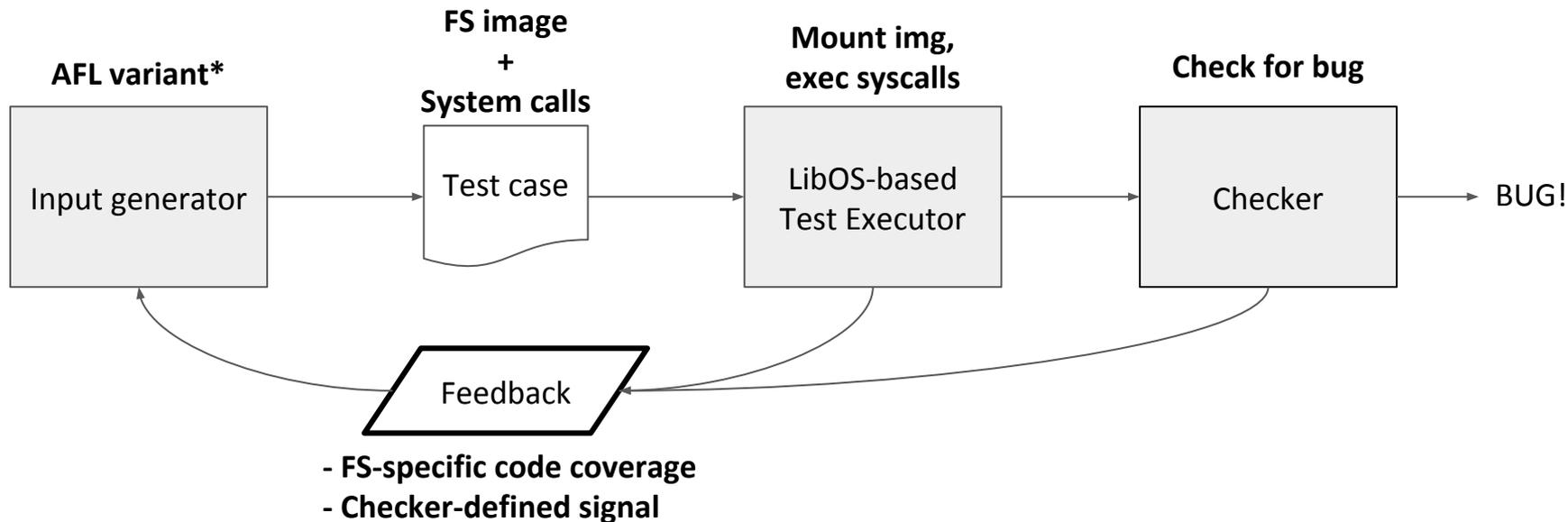
HYDRA overview - LibOS-based test executor



HYDRA overview - Checker

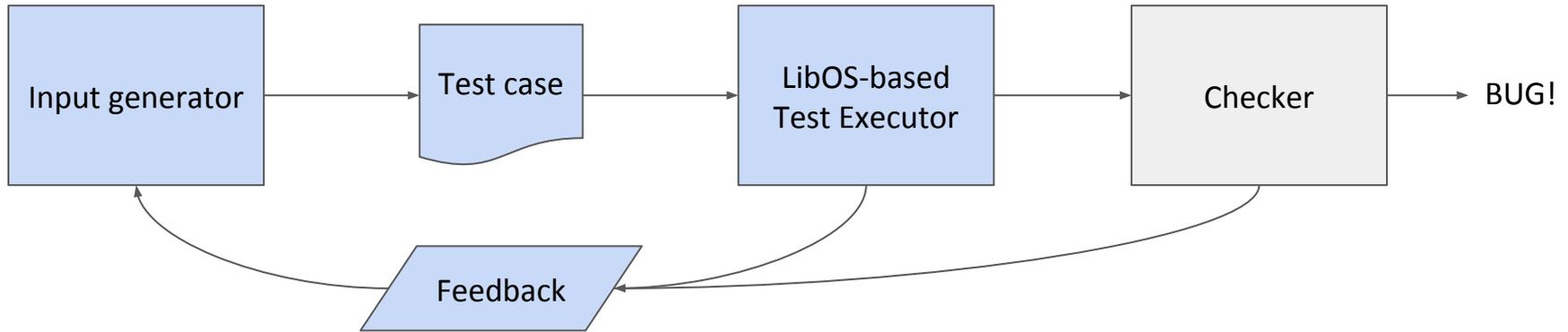


HYDRA overview - Feedback



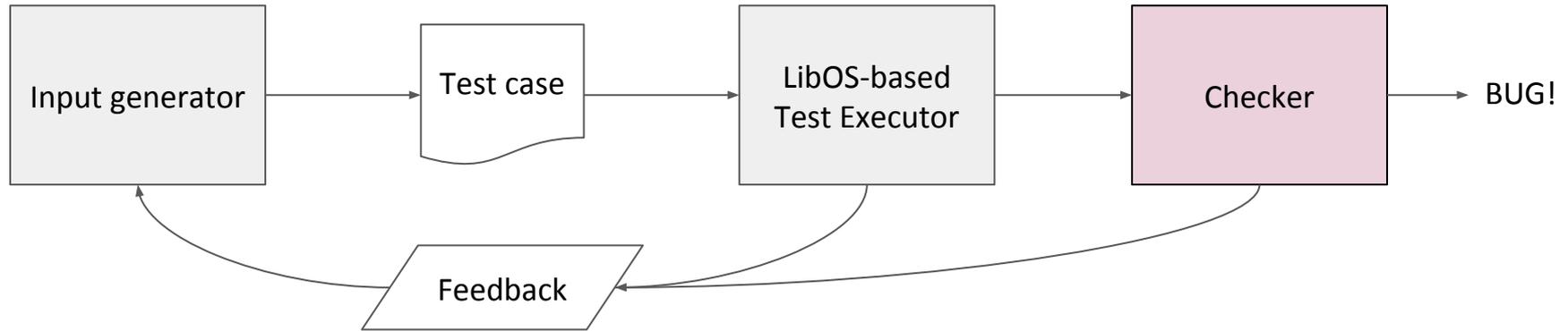
Hydra framework takes care of

- Automated input space exploration
 - Test execution
 - Incorporation of checkers, ...
- Develop and plug-in a bug checker



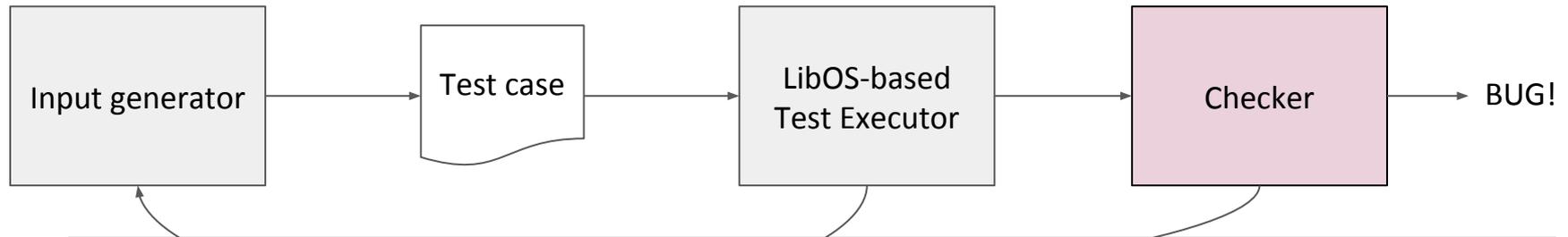
In the meantime.. **a tester** can

- Automated input space exploration
 - Test execution
 - Incorporation of checkers, ...
- **Develop and plug-in a specialized bug checker**



Separation of concern!

