Finding Consensus Bugs in Ethereum via Multi-transaction Differential Fuzzing

Youngseok Yang; Taesoo Kim, Byung-Gon Chun

Seoul National University ;Georgia Institute of Technology; Seoul National University

OSDI2021

OVERVIEW

- The auther finds two **consensus bugs** in ethereum via **multi-transaction differential fuzzing**.

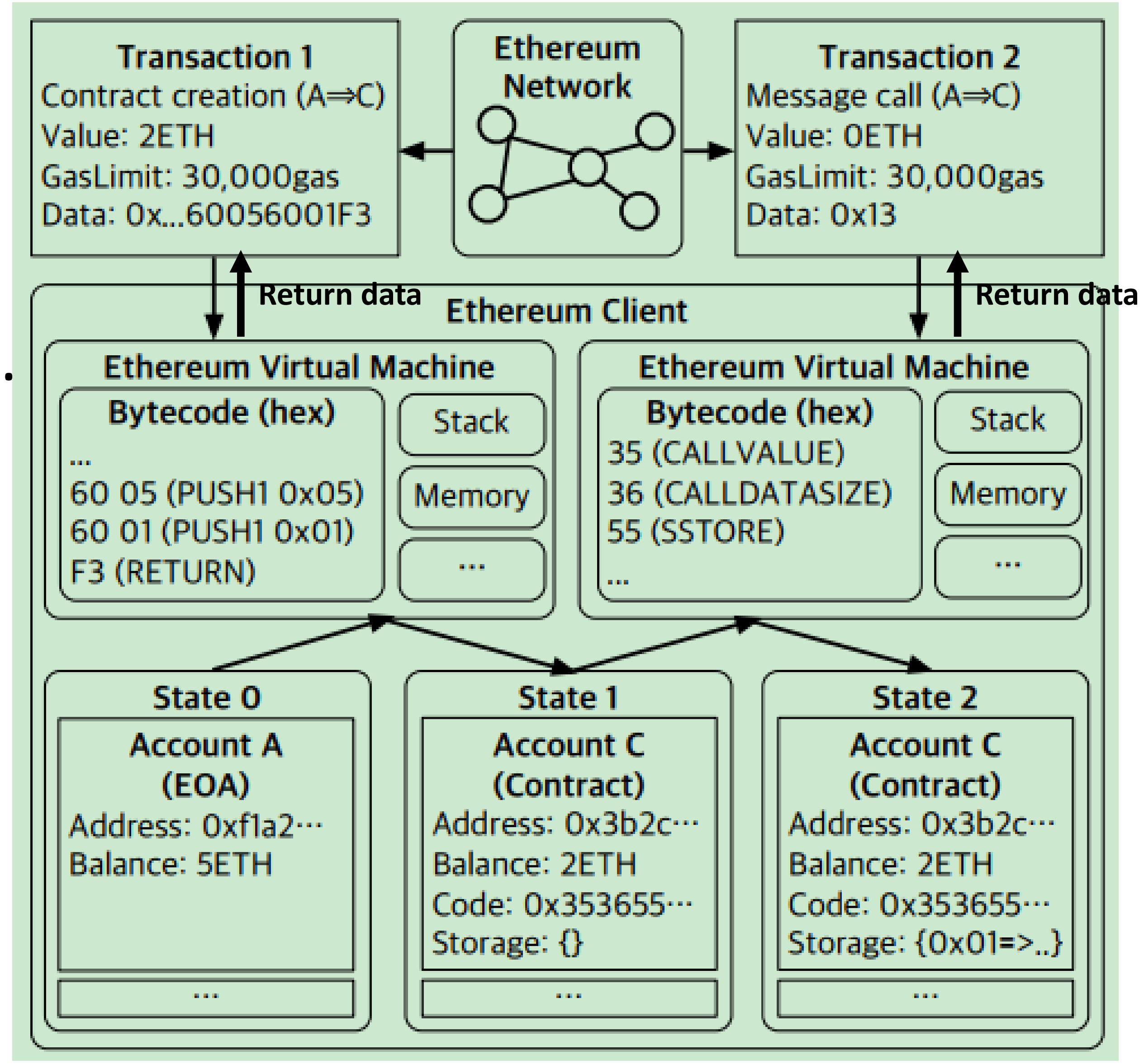
➢ two consensus bugs.

➢ multi-transaction differential fuzzing technology.

- Summary： 漏洞挺有价值，但是fuzzing技术有点玄。

backgroud

1. **Ethereum**: Blockchain platform.
2. **Transaction**: Transform, contract create and invoke.
3. **Smart Contract**: Execution program.
4. **EVM**: Virtual machine.
5. **EVM instructions**: Architectural instruction set.
6. **State**: Persistent storage.
7. **Acount**: EOA and Contract account.
8. **ETH**： Cryptocurrency of the Ethereum.
9. **Gas**： Price of contract execution unit.
10. **Return data:** data of the execute results.
11. **Deployed bytecode** and **execute bytecode**



backgroud

1.Consensus

Consensus is reached by decentralized clients that various implements the Ethereum Virtual Machine (EVM) specification.

2.Various ethereum implements： support multiple program language

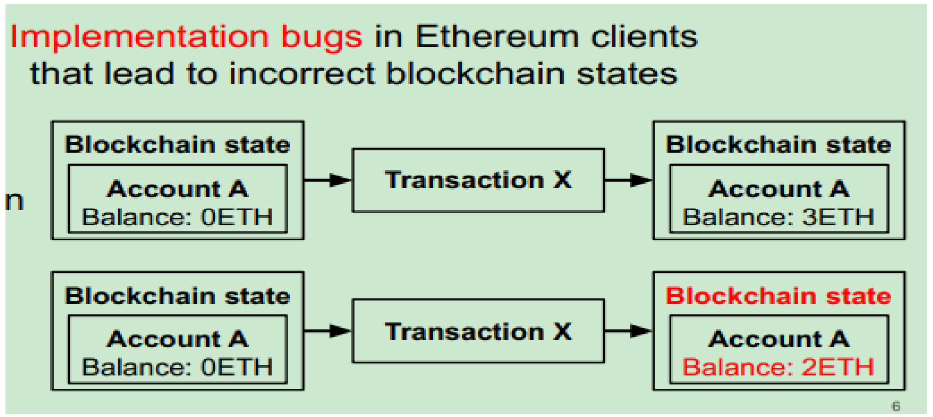
- Go-ethereum(geth), written in Golang
- open-ethereum,written in Rust
- cpp-ethereum, written in C++
- pyethereum, written in Python

$\xrightarrow{\text{follow}}$ Ethereum Yellow paper： document with Formal definition

