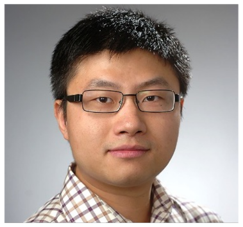
33RD USENIX SECURITY SYMPOSIUM

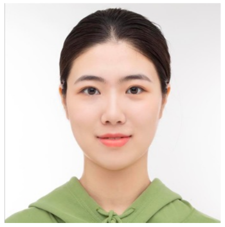
ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Understanding Ethereum Mempool Security under Asymmetric DoS by Symbolized Stateful Fuzzing





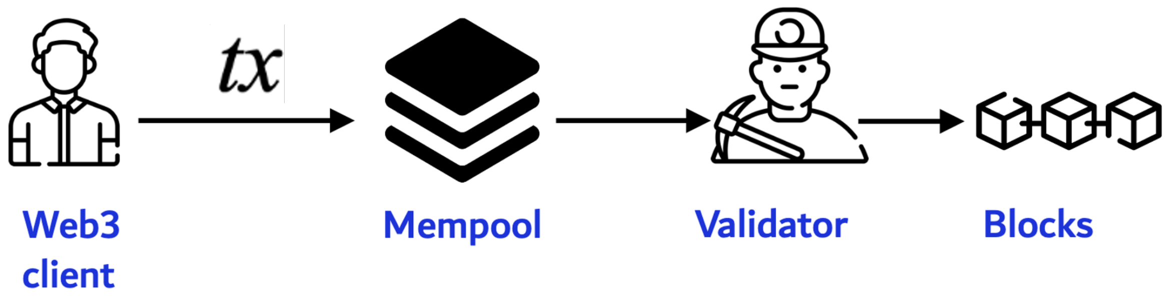
Yibo Wang, Yuzhe Tang, Kai Li, Wanning Ding, Zhihua Yang

Full Stack Security Lab @ Syracuse University

Introduction: Asymmetric Denial of Ethereum Mempool Service

Mempool - critical subsystem in blockchain

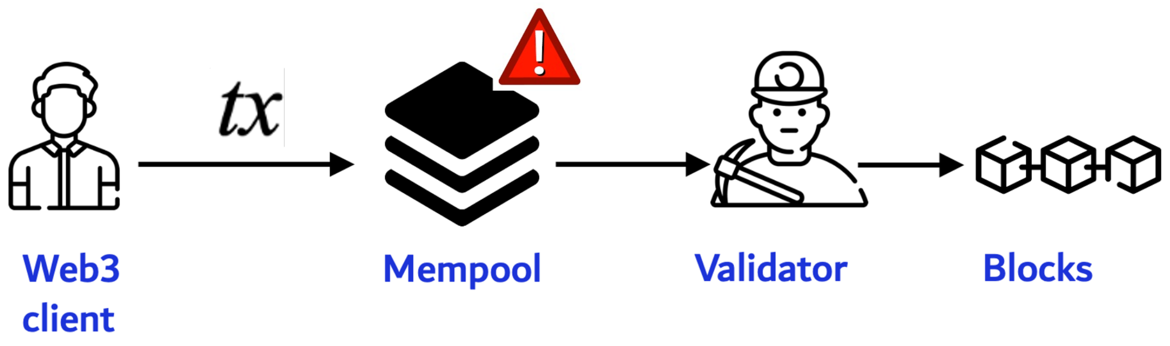
❖ A buffer of unconfirmed txs to feed validators.