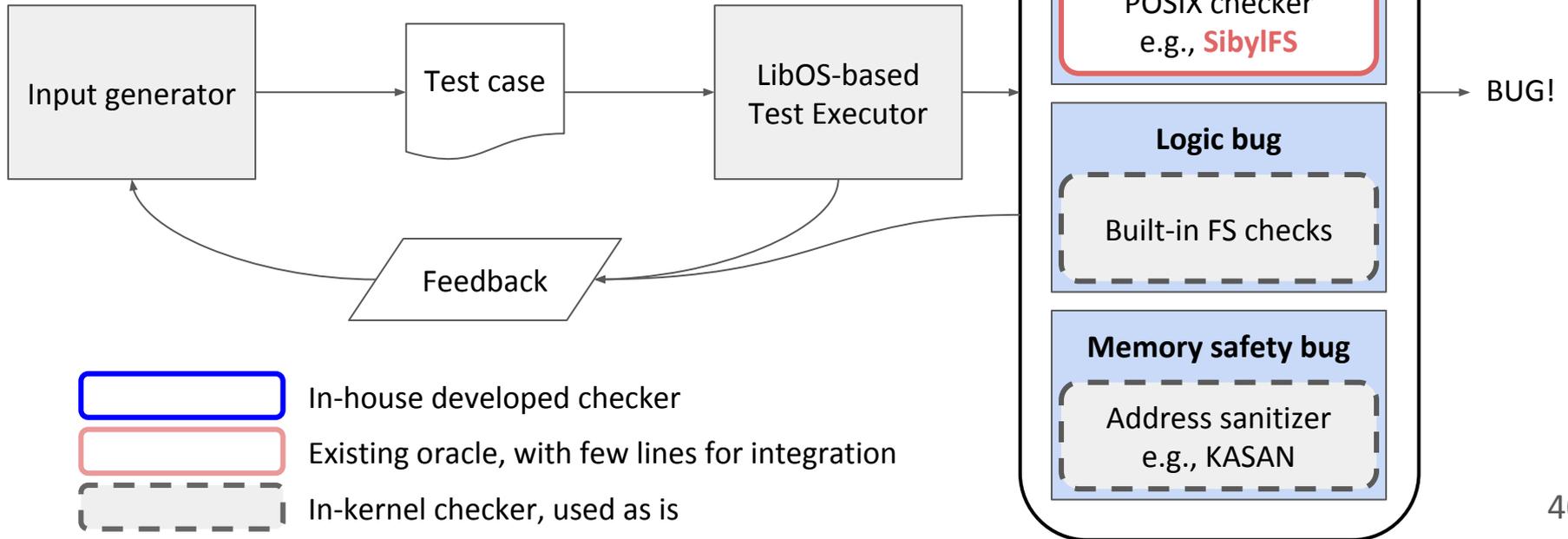
- Automated input space exploration
 - Test execution
 - Incorporation of checkers, ...
- **Develop and plug-in a specialized bug checker**



Developers may focus solely on **describing the bugs of their own interests**

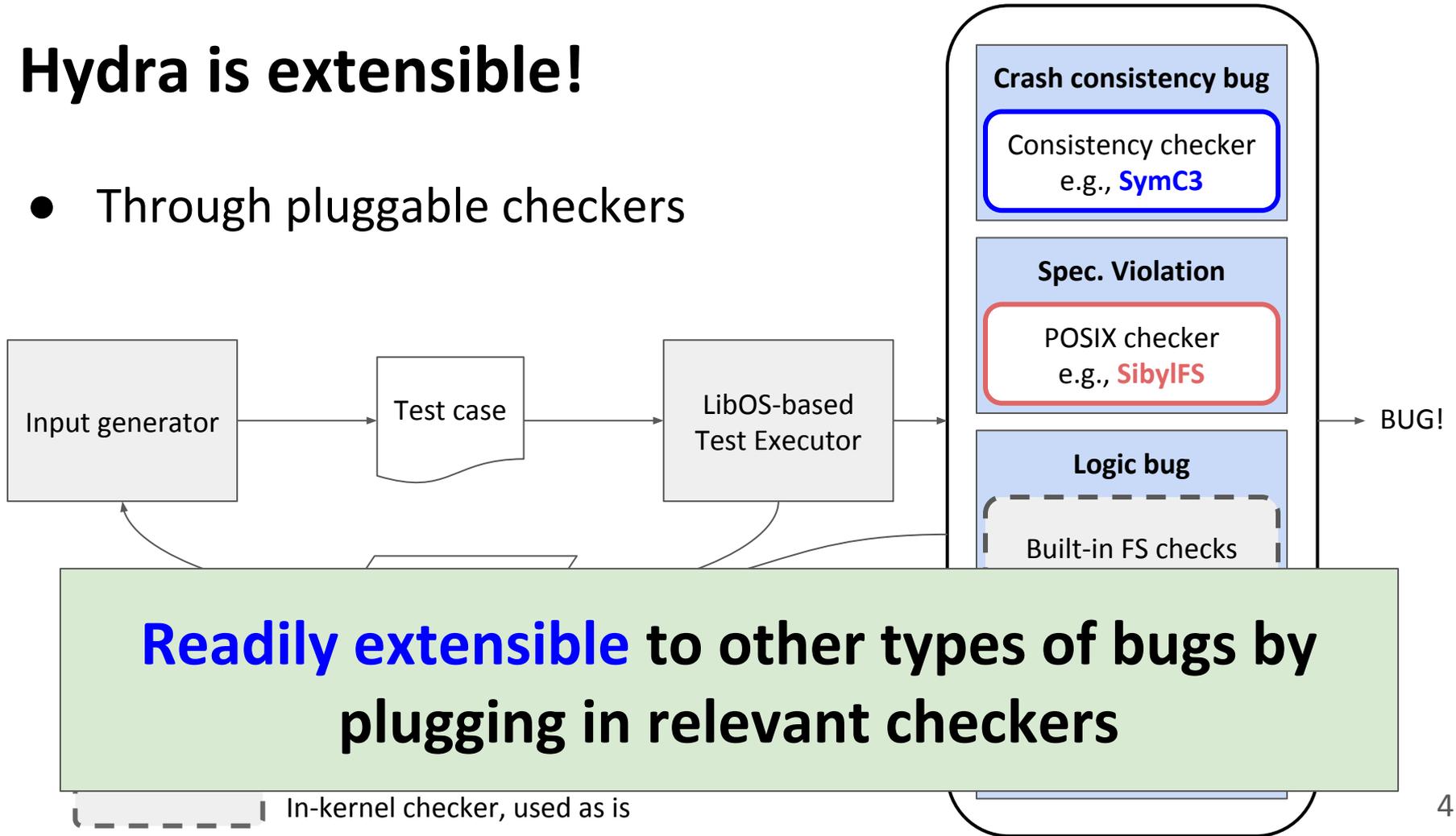
Hydra is extensible!

- Through pluggable checkers



Hydra is extensible!

- Through pluggable checkers



Hydra in action

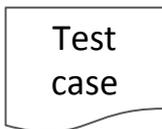
Finding crash consistency bug
utilizing SymC3 checker with Hydra

Hydra in action - Crash consistency testing

- SymC3: Symbolically evaluate crashing states
(i.e., keeping in-memory and on-disk states, like real FS implementation)
 - Input : a list of system calls, initial state
 - Output: a list of legitimate post-crash states

Hydra in action - Crash consistency testing

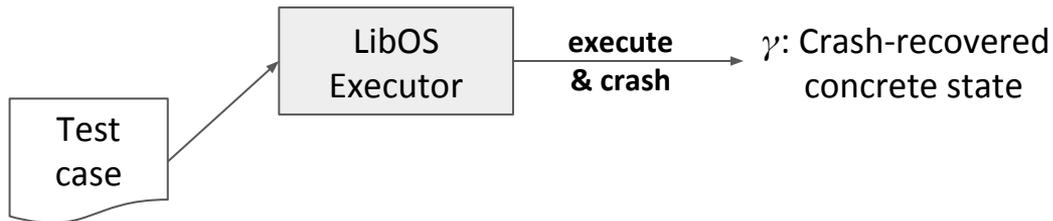
- SymC3: Symbolically evaluate crashing states
(i.e., keeping in-memory and on-disk states, like real FS implementation)
 - Input : a list of system calls, initial state
 - Output: a list of legitimate post-crash states
- Checking errors:



Test
case

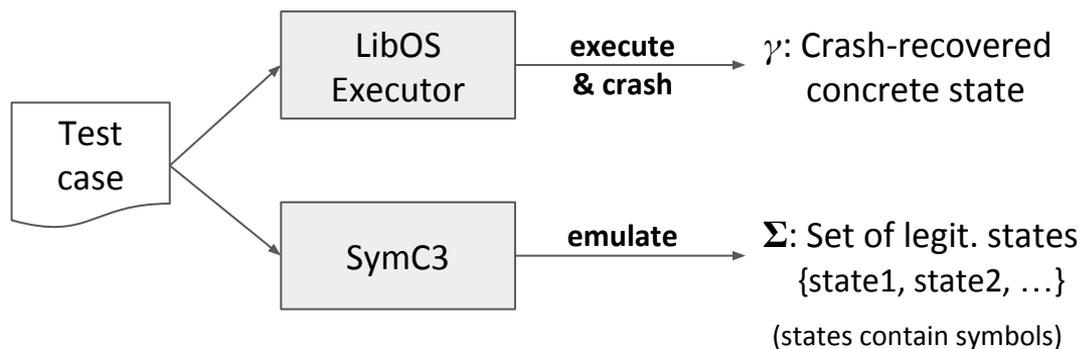
Hydra in action - Crash consistency testing

- SymC3: Symbolically evaluate crashing states
(i.e., keeping in-memory and on-disk states, like real FS implementation)
 - Input : a list of system calls, initial state
 - Output: a list of legitimate post-crash states
- Checking errors:



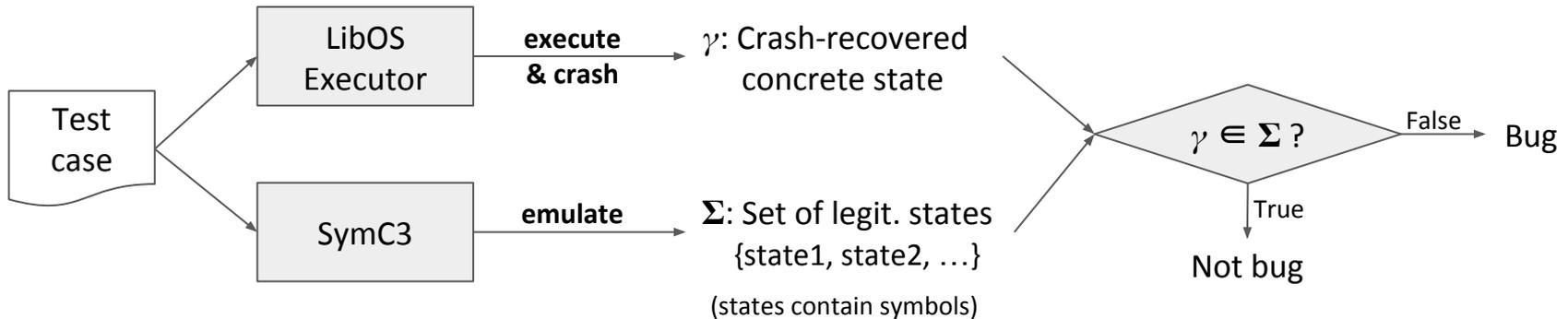
Hydra in action - Crash consistency testing

- SymC3: Symbolically evaluate crashing states
(i.e., keeping in-memory and on-disk states, like real FS implementation)
 - Input : a list of system calls, initial state
 - Output: a list of legitimate post-crash states
- Checking errors:



Hydra in action - Crash consistency testing

- SymC3: Symbolically evaluate crashing states
(i.e., keeping in-memory and on-disk states, like real FS implementation)
 - Input : a list of system calls, initial state
 - Output: a list of legitimate post-crash states
- Checking errors:



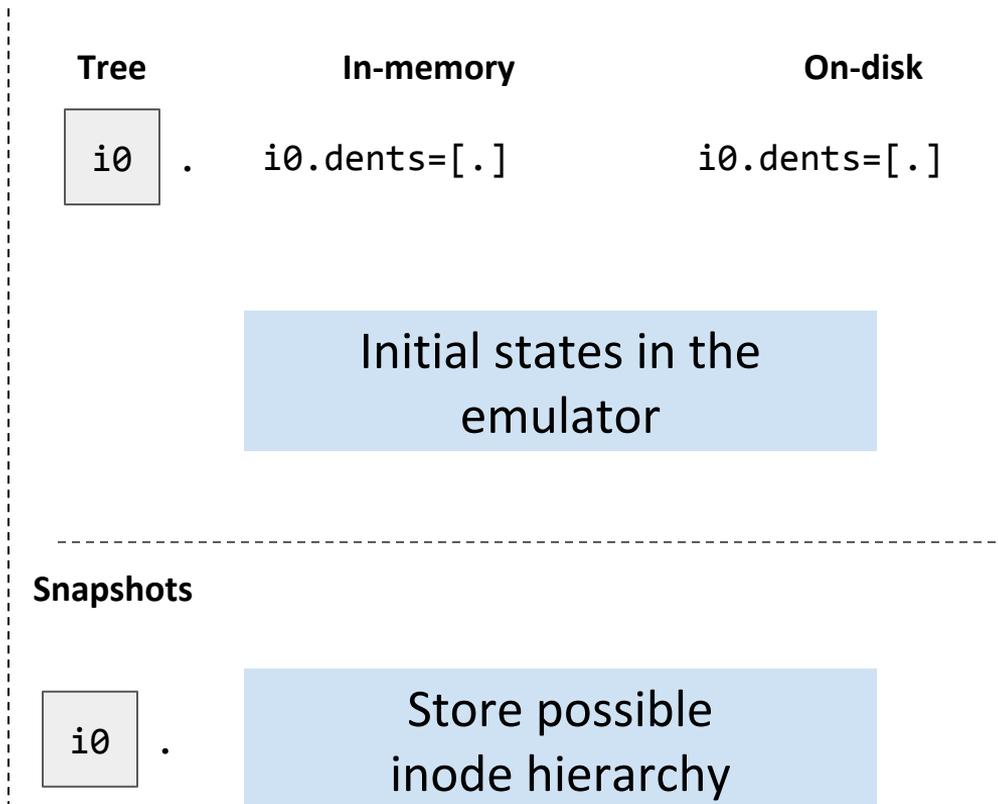
Hydra in action - Fuzzer-generated test case

- Simplest test case (but it was a **real bug** in F2FS!)

```
1  mkdir "A" 0775
2  sync
3  chmod "A" 0600
4  fsync "A"
```