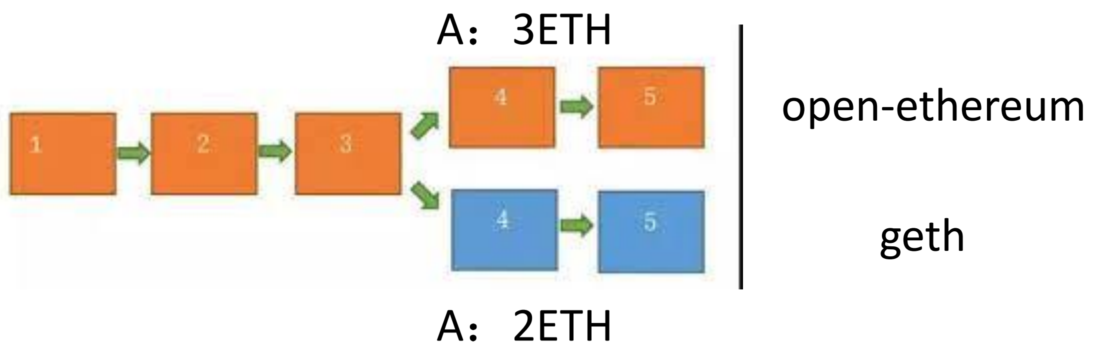
3.Consensus bug

open-ethereum

geth


Implementation bugs in Ethereum clients that lead to incorrect blockchain states

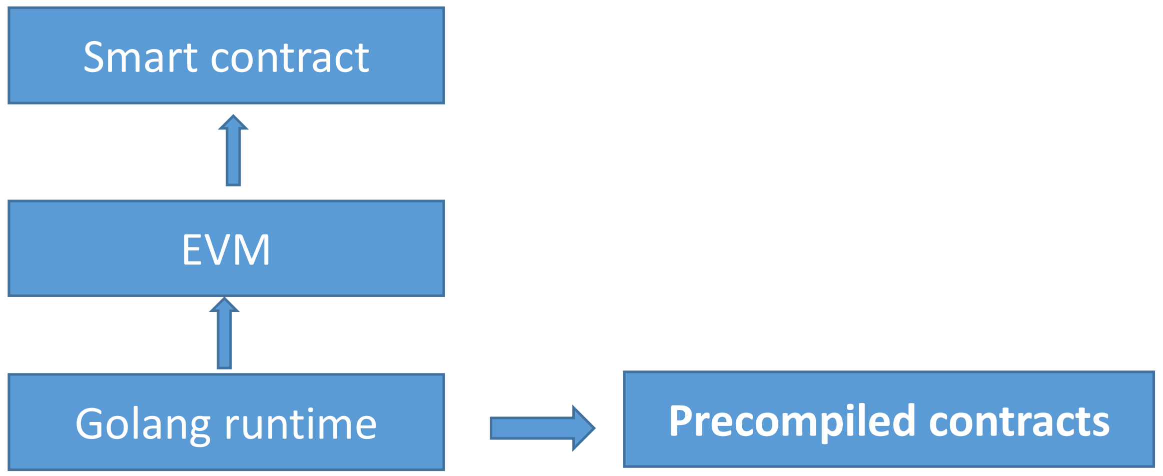
区块链分叉



background

1.Precompiled contracts： Precompiled contracts meant as a preliminary piece of architecture that may later become native extensions.

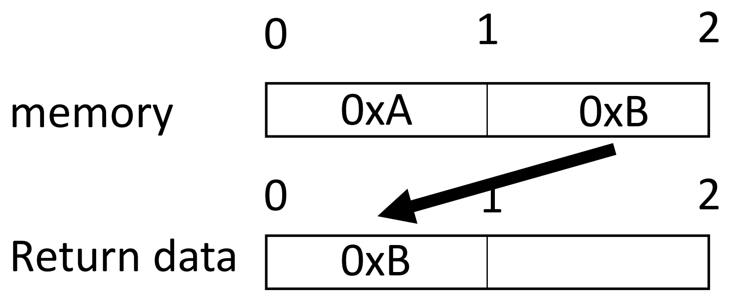
Precompiled contract name	Features	Address
ecrecover()	Recovery of ECDSA signature	0x1
sha256hash()	Hash function SHA256	0x2
ripemd160hash()	Hash function RIPEMD160	0x3
dataCopy()	Identity	0x4
bigModeExp()	Modular exponentiation	0x5
bn256Add()	Addition on elliptic curve alt_bn128	0x6
bn256ScalarMul()	Scalar multiplication on elliptic curve alt_bn128	0x7
bn256Pairing()	Checking a pairing equation on curve alt_bn128	0x8



2.datacopy()： It copies its input(memory) to its output(return data).

Call(addr, inoffset, insize , retoffset , retsize)

- addr=0x4
- Inoffset=1
- Insize=1
- retoffset=0
- retsize=1



Consensus bug

- ➢ Transfer-after-destruct bug
- ➢Shallow copy bug

- 这两个bug都是因为go-ethereum(geth)未按照黄皮书中的标准来实现以太坊虚拟机（EVM），从而导致区块链的共识被打破。
- 两个bug都有点难理解。