Introduction: Asymmetric Denial of Ethereum Mempool Service

Mempool - critical subsystem in blockchain

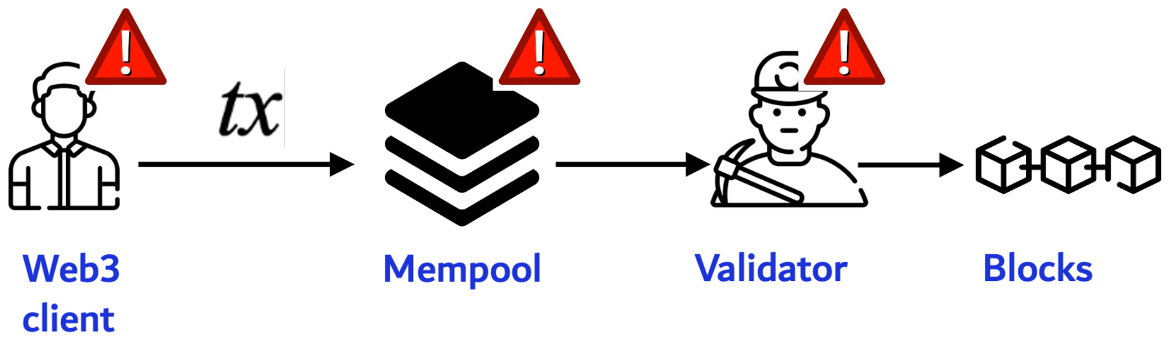
- ❖ A buffer of unconfirmed txs to feed validators.
- ❖ Victims of a denied mempool?



Introduction: Asymmetric Denial of Ethereum Mempool Service

Mempool - critical subsystem in blockchain

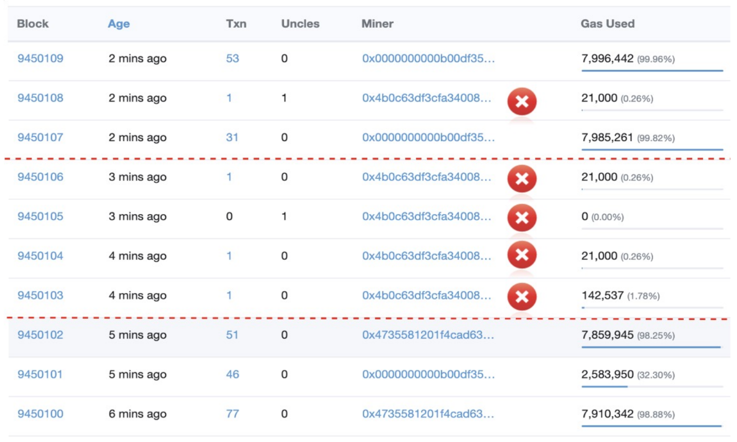
- A buffer of unconfirmed txs to feed validators.
- Victims of a denied mempool?
  - ⇒ Validator collecting zero block revenue.
  - ⇒ Web3 users unable to trade.



Introduction: Asymmetric Denial of Ethereum Mempool Service

Asymmetric mempool-DoS attacks (ADAMS)

- ❖ Existing attacks (limited): DETER [CCS'21] and MemPurge [SEC'24]
- ❖ Very practical and tested successful on Ethereum testnets.