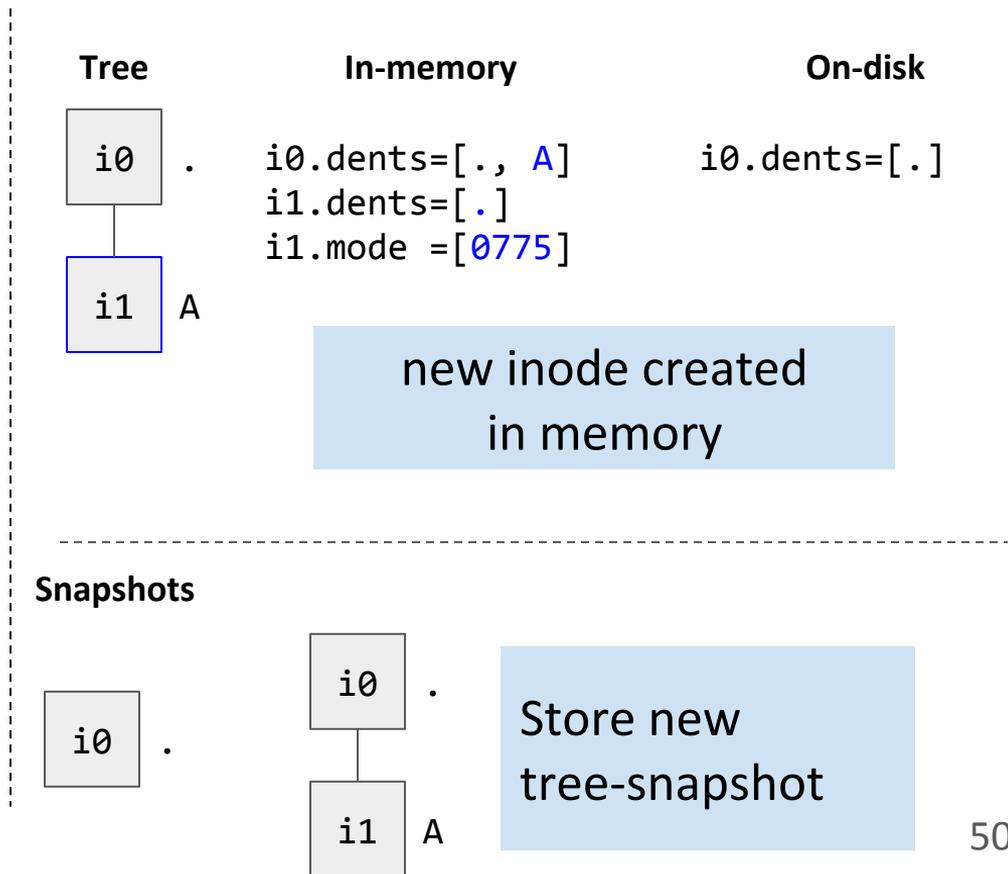
Hydra in action - Initial emulator states

```
1 mkdir "A" 0775
2 sync
3 chmod "A" 0600
4 fsync "A"
```



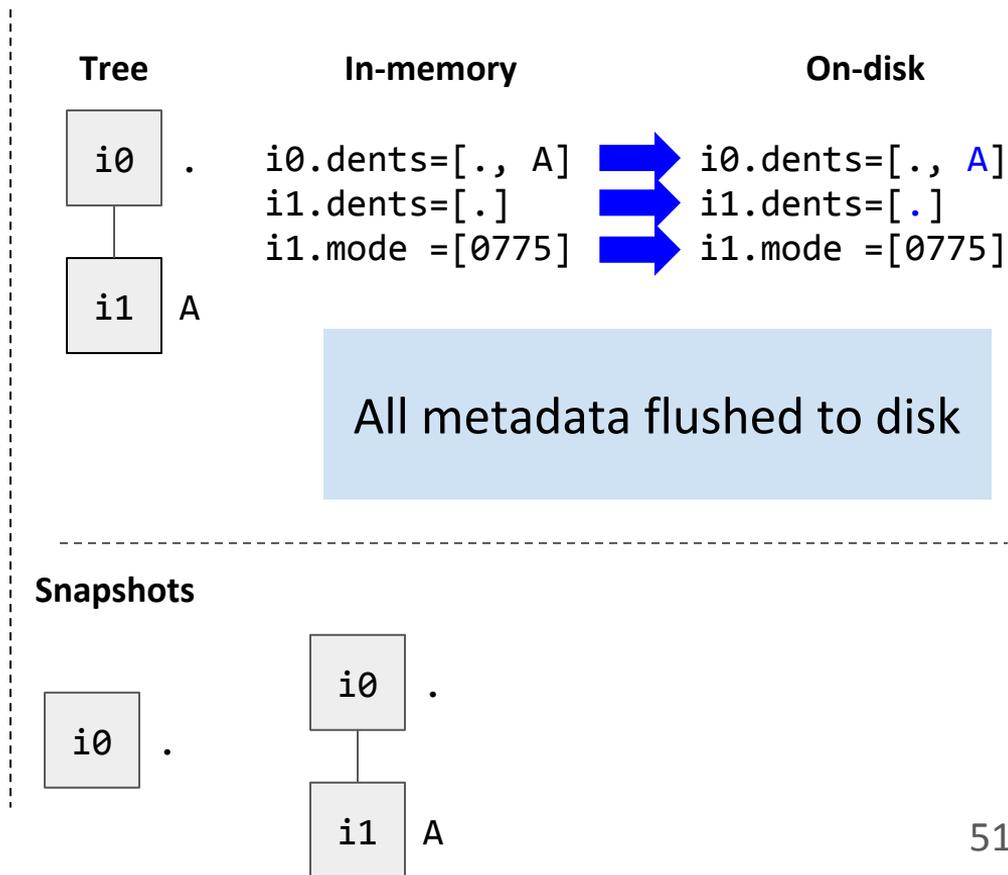
Hydra in action - Emulation of test case

```
1 mkdir "A" 0775
2 sync
3 chmod "A" 0600
4 fsync "A"
```



Hydra in action - Emulation of test case

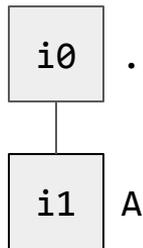
```
1 mkdir "A" 0775
2 sync
3 chmod "A" 0600
4 fsync "A"
```



Hydra in action - Emulation of test case

```
1 mkdir "A" 0775
2 sync
3 chmod "A" 0600
4 fsync "A"
```

Tree



In-memory

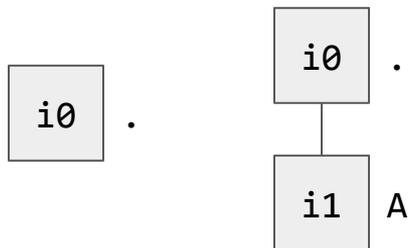
```
i0.dents=[., A]
i1.dents=[.]
i1.mode = [0775, 0600]
```

On-disk

```
i0.dents=[., A]
i1.dents=[.]
i1.mode = [0775]
```

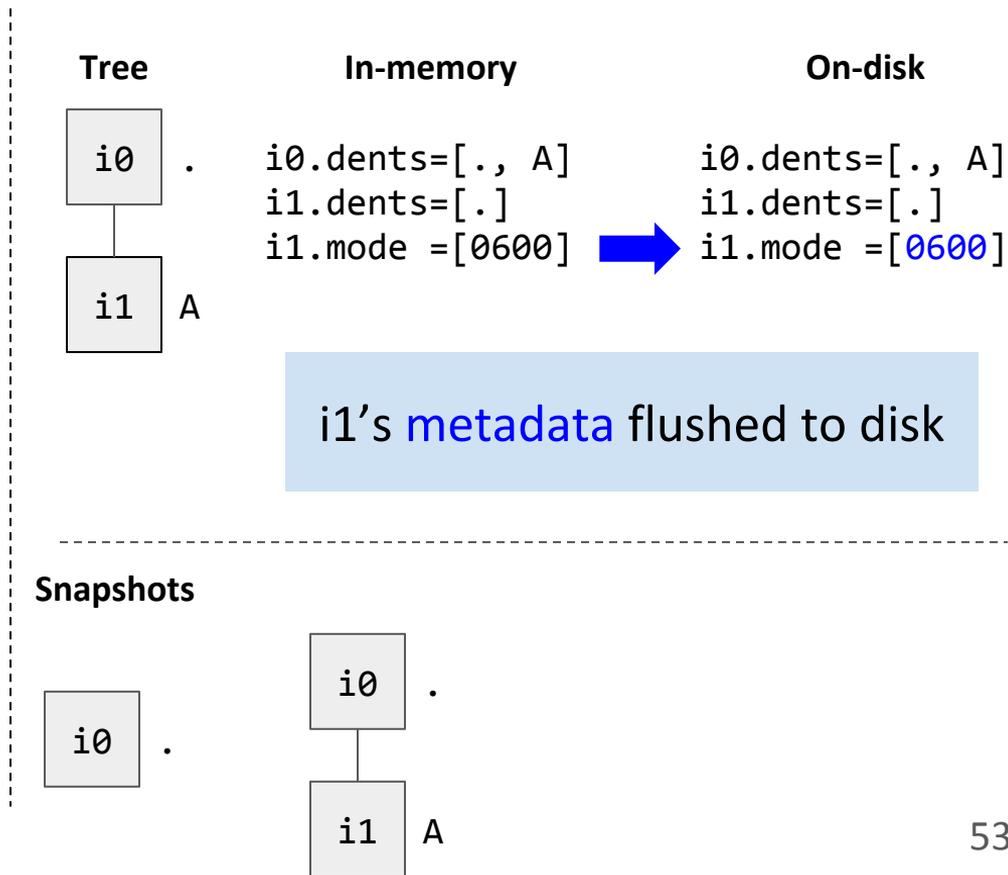
New metadata is written
History of metadata changes
is maintained