Consensus bug Transfer-after-destruct bug

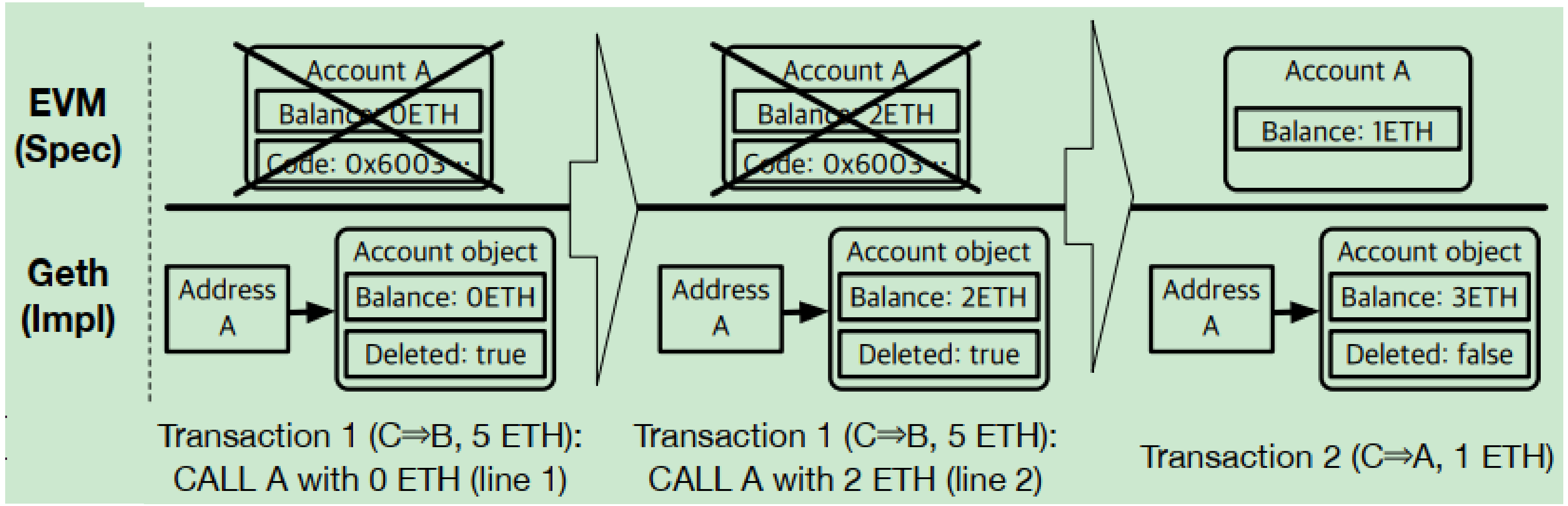
合约A

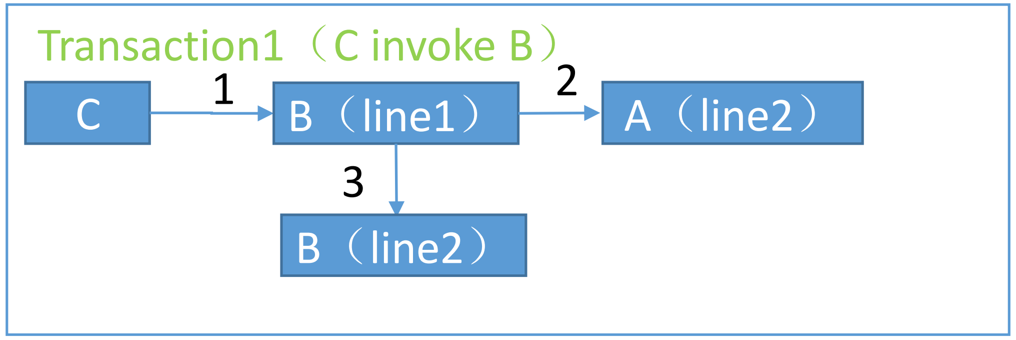
```
// Contract (Address: A)
1: If VALUE == 0
2:   SELFDESTRUCT
3: ELSE
4:   STOP
```

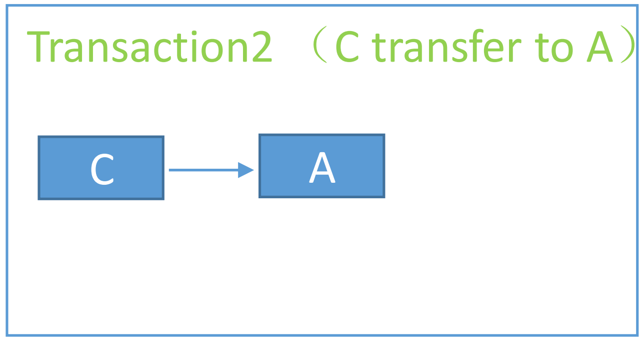
合约B

```
// Contract (Address: B)
1: CALL A with 0 ETH
2: CALL A with 2 ETH
```

攻击者C

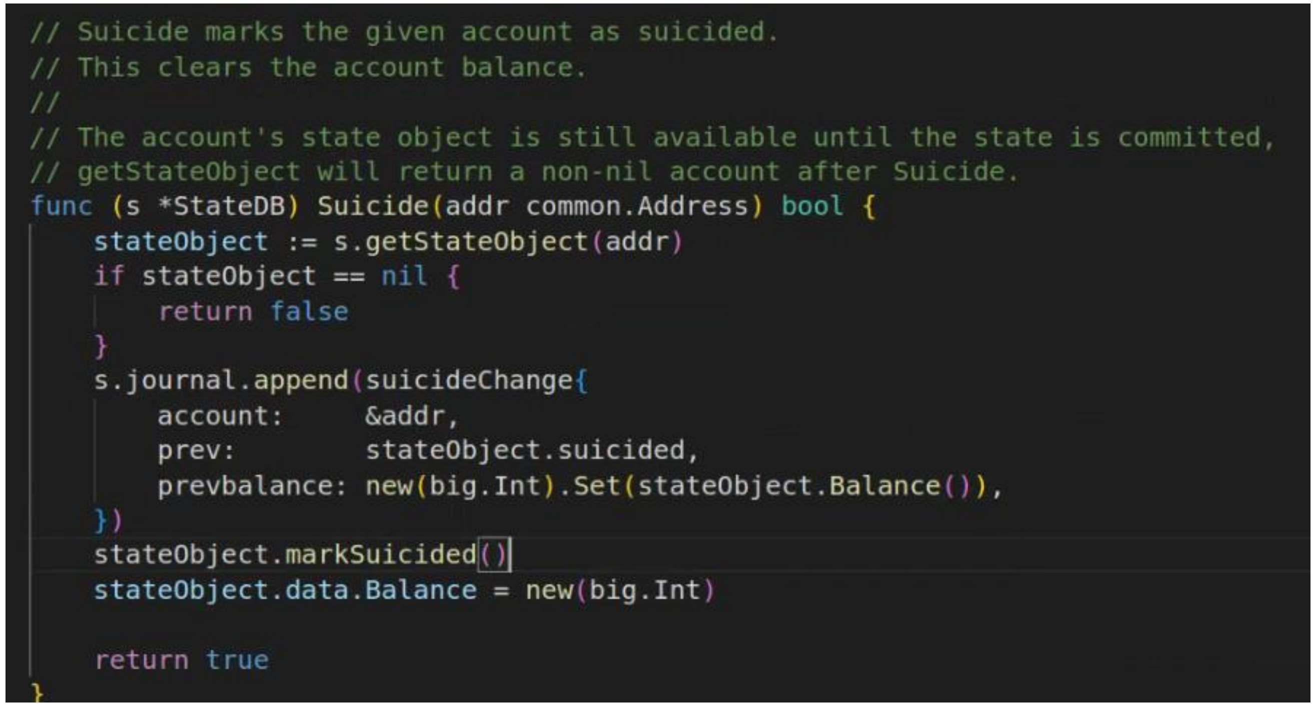






Tips
Selfdestruct： 函数即可自毁合约。
存在合约中的以太币将会发送到设计好的地址里，剩下的代码和存储变量将会在状态机中被移除。
被销毁的账户可以被重新创建。