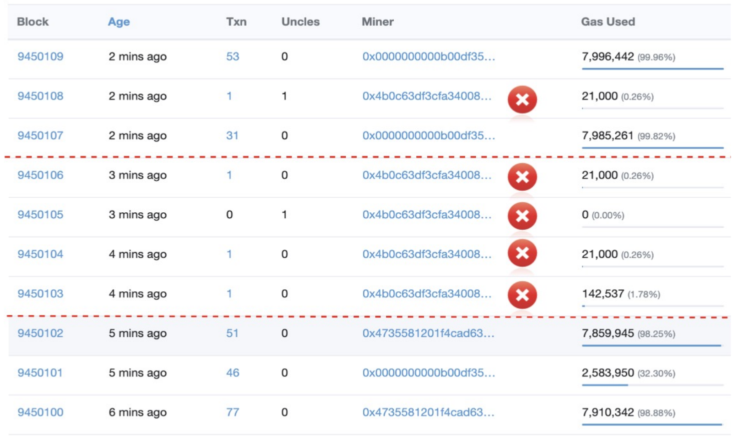


Block	Age	Txn	Uncles	Miner		Gas Used
9450109	2 mins ago	53	0	0x0000000000b00df35...		7,996,442 (99.96%)
9450108	2 mins ago	1	1	0x4b0c63df3cfa34008...	✗	21,000 (0.26%)
9450107	2 mins ago	31	0	0x0000000000b00df35...		7,985,261 (99.82%)
9450106	3 mins ago	1	0	0x4b0c63df3cfa34008...	✗	21,000 (0.26%)
9450105	3 mins ago	0	1	0x4b0c63df3cfa34008...	✗	0 (0.00%)
9450104	4 mins ago	1	0	0x4b0c63df3cfa34008...	✗	21,000 (0.26%)
9450103	4 mins ago	1	0	0x4b0c63df3cfa34008...	✗	142,537 (1.78%)
9450102	5 mins ago	51	0	0x4735581201f4cad63...		7,859,945 (98.25%)
9450101	5 mins ago	46	0	0x0000000000b00df35...		2,583,950 (32.30%)
9450100	6 mins ago	77	0	0x4735581201f4cad63...		7,910,342 (98.88%)

Introduction: Asymmetric Denial of Ethereum Mempool Service

Asymmetric mempool-DoS attacks (ADAMS)

- ❖ Existing attacks (limited): DETER [CCS'21] and MemPurge [SEC'24]
- ❖ Very practical and tested successful on Ethereum testnets.



Block	Age	Txn	Uncles	Miner		Gas Used
9450109	2 mins ago	53	0	0x0000000000b00df35...		7,996,442 (99.96%)
9450108	2 mins ago	1	1	0x4b0c63df3cfa34008...	✗	21,000 (0.26%)
9450107	2 mins ago	31	0	0x0000000000b00df35...		7,985,261 (99.82%)
9450106	3 mins ago	1	0	0x4b0c63df3cfa34008...	✗	21,000 (0.26%)
9450105	3 mins ago	0	1	0x4b0c63df3cfa34008...	✗	0 (0.00%)
9450104	4 mins ago	1	0	0x4b0c63df3cfa34008...	✗	21,000 (0.26%)
9450103	4 mins ago	1	0	0x4b0c63df3cfa34008...	✗	142,537 (1.78%)
9450102	5 mins ago	51	0	0x4735581201f4cad63...		7,859,945 (98.25%)
9450101	5 mins ago	46	0	0x0000000000b00df35...		2,583,950 (32.30%)
9450100	6 mins ago	77	0	0x4735581201f4cad63...		7,910,342 (98.88%)

Limitations of mempool-DoS research

- ❖ Attacks are manually discovered.
- ❖ Cumbersome process for always-evolving SW.
- ❖ Goal: Automatically discover mempool DoS vulnerabilities?

Introduction: Finding ADAMS Bugs by Mempool Fuzzing

Why not use existing blockchain fuzzers?

- ❖ Not to find implementation bugs or crashes:
    - ➢ AFL, Loki [NDSS’23] insufficient.
- ❖ Mempool‘s input space larger than consensus protocols
    - ➢ Mempool takes in invalid txs and txs of varying prices.
    - ➢ Consensus fuzzer, Tyr [SP’23], insufficient.
- ❖ Not a differential-fuzzing problem, as two nodes’ mempools differ
    - ➢ Differential fuzzer, Fluffy [OSDI’20], inapplicable.

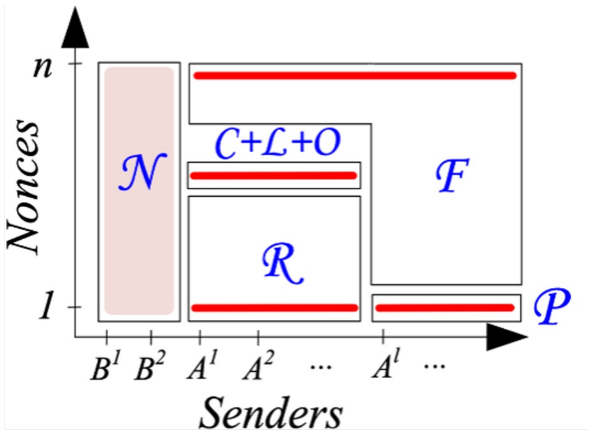
Introduction: Our Proposed Solution

mpfuzz : A symbolized stateful mempool fuzzer for finding DoS vulnerabilities.