Snapshots



Hydra in action - Emulation of test case

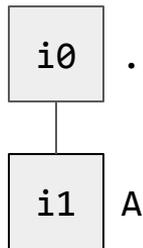
```
1 mkdir "A" 0775
2 sync
3 chmod "A" 0600
4 fsync "A"
```



Hydra in action - End of test case emulation

```
1 mkdir "A" 0775
2 sync
3 chmod "A" 0600
4 fsync "A"
```

Tree



In-memory

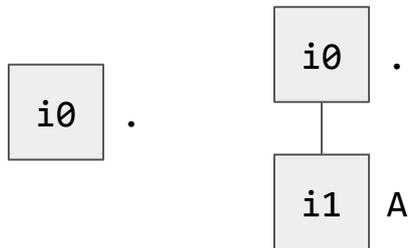
```
i0.dents=[., A]
i1.dents=[.]
i1.mode =[0600]
```

On-disk

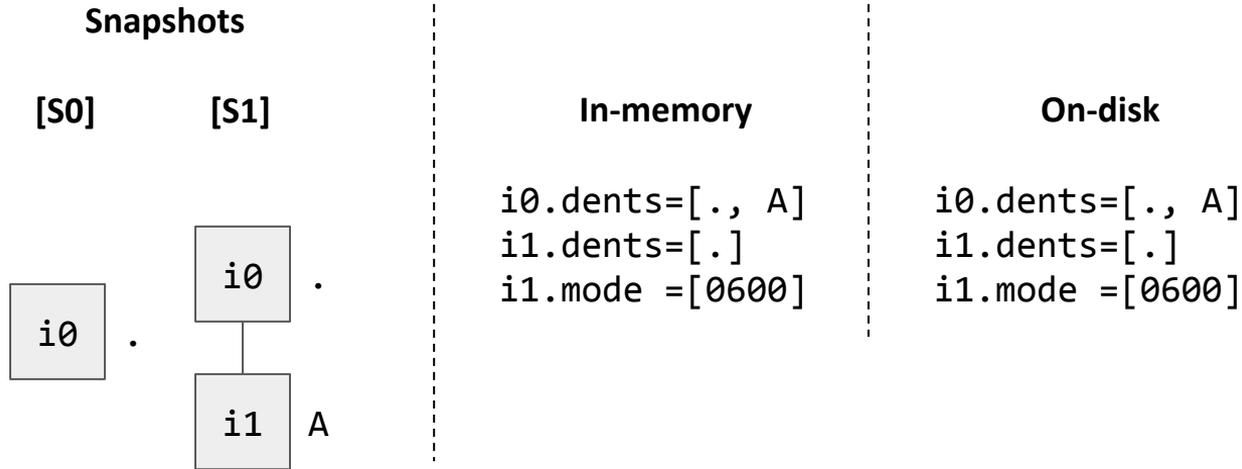
```
i0.dents=[., A]
i1.dents=[.]
i1.mode =[0600]
```

Enumerate legitimate post-crash states

Snapshots

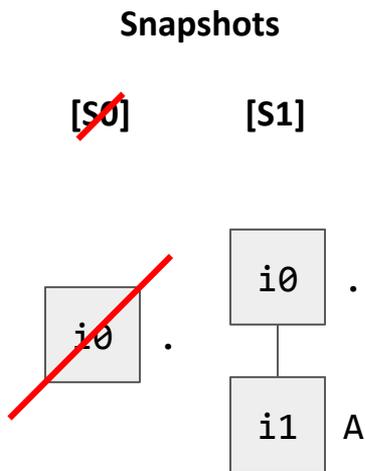


Hydra in action - Enumerating legitimate states



Hydra in action - Enumerating legitimate states

1. Check validity of snapshots



Drop **S0** (i1 is persisted)

In-memory

```
i0.dents=[., A]
i1.dents=[.]
i1.mode =[0600]
```

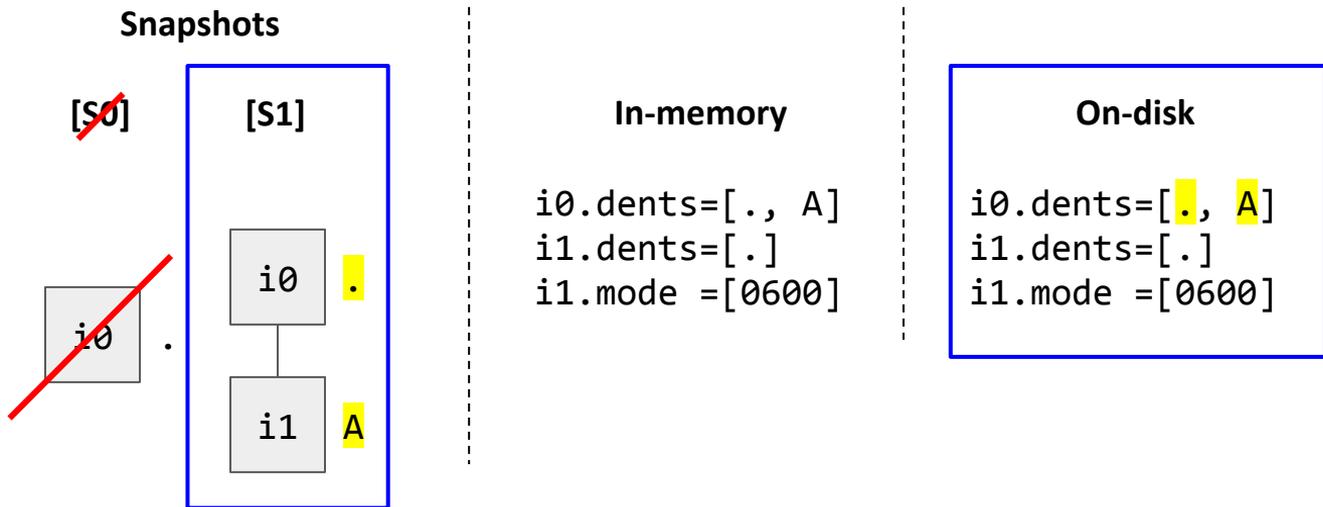
On-disk

```
i0.dents=[., A]
i1.dents=[.]
i1.mode =[0600]
```

./A must exist!