Consensus bug Transfer-after-destruct bug



```
// Suicide marks the given account as suicided.
// This clears the account balance.
//
// The account's state object is still available until the state is committed,
// getStateObject will return a non-nil account after Suicide.
func (s *StateDB) Suicide(addr common.Address) bool {
    stateObject := s.getStateObject(addr)
    if stateObject == nil {
        return false
    }
    s.journal.append(suicideChange{
        account:     &addr,
        prev:        stateObject.suicided,
        prevbalance: new(big.Int).Set(stateObject.Balance()),
    })
    stateObject.markSuicided()
    stateObject.data.Balance = new(big.Int)

    return true
}
```

交易1执行完了之后，account A才会删除

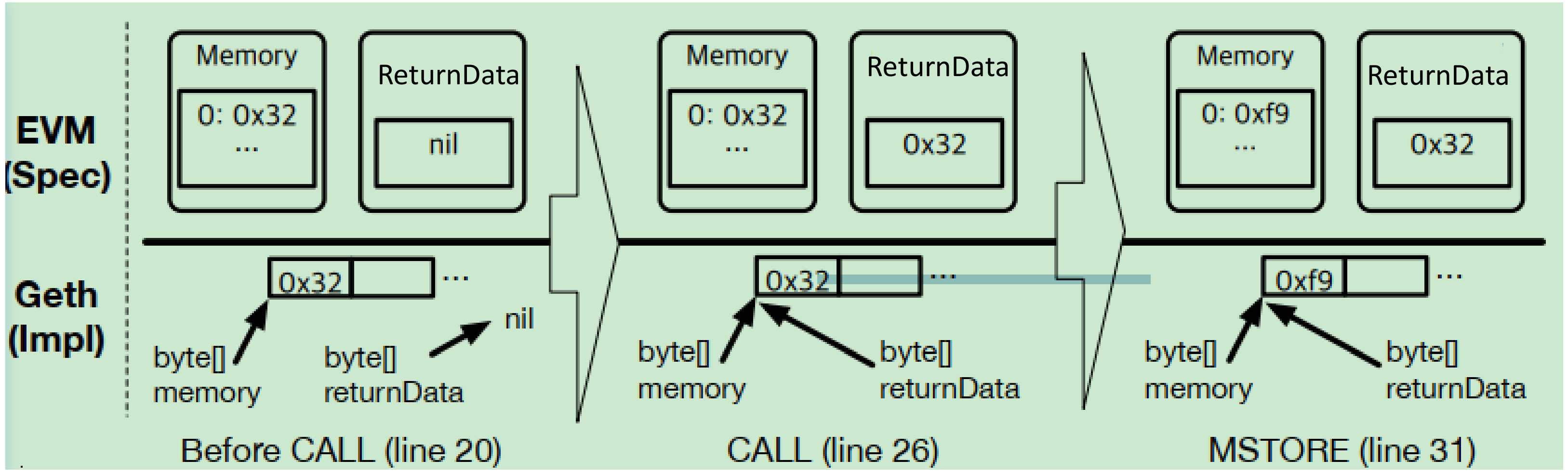
Consensus bug Shallow copy bug

构造攻击合约，调用datacopy()预编译合约

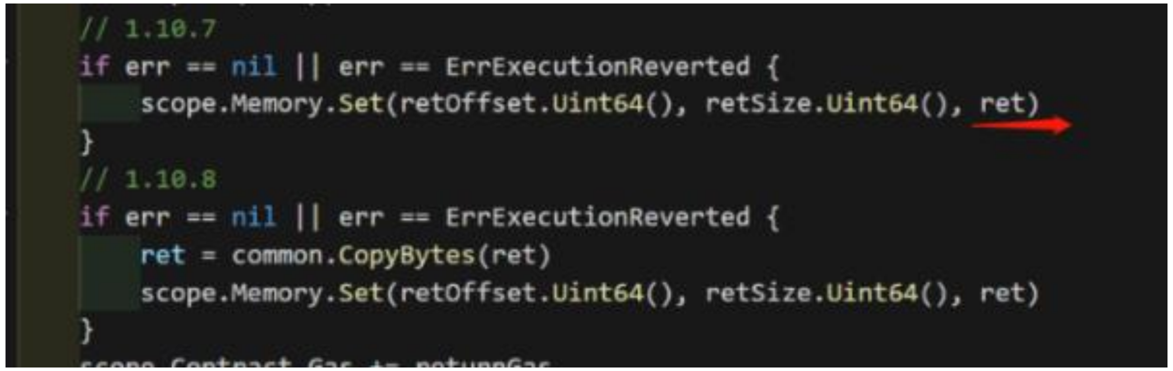
Line 1-20
move 0x32 to memory[0:]
init the parameter for datacopy()

21: PUSH1 0X04 // addr
23: PUSH2 0xffff // gas
26: CALL

//破坏memory（return data）
27. PUSH1 0xf9 // value
29: PUSH1 0x00 // offset
31: MSTORE



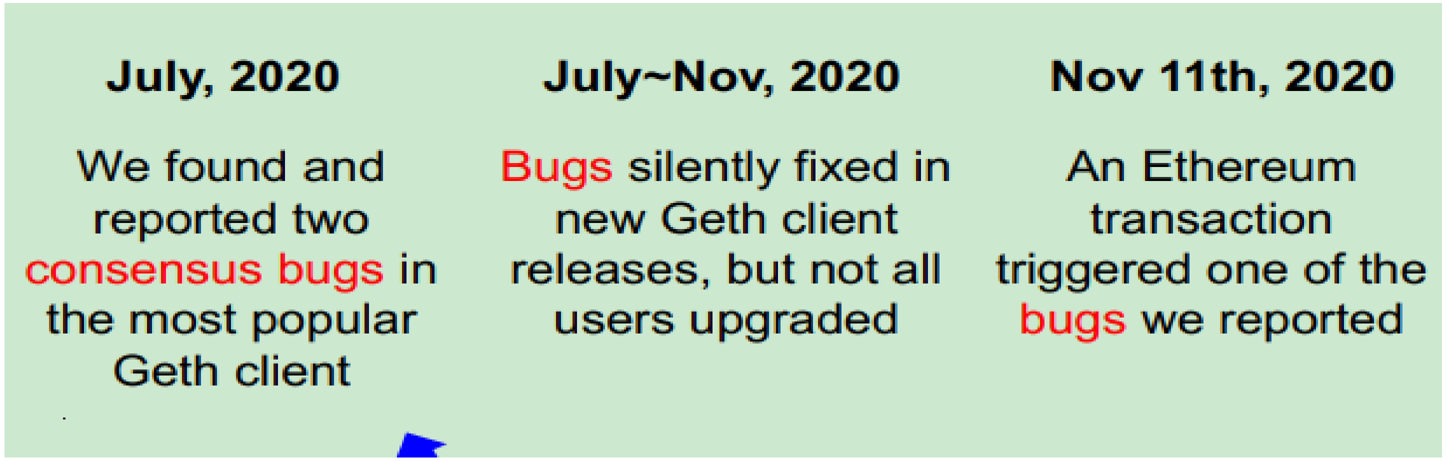
完成上述的步骤后，此时的bug造成的影响尚在内存中，还需要借助以下三个指令将Bug 造成的影响写入持久化存储层（state），RETURNDATACOPY，MLOAD，SSTORE.



```
// 1.10.7
if err == nil || err == ErrExecutionReverted {
    scope.Memory.Set(retOffset.Uint64(), retSize.Uint64(), ret)
}
// 1.10.8
if err == nil || err == ErrExecutionReverted {
    ret = common.CopyBytes(ret)
    scope.Memory.Set(retOffset.Uint64(), retSize.Uint64(), ret)
}
```