Fuzzing Approach: Design Rationale

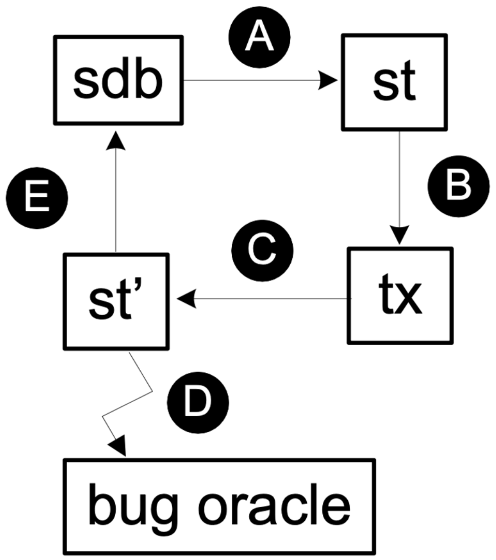
- Challenges: Huge search space
  - Stateful fuzzing⇒ but still, the state-explosion problem.
- Key idea: Search symbolized txs/states, not concrete ones
  - Observation: Same mempool behavior for a future tx $\mathcal{F}$, no matter whoever the sender is or whatever nonce value.

Symbolization: Summarize tx space into seven symbols and cover one tx per symbol during fuzzing.



n
1
Nonces
$\mathcal{N}$
$\mathcal{C}+\mathcal{L}+\mathcal{O}$
$\mathcal{F}$
$\mathcal{R}$
$\mathcal{P}$
B^1 B^2 A^1 A^2 … A^l …
Senders

Fuzzing Approach: Workflow Overview



Symbolized stateful fuzzer: seed database *sdb*

A. Select seed-state *st* from *sdb* by **energy**.
B. **Mutate tx** by instantiating symbols under *st*.
C. Send tx to MUT of state *st* and observe the end state *st'*.
D. Emit tx sequence if *st'* meets **bug oracle**.
E. Otherwise check **state feedback**, add *st'* to *sdb*; go to A.

Fuzzing Approach: Bug Oracle and Tx Mutation

- ❖ Bug oracle 1: Eviction mempool DoS
  - ➢ Full damage: No normal tx at the end state.

$$\boxed{st_0 \cap st_n = \varnothing}$$

  - ➢ Low attack cost: Low adv tx fee at the end state.

$$\boxed{asym_E(st_0, ops) \overset{def}{=} \frac{\sum_{tx \in st_n} tx.fee}{\sum_{tx \in st_0} tx.fee} < \varepsilon}$$

- ❖ Bug oracle 2: Locking mempool DoS

- ❖ Symbol-based tx mutation:
  - ➢ Txs are instantiated by symbols based on the given state.

Fuzzing Approach: Seed Selection and State Feedback

- Select seed states from sdb by energy:
  - Seed energy is determined by its symbolized cost.
  - Energy reflects its potential to trigger bug oracles.

- State feedback:
  - Symbolized state coverage feedback.
  - State promising-ness in reaching bug oracle.
    - Fewer normal txs or adversarial txs.

Evaluation: Discovered New Attacks

New attacks on latest Ethereum clients.

- ❖ Stealthier "turning"-based eviction attacks
  - ➢ Valid-turned-invalid
    - ■ Turn valid tx into overdraft.
    - ■ Turn valid tx into future.
  - ➢ Larger impact: Propagated to all nodes.