Hydra in action - Enumerating legitimate states

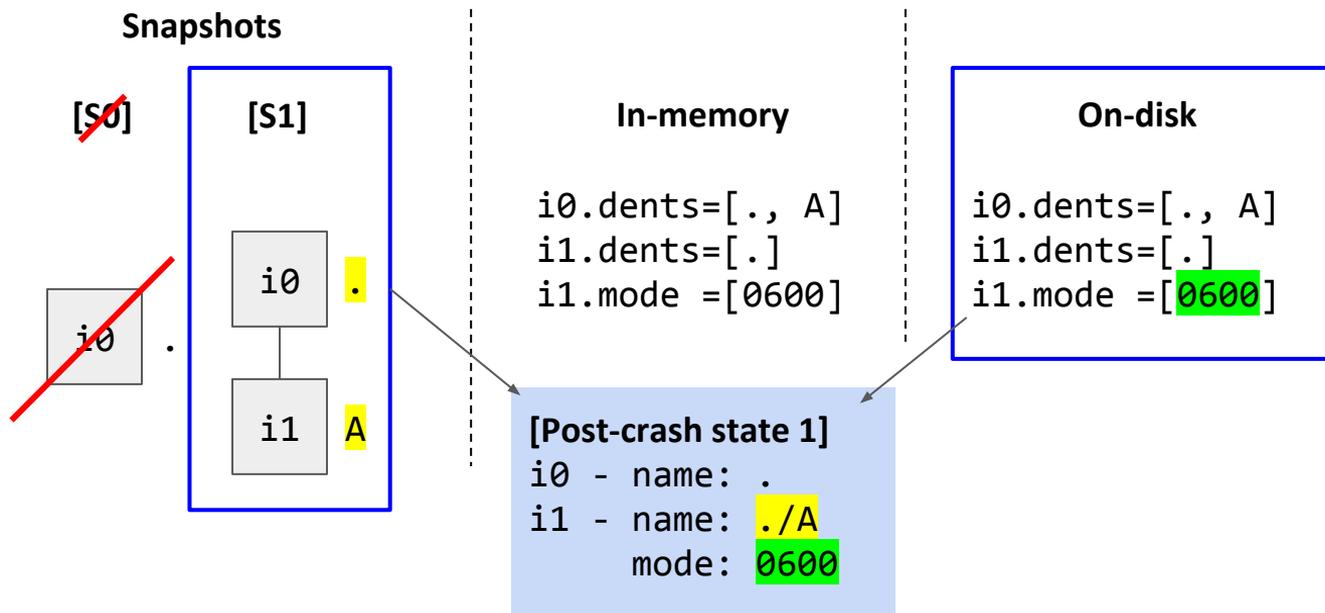
1. Check validity of snapshots



`S1` is valid
(does not violate persisted state)

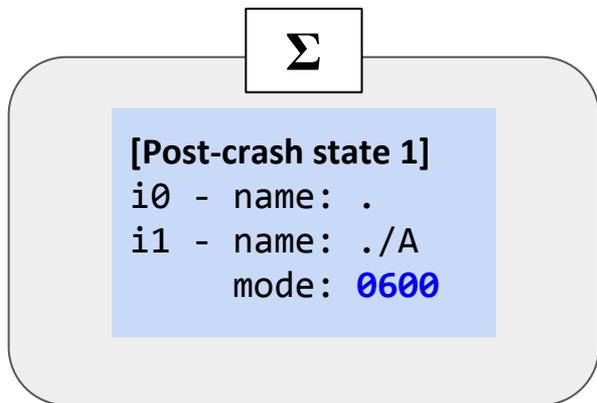
Hydra in action - Enumerating legitimate states

2. Generate possible crash states from valid snapshots



Hydra in action - Bug checking

3. Check if the set of legitimate states Σ has crashed state γ as a member

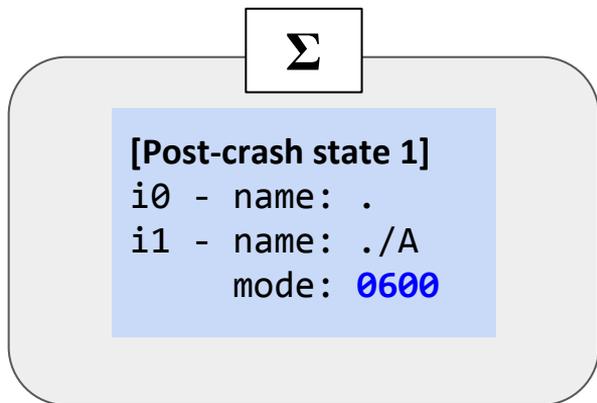


?
 \ni

Crashed F2FS image from Executor (γ)
\$ cd mnt_point
\$ stat A
Access: (0775/drwxrwxr-x)

Hydra in action - Bug found

3. Check if the set of legitimate states Σ has crashed state γ as a member



```
Crashed F2FS image from Executor ( $\gamma$ )
$ cd mnt_point
$ stat A
Access: (0775/drwxrwxr-x)
```

**None of the states have A's mode as 0775.
This is a bug! (reported and patched)**

Evaluation

Effectiveness and performance
as a fuzzing framework

Evaluation - Hydra is effective

- Hydra found **36 new semantic bugs** (+ 33 memory errors)
 - including a crash consistency bug in FSCQ, a verified file system

File System (checker)	Crash Consistency (SymC3)	Logic Bugs (In-kernel checks)	Spec. Violation (SibylFS)
ext4	1	0	1
Btrfs	4	7	2
F2FS	3	16	1
FSCQ	1	-	-
Total	9	23	4

Evaluation - Hydra is effective

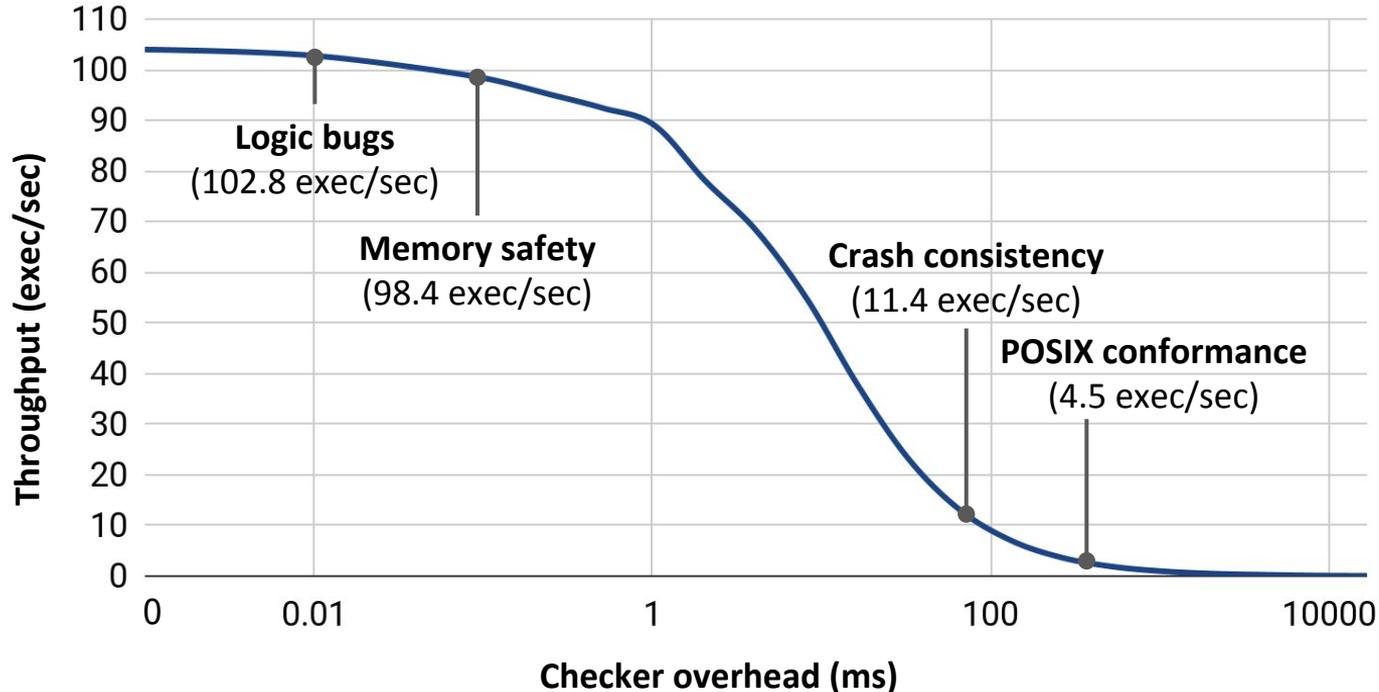
- Hydra found **36 new semantic bugs** (+ 33 memory errors)
 - including a crash consistency bug in FSCQ, a verified file system