Consensus bug Shallow copy bug

Shallow copy bug 实际被利用并造成损失



July, 2020

We found and reported two consensus bugs in the most popular Geth client

July~Nov, 2020

Bugs silently fixed in new Geth client releases, but not all users upgraded

Nov 11th, 2020

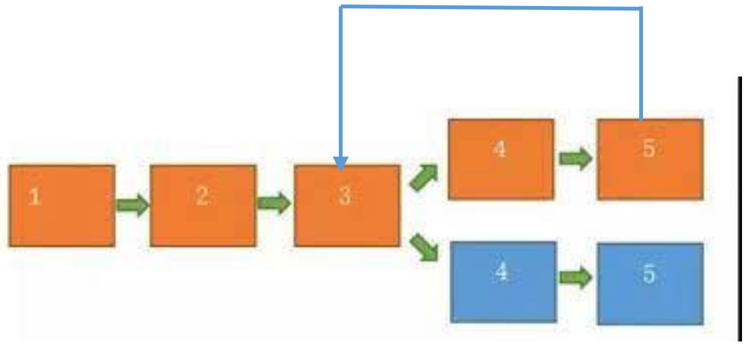
An Ethereum transaction triggered one of the bugs we reported

谁利用了？

- 普通用户无意触发。
- 漏洞发现者。
- 有人审计代码发现了，想套利。

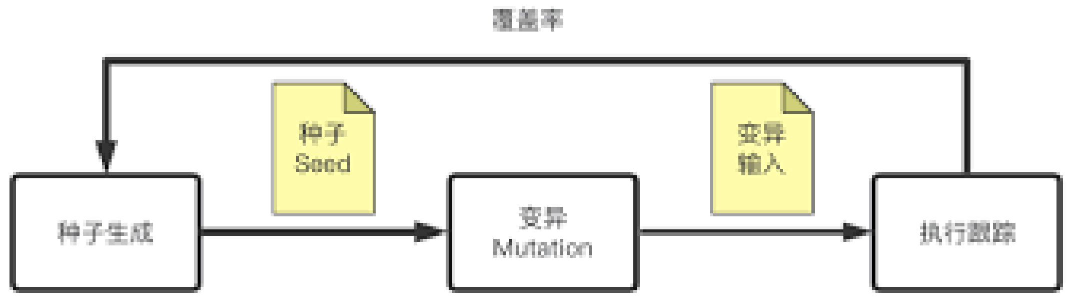
造成损失：30个区块被回滚，涉及到价值8.6M $的ETH。

回滚：



Fuzzing technology Fuzzing

Fuzzing (模糊测试)：自动化程序分析和测试技术。

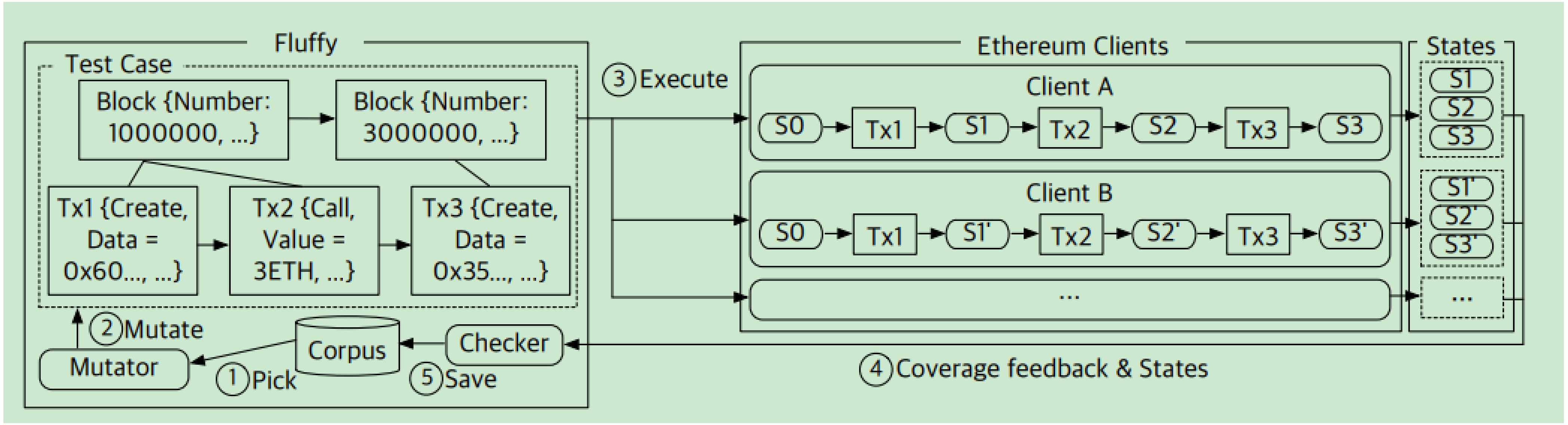


1. 选择一些**种子库**，作为初始测试集加入输入队列；
2. 将队列中的测试用例按一定的策略进行“**变异**”；（0xAA=>0xBB,0xAAAA)
3. 将变异后的测试用例放入程序执行，如果执行发现覆盖范围增补，则将其保留添加到种子库中；
4. 上述过程会一直循环进行，期间触发了异常的测试用例会被记录下来，用以分析。