- ❖ New locking attacks
  - ➢ Occupy mempool by adversarial txs.
  - ➢ Decline arriving normal txs.
  - ➢ Lower average attack fee than victim's.

Go Ethereum

HYPERLEDGER BESU

ERIGON

NETHERMIND®

BINANCE

Reth
Made in Rust

open ethereum

Evaluation: Performance

- > 100× speedup against 4 baselines.
  - ➢ Detect DETER attacks.
  - ➢ B1: Stateless fuzzer.
  - ➢ B2: Concrete-state-coverage fuzzer.
  - ➢ B3: B2 enhanced using number of invalid txs as energy.
  - ➢ B4: mpfuzz without state promising-ness feedback.

Settings	B1	B2	B3	B4	mpfuzz
6slot-2h	Timeout	54 (min)	8 (min)	1.22 (min)	**0.03 (min)**
16slot-16h	Timeout	Timeout	447 (min)	Timeout	**0.06 (min)**

Acknowledgement & Open Source

❖ Ethereum Foundation's support via bug bounty & academic grants.

- *DoS-secure tx propagation on Ethereum: Exploit generation and attack detection.*

❖ Partially supported by NSF awards.

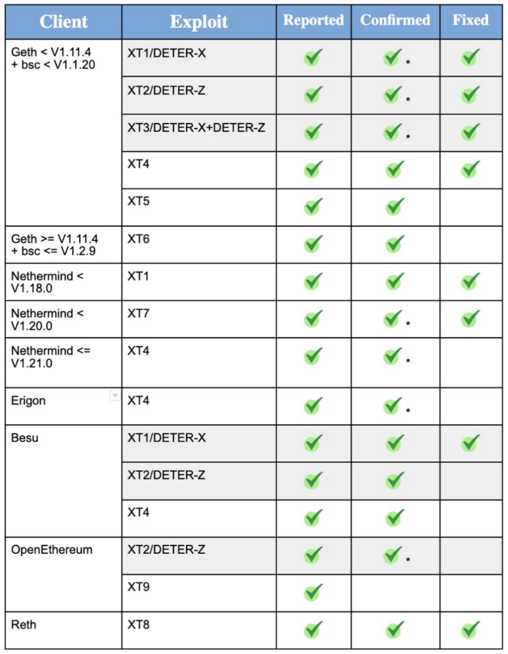
❖ Open source

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

❖ Bugs reported & confirmed by EF



Client	Exploit	Reported	Confirmed	Fixed
Geth < V1.11.4 + bsc < V1.1.20	XT1/DETER-X	✓	✓*	✓
	XT2/DETER-Z	✓	✓*	✓
	XT3/DETER-X+DETER-Z	✓	✓*	✓
	XT4	✓	✓	✓
	XT5	✓	✓	
Geth >= V1.11.4 + bsc <= V1.2.9	XT6	✓	✓	
Nethermind < V1.18.0	XT1	✓	✓	✓
Nethermind < V1.20.0	XT7	✓	✓*	✓
Nethermind <= V1.21.0	XT4	✓	✓*	
Erigon	XT4	✓	✓*	
Besu	XT1/DETER-X	✓	✓	✓
	XT2/DETER-Z	✓	✓	
	XT4	✓	✓	
OpenEthereum	XT2/DETER-Z	✓	✓*	
	XT9	✓		
Reth	XT8	✓	✓	✓

Paper Link

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Understanding Ethereum Mempool Security under Asymmetric DoS by Symbolized Stateful Fuzzing

Yibo Wang
Syracuse University
ywang349@syr.edu

Yuzhe Tang ✉*
Syracuse University
ytang100@syr.edu

Kai Li
San Diego State University
kli5@sdsu.edu

Wanning Ding
Syracuse University
wding04@syr.edu

Zhihua Yang
Syracuse University
zyang47@syr.edu

Abstract

In blockchains, mempool controls transaction flow before consensus, denial of whose service hurts the health and security of blockchain networks. This paper presents MPFUZZ, the first mempool fuzzer to find asymmetric DoS bugs by exploring the space of symbolized mempool states and optimistically estimating the promisingness of an intermediate state in reaching bug oracles. Compared to the baseline blockchain fuzzers, MPFUZZ achieves a $> 100\times$ speedup in finding known DETER exploits. Running MPFUZZ on major Ethereum clients leads to discovering new mempool vulnerabilities, which exhibit a wide variety of sophisticated patterns, including stealthy mempool eviction and mempool locking. Rule-based mitigation schemes are proposed against all newly discovered vulnerabilities.

