ContractFuzzer: Fuzzing Smart Contracts for Vulnerability Detection

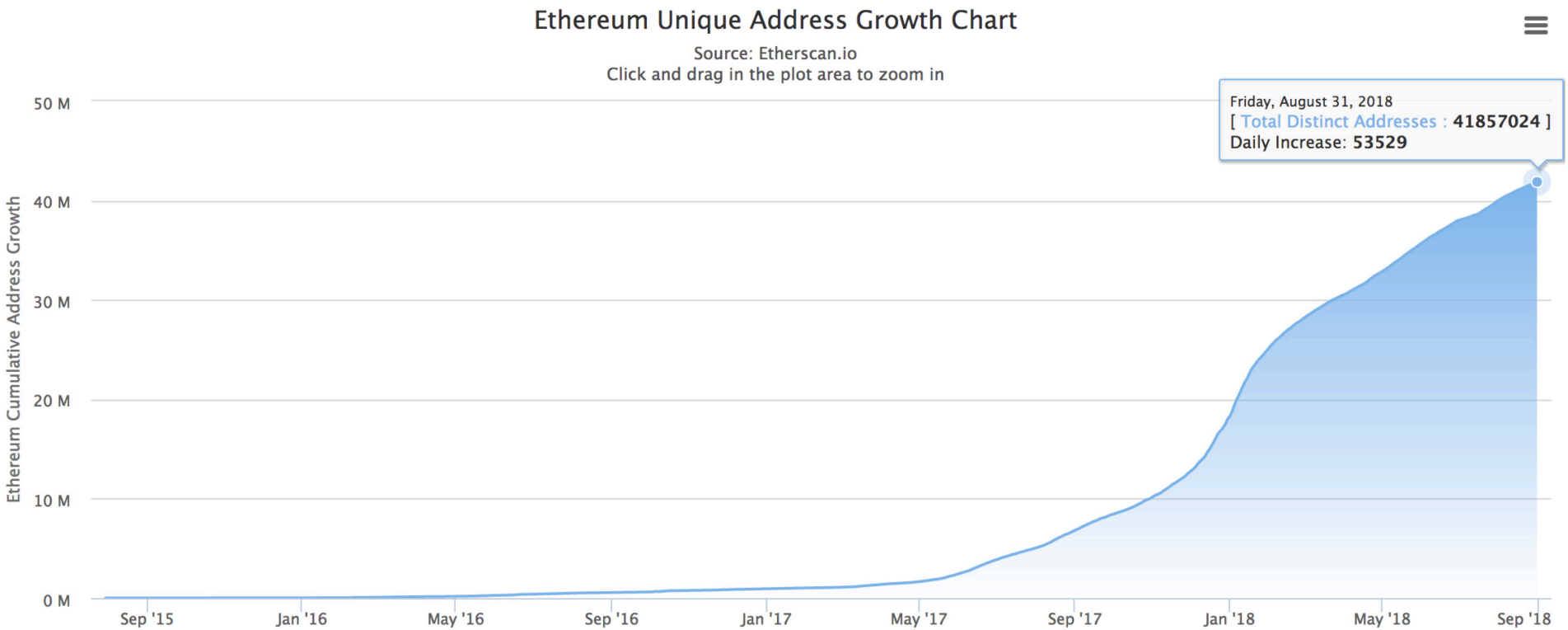
Authors: Bo Jiang*, Ye Liu*, and W.K. Chan†
*Beihang University
†City University of Hong Kong

Bo Jiang: gongbell@gmail.com

Contents

The Rise of Blockchain and Cryptocurrencies

The Rapid Growth of Ethereum Ecosystem



source: https://ethereum.stackexchange.com/questions/383/what-is-a-dapp

Smart Contracts with Verified Source Code

Etherscan
The Ethereum Block Explorer

LOGIN | Search by Address / Txhash / Block / Token / Ens | GO

HOME | BLOCKCHAIN | TOKENS | RESOURCES | MORE

Contracts With Verified Source Codes Only

Home / Verified Contracts

Sponsored: **Qravity Presale is Live** - Exclusive 30% Bonus until August 31. **Don't miss out!**