模糊测试中最重要的是**输入的测试用例**，好的测试用例能够更容易的发现问题。
影响输入=>**种子库**和**编译策略**。

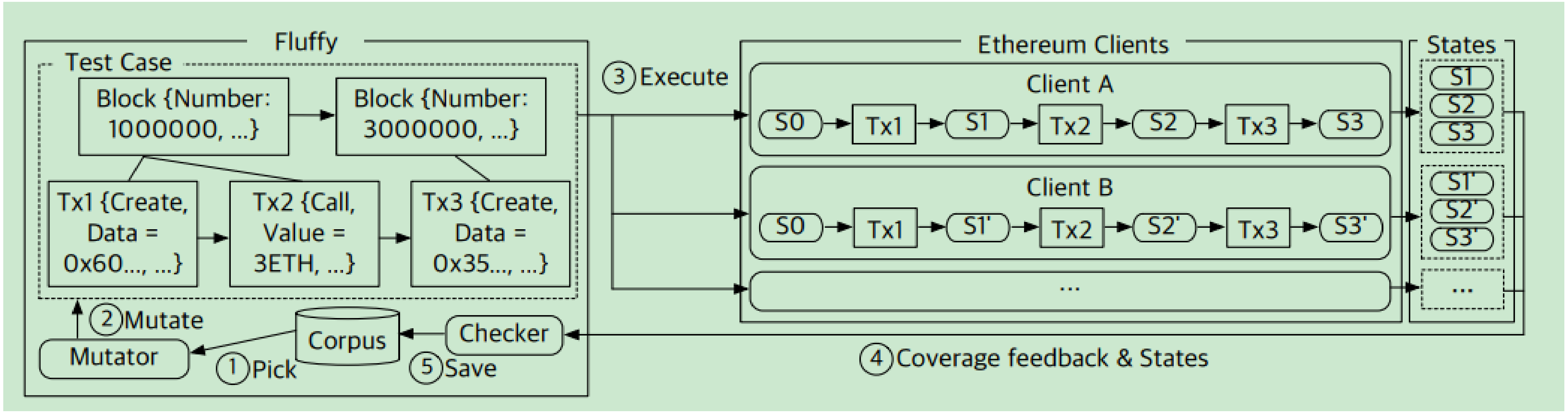
Fuzzing technology Fluffy

The author develop a tool called **Fluffy,** a multi-transaction[1] differential[2] fuzzer for finding consensus bugs in Ethereum


Fluffy
Test Case
Block {Number: 1000000, ...}
Block {Number: 3000000, ...}
Tx1 {Create, Data = 0x60..., ...}
Tx2 {Call, Value = 3ETH, ...}
Tx3 {Create, Data = 0x35..., ...}
②Mutate
Mutator
①Pick
Corpus
Checker
⑤Save
③Execute
Ethereum Clients
Client A
S0 Tx1 S1 Tx2 S2 Tx3 S3
Client B
S0 Tx1 S1' Tx2 S2' Tx3 S3'
...
States
S1 S2 S3
S1' S2' S3'
...
④Coverage feedback & States

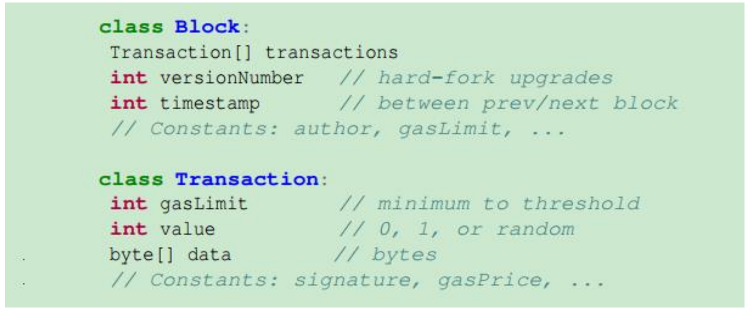
1.从种子库中获取数据（现**变异策略**有区块和交易序列）。
2.对交易序列采用，生成新的测试用例（区块和交易序列）。=》与现有工作不同，也因此发现了第一个bug。
3. Fluffy在两个以太坊客户端上执行新的测试用例。
4.执行完成后，Fluffy收集新的状态和覆盖反馈。
5.如果有新的代码路径，Fluffy保存新的测试用例；如果有状态不一致，则可能发现共识漏洞。

Fuzzing technology Mutation stra


Fluffy
Test Case
Block {Number: 1000000, ...}
Block {Number: 3000000, ...}
Tx1 {Create, Data = 0x60..., ...}
Tx2 {Call, Value = 3ETH, ...}
Tx3 {Create, Data = 0x35..., ...}
③Execute
Ethereum Clients
Client A
S0
Tx1
S1
Tx2
S2
Tx3
S3
Client B
S0
Tx1
S1'
Tx2
S2'
Tx3
S3'
...
States
S1
S2
S3
S1'
S2'
S3'
...
②Mutate
Mutator
①Pick
Corpus
Checker
⑤Save
④Coverage feedback & States

变异策略

上下文变异：对交易list进行添加和删除，以及创建交易的副本或将其内容复制到其他交易中；
字节码变异：对创建智能合约的交易中提供的智能合约字节码进行精确的变异；
参数变异：随机生成交易的参数，如交易接受者，gas值等参数。