1 Introduction

In Ethereum, a mempool buffers unconfirmed transactions from web3 users before they are included in the next blocks. Mempool provides the essential functionality to bridge the gap between varying rates of submitted transactions and rates of produced blocks, regardless of public or private transactions it serves. As shown in recent studies [29], denying a mempool service can force the blockchain to produce blocks of low or even zero (Gas) utilization, undermining validators' incentives and shrinking the blockchain networks in the long run, re-introducing the 51% attacks. Besides, a denied mempool service can prevent normal transactions from block inclusion, cutting millions of web3 users off the blockchain and failing the DApps relying on real-time blockchain access.

Problem: Spamming the mempool to deny its service has been studied for long [15, 19, 24, 34]. Early designs by sending spam transactions at high prices burden attackers with high costs and are of limited practicality. What poses a real threat is Asymmetric DeniAl of Mempool Service, coined by ADAMS, in which the mempool service is denied at an asymmetrically low cost. That is, the attack costs, in terms of the fees of adversarial transactions, are significantly lower than those of normal transactions victimized by the denied mempool. In the existing literature, DETER [29] is the first ADAMS attack, and it works by sending invalid transactions to *directly evict* normal transactions in the mempool. MemPurge [36] is a similar mempool attack that finds a way to send overdraft transactions into Geth's pending transaction pool and causes eviction there. These known attacks are easy to detect (i.e., following the same direct-eviction pattern). In fact, the DETER bugs reported in 2021 have been successfully fixed in all major Ethereum clients as of Fall 2023, including Geth, Nethermind, Erigon, and Besu. Given this state of affairs, we pose the following research question: Are there new ADAMS vulnerabilities in the latest Ethereum clients already patched against direct-eviction based attacks?

This work takes a systematic and semi-automated approach to discovering ADAMS vulnerabilities, unlike the existing DETER bugs that are manually found. Fuzzing mempool implementations is a promising approach but also poses unique challenges: Unlike the consensus implementation that reads only valid confirmed transactions, the mempool, which resides in the pre-consensus phase, needs to handle various unconfirmed transactions, imposing a much larger input space for the fuzzer. For instance, a mempool can receive *invalid transactions under legitimate causes*,[1] and factors such as fees or prices are key in determining transaction admission outcomes. Existing blockchain fuzzers including Fluffy [37], Loki [30] and Tyr [26] all focus on fuzzing consensus implementation and don't explore the extra transaction space required by mempool fuzzing. As a result, directly re-purposing a consensus fuzzer to fuzz mempool would be unable to detect the DETER bugs as evaluated in Appendix § A, let alone discover more sophisticated new ADAMS bugs.

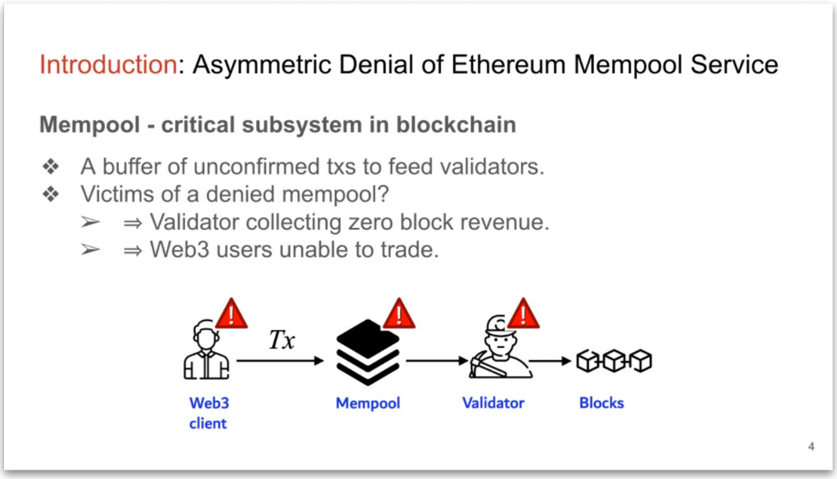
Proposed methods: To efficiently fuzz mempools, our key observation is that real-world mempool implementation admits transactions based on abstract "symbols", such as pending