File System (checker)	Crash Consistency (SymC3)	Logic Bugs (In-kernel checks)	Spec. Violation (SibylFS)
ext4	1	0	1
Btrfs	4	7	2
F2FS	3		
FSCQ	1		
Total	9		

Bug: dir is lost upon crash, if another file is truncated
Dev: “ftruncate was broken, and used an **unverified** helper function”

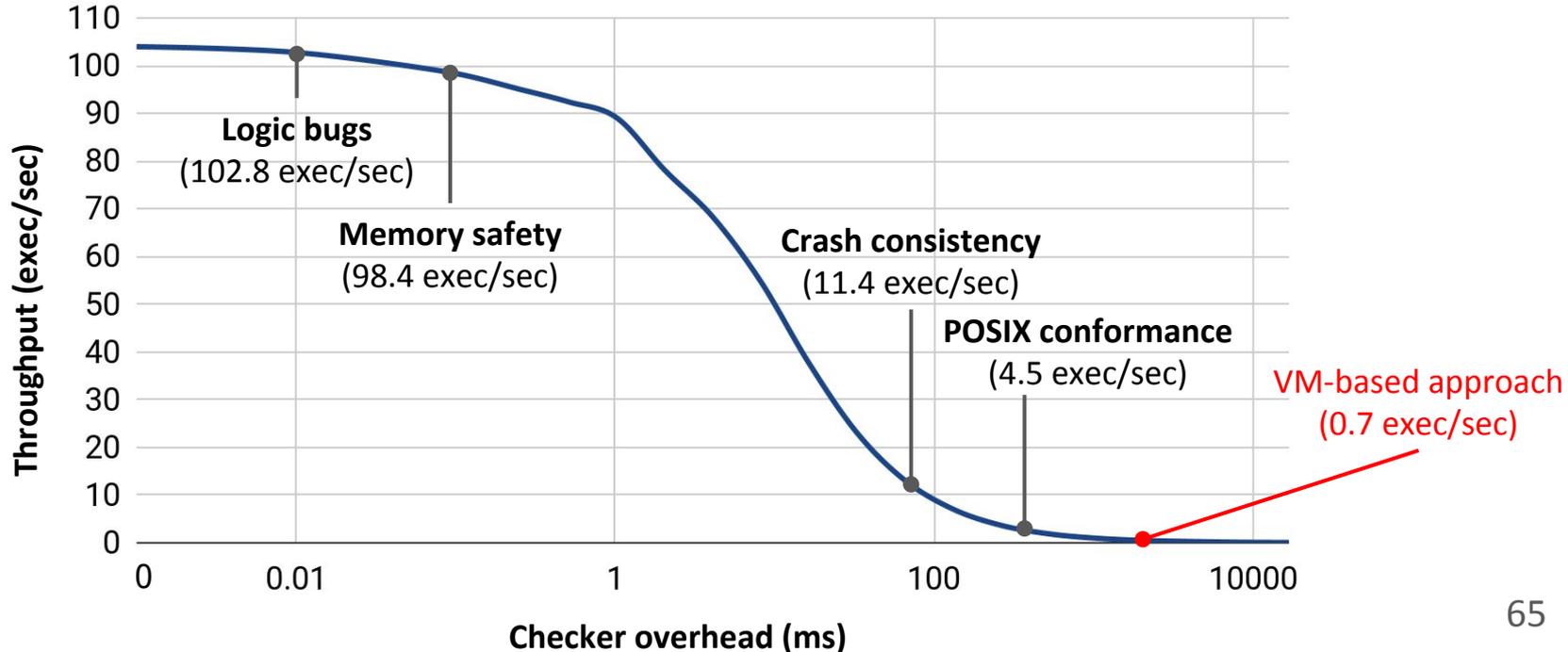
Evaluation - Hydra quickly explores input space

- Performance of Hydra's state exploration with checkers



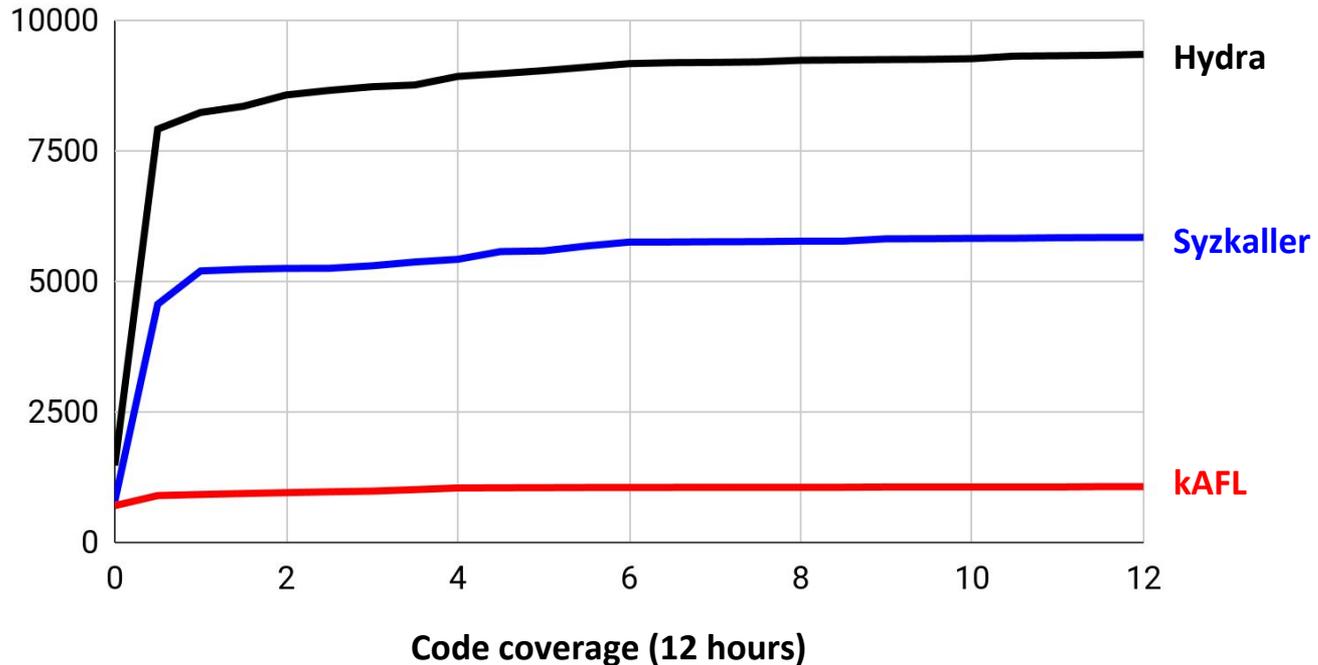
Evaluation - Hydra quickly explores input space