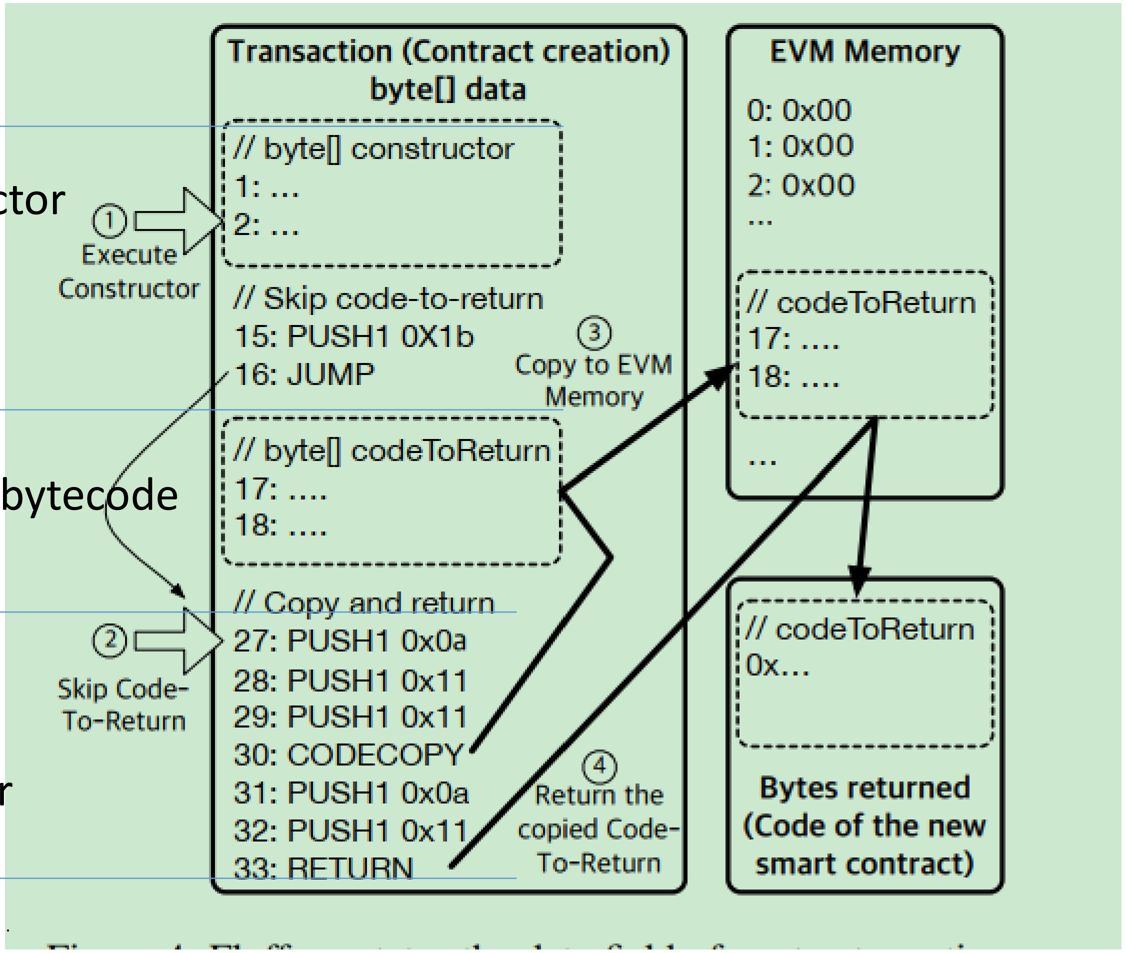
```
class Block:
  Transaction[] transactions
  int versionNumber    // hard-fork upgrades
  int timestamp        // between prev/next block
  // Constants: author, gasLimit, ...

class Transaction:
  int gasLimit         // minimum to threshold
  int value            // 0, 1, or random
  byte[] data          // bytes
  // Constants: signature, gasPrice, ...
```

Fuzzing technology bytecode Mutation strategy（字节码变异策略）

Constructor

Execute bytecode

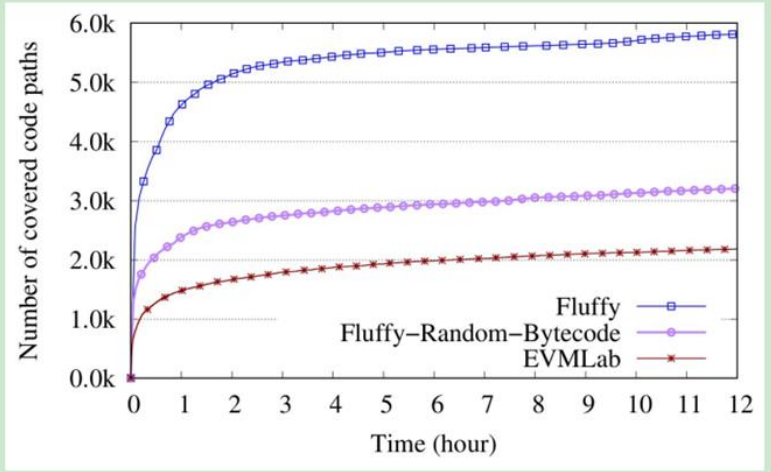
Loader



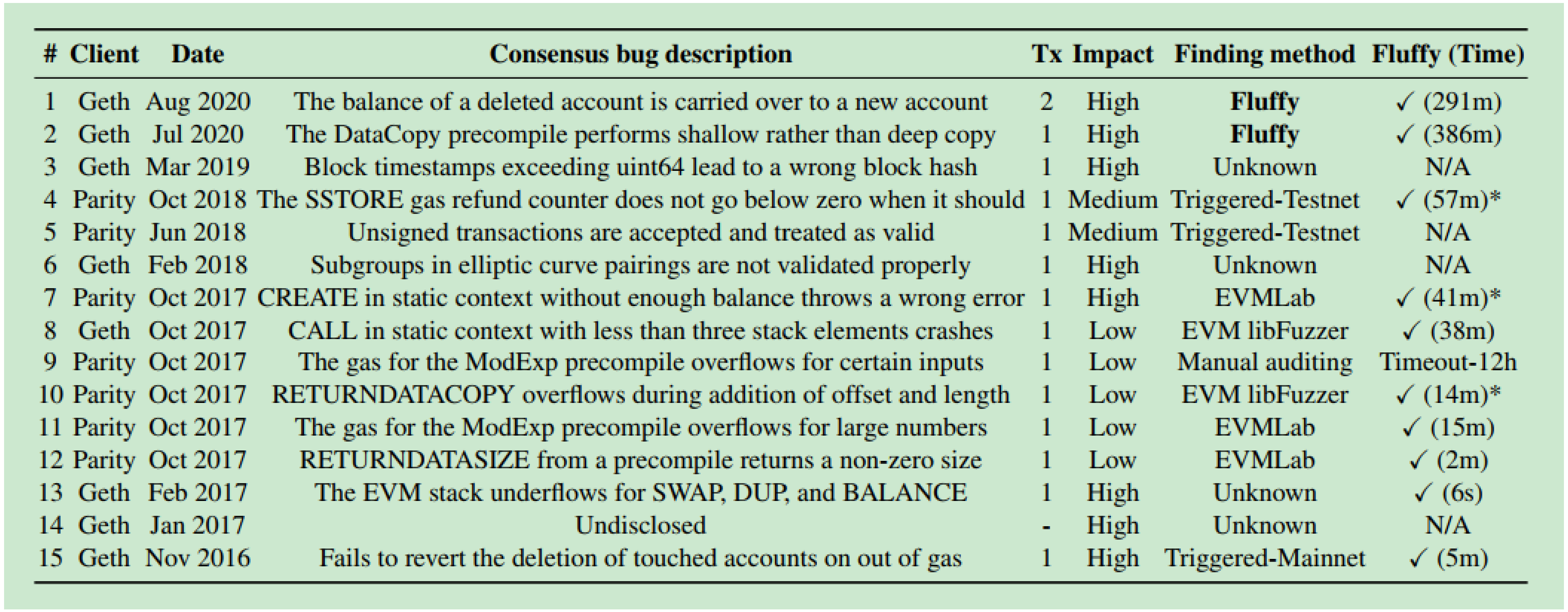
Deployed Contract ByteCode

- Constructor： 初始化参数状态 <=变异
- Execute bytecode： 实际代码逻辑
- Loader: 状态代码进入memory

Code coverage (Higher is better)