*✉ Yuzhe Tang is the corresponding author.

[1] For instance, future transactions can be caused by out-of-order information propagation in Ethereum.

Q/A

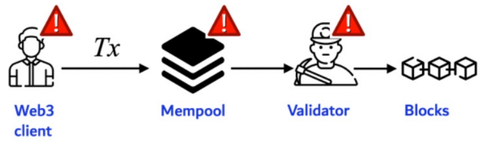
Yibo Wang ywang349@syr.edu
Dr. Yuzhe Tang ytang100@syr.edu



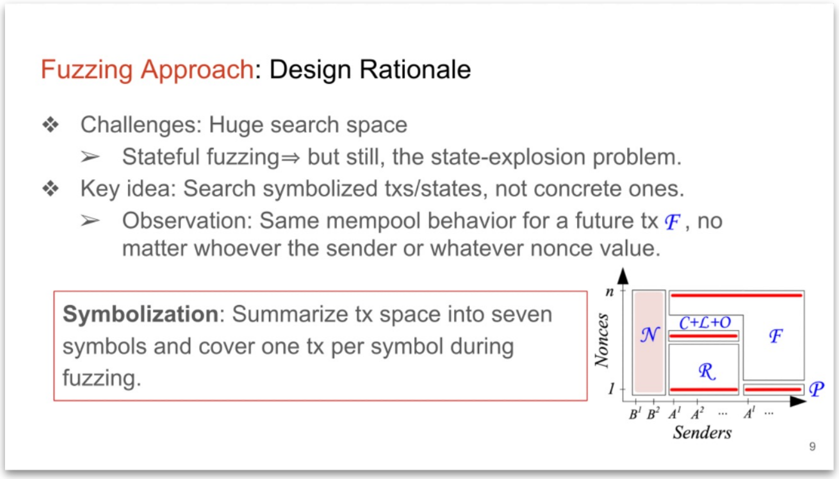
Introduction: Asymmetric Denial of Ethereum Mempool Service

Mempool - critical subsystem in blockchain

- ❖ A buffer of unconfirmed txs to feed validators.
- ❖ Victims of a denied mempool?
  - ➢ ⇒ Validator collecting zero block revenue.
  - ➢ ⇒ Web3 users unable to trade.