- Faster than VM-based kernel fuzzing



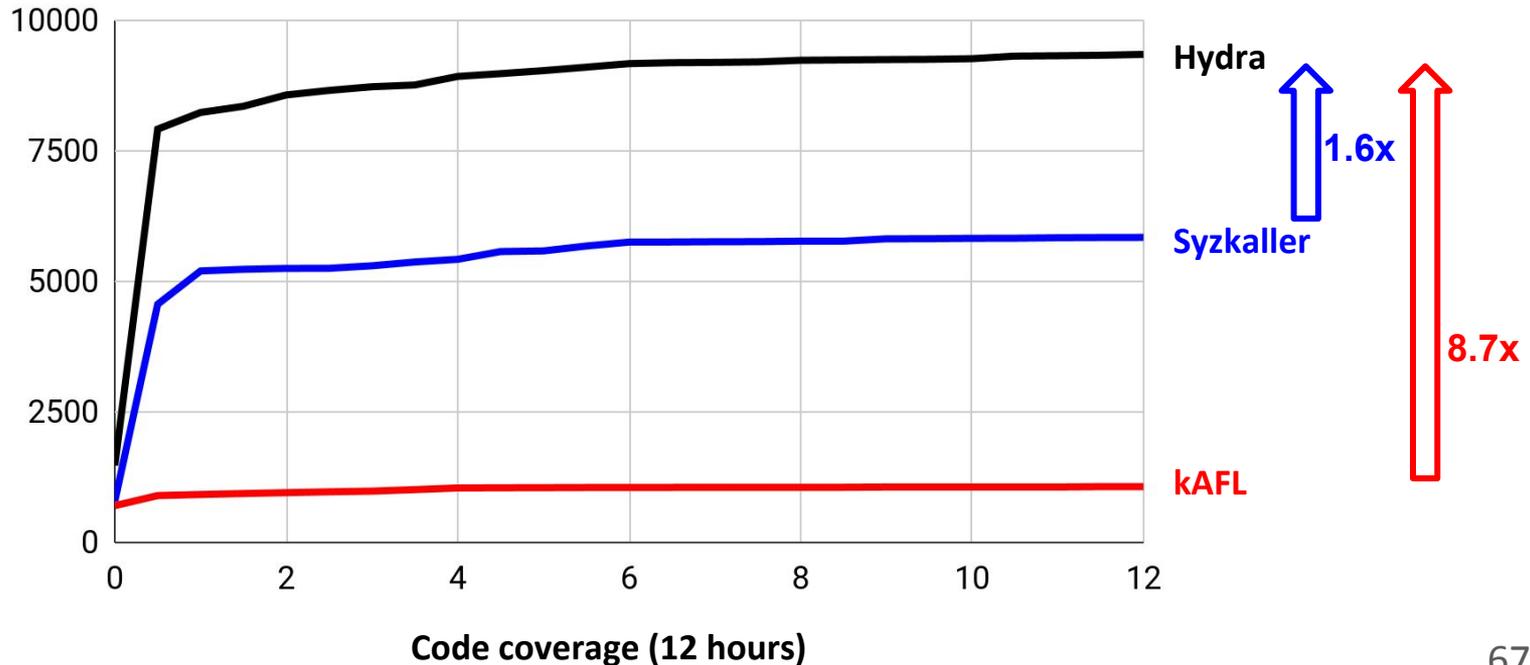
Evaluation - Hydra generates better test cases

- ext4 code coverage of Hydra vs kernel fuzzers



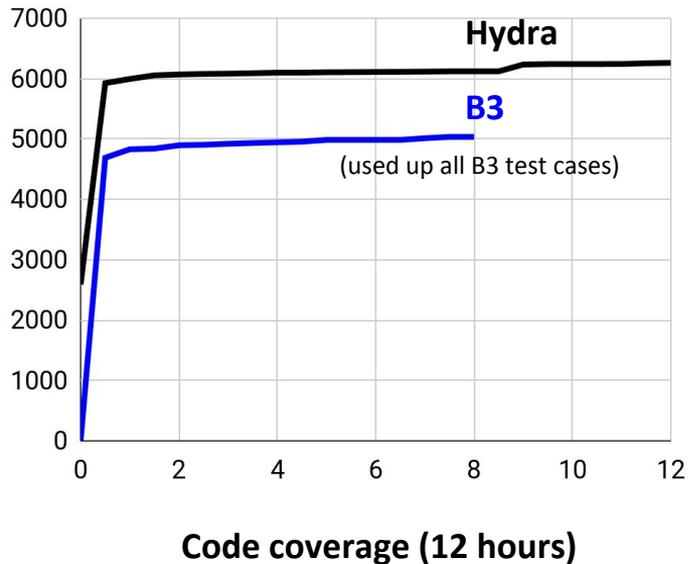
Evaluation - Hydra generates better test cases

- Hydra reaches more code paths



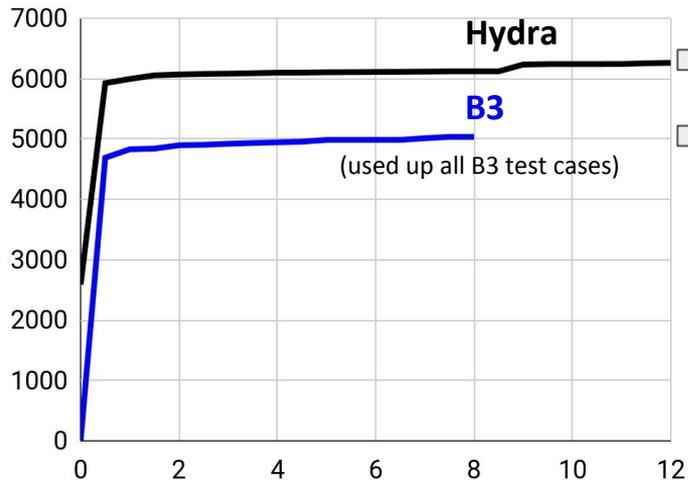
Evaluation - Hydra test cases vs B3 test suite

- B3 generates test cases by **enumerating** FS operations
 - Limits input space with bounds (e.g., #ops ≤ 3)



Evaluation - Hydra test cases vs B3 test suite

- B3 generates test cases by **enumerating** FS operations
 - Limits input space with bounds (e.g., #ops ≤ 3)



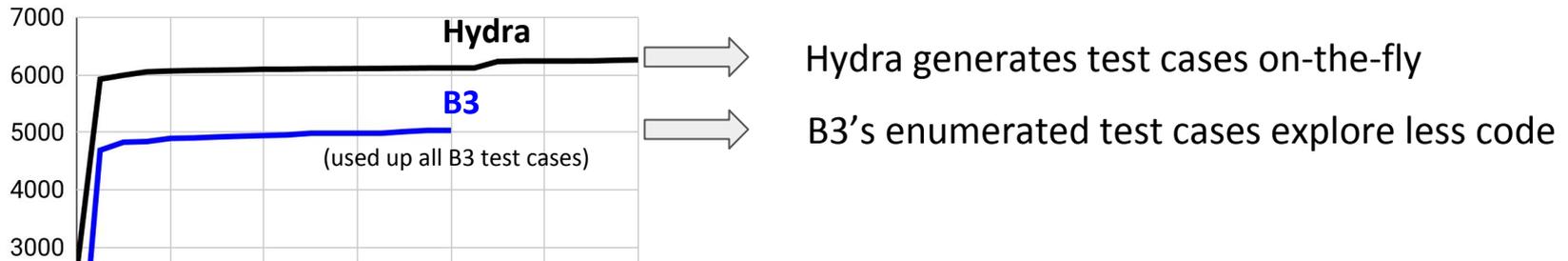
Code coverage (12 hours)

Hydra generates test cases on-the-fly

B3's enumerated test cases explore less code

Evaluation - Hydra test cases vs B3 test suite

- B3 generates test cases by **enumerating** FS operations
 - Limits input space with bounds (e.g., #ops \leq 3)



B3 missed all of the crash consistency bugs found by Hydra & SymC3

Summary

- Hydra is an extensible fuzzing framework for one-stop testing on multiple aspects of file systems
 - Open-sourced at <https://github.com/sslab-gatech/hydra>

Summary

- Hydra is an extensible fuzzing framework for one-stop testing on multiple aspects of file systems
 - Open-sourced at <https://github.com/sslab-gatech/hydra>
- Discovered hard-to-detect semantic bugs (& memory bugs)
 - 9 crash consistency bugs (1 in verified file system, FSCQ)
 - 4 POSIX violations, 23 Logic bugs, and 33 memory bugs

Summary

- Hydra is an extensible fuzzing framework for one-stop testing on multiple aspects of file systems
 - Open-sourced at <https://github.com/sslab-gatech/hydra>
- Discovered hard-to-detect semantic bugs (& memory bugs)
 - 9 crash consistency bugs (1 in verified file system, FSCQ)
 - 4 POSIX violations, 23 Logic bugs, and 33 memory bugs
- Further extensions as future work
 - More bug checkers, e.g., data race checker
 - Support for distributed file systems

Demonstration - fuzzing for 10 mins

Wait, the fuzzing result?

Thank you!

Q & A

This research is supported by