Evalution bug finding capability



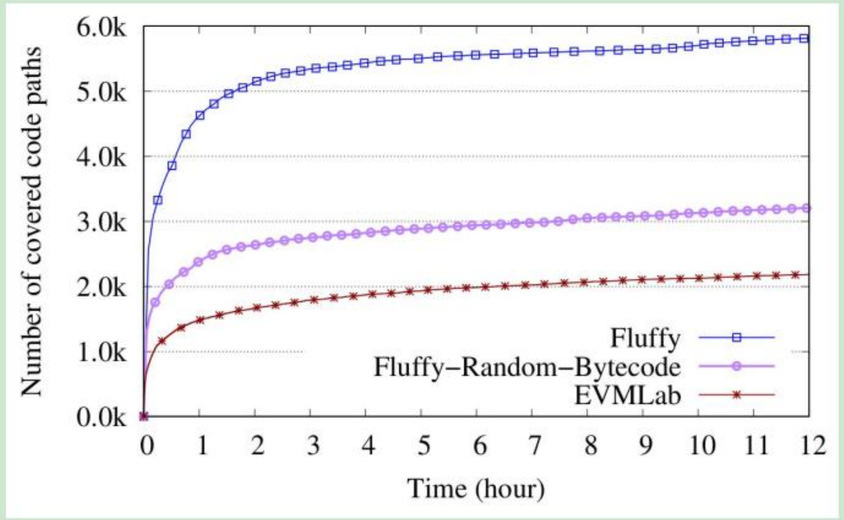
#	Client	Date	Consensus bug description	Tx	Impact	Finding method	Fluffy (Time)
1	Geth	Aug 2020	The balance of a deleted account is carried over to a new account	2	High	**Fluffy**	✓ (291m)
2	Geth	Jul 2020	The DataCopy precompile performs shallow rather than deep copy	1	High	**Fluffy**	✓ (386m)
3	Geth	Mar 2019	Block timestamps exceeding uint64 lead to a wrong block hash	1	High	Unknown	N/A
4	Parity	Oct 2018	The SSTORE gas refund counter does not go below zero when it should	1	Medium	Triggered-Testnet	✓ (57m)*
5	Parity	Jun 2018	Unsigned transactions are accepted and treated as valid	1	Medium	Triggered-Testnet	N/A
6	Geth	Feb 2018	Subgroups in elliptic curve pairings are not validated properly	1	High	Unknown	N/A
7	Parity	Oct 2017	CREATE in static context without enough balance throws a wrong error	1	High	EVMLab	✓ (41m)*
8	Geth	Oct 2017	CALL in static context with less than three stack elements crashes	1	Low	EVM libFuzzer	✓ (38m)
9	Parity	Oct 2017	The gas for the ModExp precompile overflows for certain inputs	1	Low	Manual auditing	Timeout-12h
10	Parity	Oct 2017	RETURNDATACOPY overflows during addition of offset and length	1	Low	EVM libFuzzer	✓ (14m)*
11	Parity	Oct 2017	The gas for the ModExp precompile overflows for large numbers	1	Low	EVMLab	✓ (15m)
12	Parity	Oct 2017	RETURNDATASIZE from a precompile returns a non-zero size	1	Low	EVMLab	✓ (2m)
13	Geth	Feb 2017	The EVM stack underflows for SWAP, DUP, and BALANCE	1	High	Unknown	✓ (6s)
14	Geth	Jan 2017	Undisclosed	-	High	Unknown	N/A
15	Geth	Nov 2016	Fails to revert the deletion of touched accounts on out of gas	1	High	Triggered-Mainnet	✓ (5m)

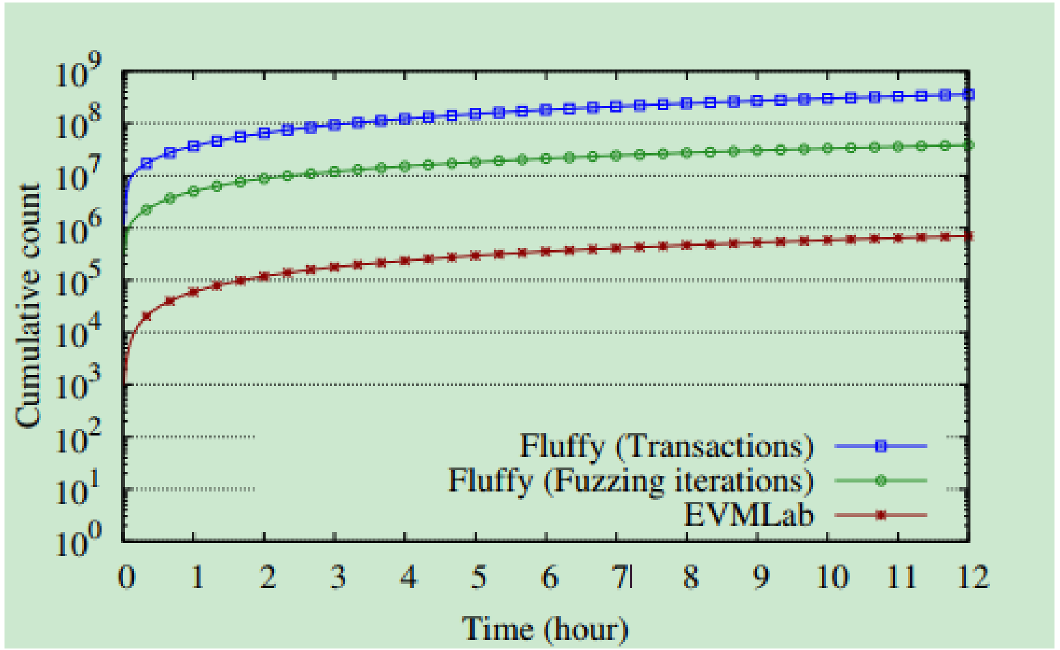
Out of 15 bugs, Fluffy finds 10 bugs within just 12 hours

Evalution performance

Fluffy: Our Fluffy implementation
Fluffy-Random-Bytecode: Modified Fluffy that randomly generates bytecode
EVMLab: A state-of-the-art fuzzer for Ethereum

Code coverage **(Higher is better)**





1.8X Random bytecode
2.7X EVMLab

510X Transactions
55X Iterations

Conclusion

- two consensus bugs.
- multi-transaction differential fuzzing technology.

THX