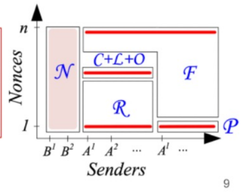
4



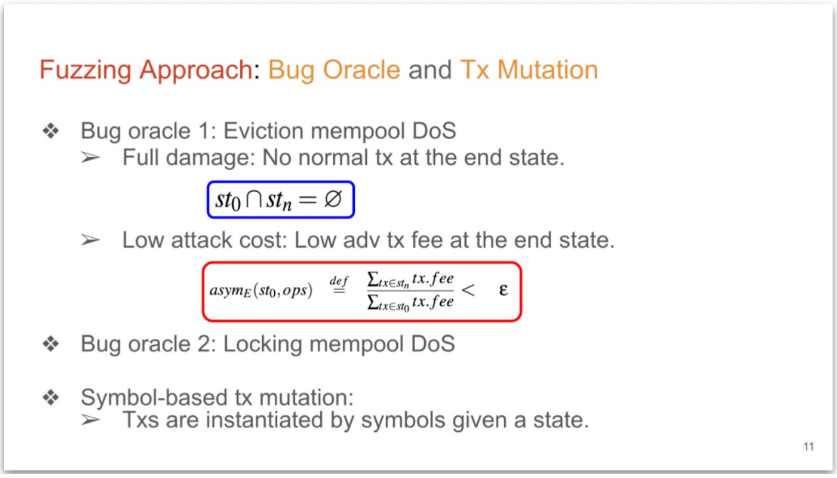
Fuzzing Approach: Design Rationale

- ❖ Challenges: Huge search space
  - ➢ Stateful fuzzing⇒ but still, the state-explosion problem.
- ❖ Key idea: Search symbolized txs/states, not concrete ones.
  - ➢ Observation: Same mempool behavior for a future tx $\mathcal{F}$, no matter whoever the sender or whatever nonce value.

Symbolization: Summarize tx space into seven symbols and cover one tx per symbol during fuzzing.


n
Nonces
1
$\mathcal{N}$
$\mathcal{C}+\mathcal{L}+\mathcal{O}$
$\mathcal{F}$
$\mathcal{R}$
$\mathcal{P}$
B^1 B^2 A^1 A^2 ... A^l ...
Senders

9



Fuzzing Approach: Bug Oracle and Tx Mutation

- ❖ Bug oracle 1: Eviction mempool DoS
  - ➢ Full damage: No normal tx at the end state.

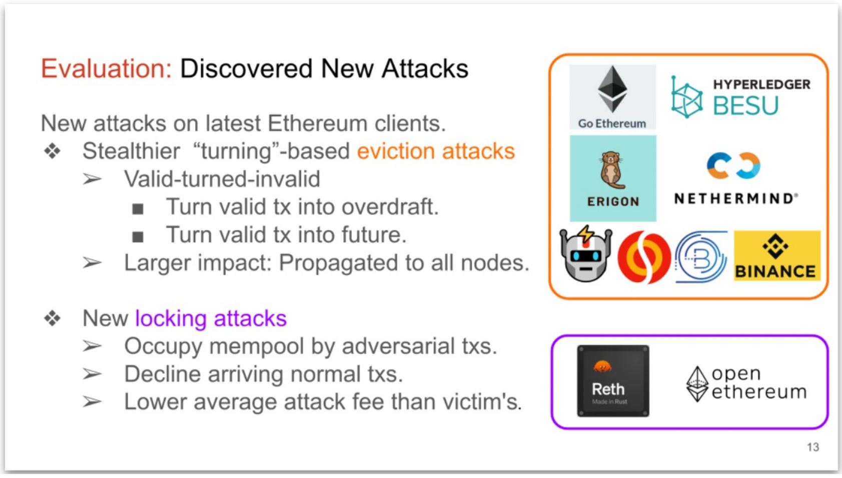
$$\boxed{st_0 \cap st_n = \emptyset}$$

  - ➢ Low attack cost: Low adv tx fee at the end state.

$$\boxed{asym_E(st_0, ops) \overset{def}{=} \frac{\sum_{tx \in st_n} tx.fee}{\sum_{tx \in st_0} tx.fee} < \varepsilon}$$

- ❖ Bug oracle 2: Locking mempool DoS
- ❖ Symbol-based tx mutation:
  - ➢ Txs are instantiated by symbols given a state.

11



Evaluation: Discovered New Attacks

New attacks on latest Ethereum clients.

- ❖ Stealthier "turning"-based eviction attacks
  - ➢ Valid-turned-invalid
    - ■ Turn valid tx into overdraft.
    - ■ Turn valid tx into future.
  - ➢ Larger impact: Propagated to all nodes.
- ❖ New locking attacks
  - ➢ Occupy mempool by adversarial txs.
  - ➢ Decline arriving normal txs.
  - ➢ Lower average attack fee than victim's.

13