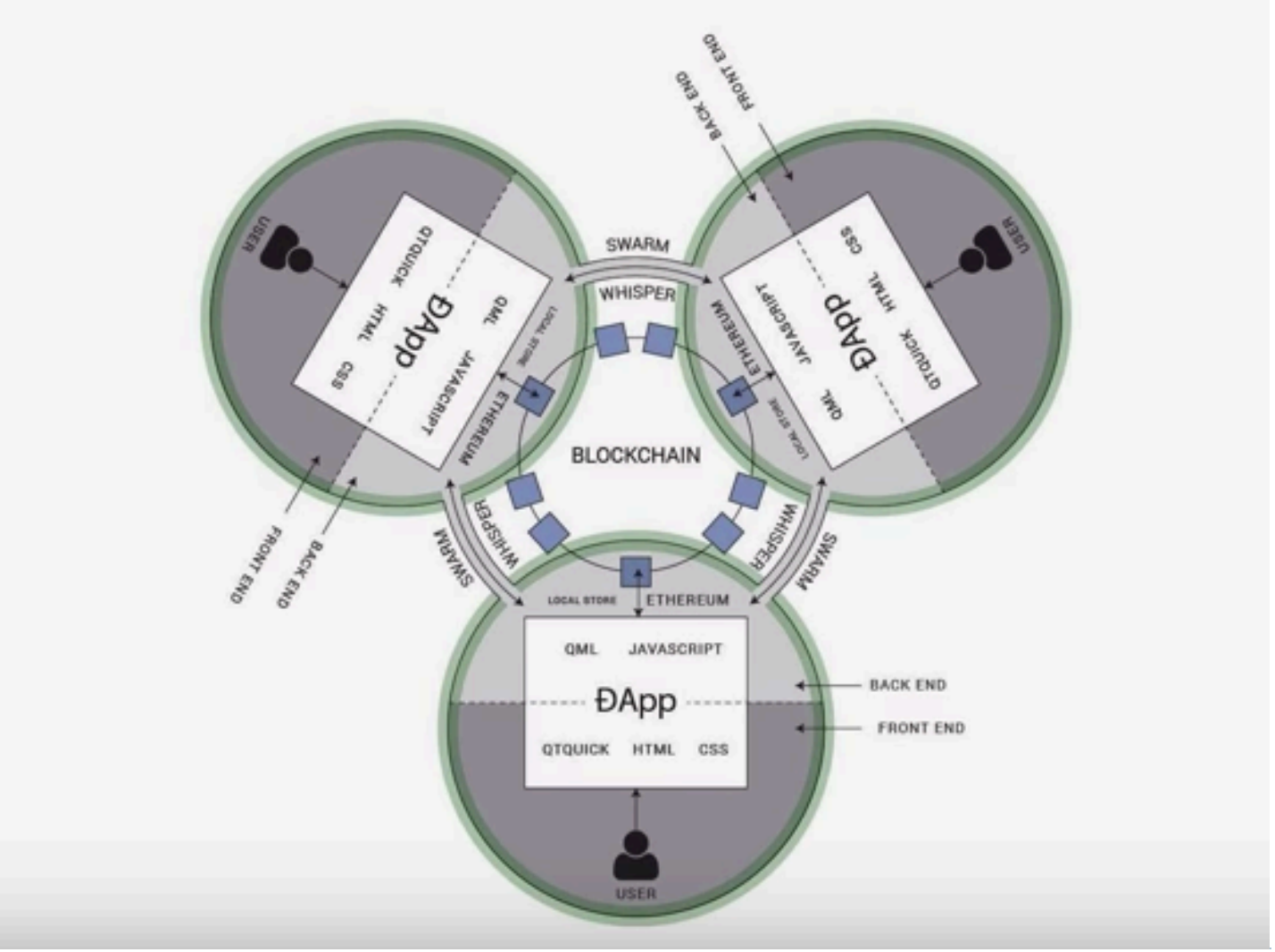
Search Contract Name | Find

A Total Of 42051 verified contract source codes found

First | Prev | Page **1** of **1683** | Next | Last

Address	ContractName	Compiler	Balance	TxCount	Settings	DateVerified
0x9ce8b0ebbf4f5b2...	EtheremonWorldNFT	v0.4.24	0 Ether	1		9/1/2018
0x70d2b63efb627a...	ZCCToken	v0.4.24	0 Ether	1	-	9/1/2018
0xEe74A44343dc25...	FILAToken	v0.4.21	0 Ether	3		9/1/2018
0x74bb7e3bddfea4...	FORZE	v0.4.24	0 Ether	1		9/1/2018
0xe8e8c2c4b65b7c...	BurnableERC20Token	v0.4.24	0 Ether	3		9/1/2018
0xfec8c5187aee4c2...	Sale	v0.4.24	0 Ether	2		9/1/2018
0x5eb4b436104680...	YumeriumManager	v0.4.24	0 Ether	2		9/1/2018
0x821285d692b585...	Morze	v0.4.24	0 Ether	5		9/1/2018
0x05a7a6bdc23dcf...	TCT	v0.4.12	0 Ether	30		9/1/2018
0x8b807547cd78e6...	CMR_Mining	v0.4.24	0 Ether	1		9/1/2018

Dapp = Frontend + Smart Contracts



source: https://ethereum.stackexchange.com/questions/383/what-is-a-dapp

Attacks on Smart Contracts

- Managing so much wealth with code will surely attract hackers…
- Attack on "The DAO bug"
  - $60 million US loss
- First round attack of parity wallet bug
  - $60 million US loss
- Second round attack of parity wallet bug
  - $150 million frozen in terms of Ether
- Attack on Integer Overflow Vulnerability
  - BEC coin on Ethereum withdrawn by attackers
  - Market value cleared

The Dangerous Delegate Call in Parity Wallet Contract

```
contract Wallet{
    function() payable { //fallback function
    if (msg.value > 0)
        Deposit(msg.sender, msg.value);
    else if (msg.data.length > 0)
        _walletLibrary.delegatecall(msg.data);
  }
}
contract WalletLibrary {
function initWallet(address[] _owners, uint _required, uint
_daylimit) {
      initDaylimit(_daylimit);
      initMultiowned(_owners, _required);
    }
}
```

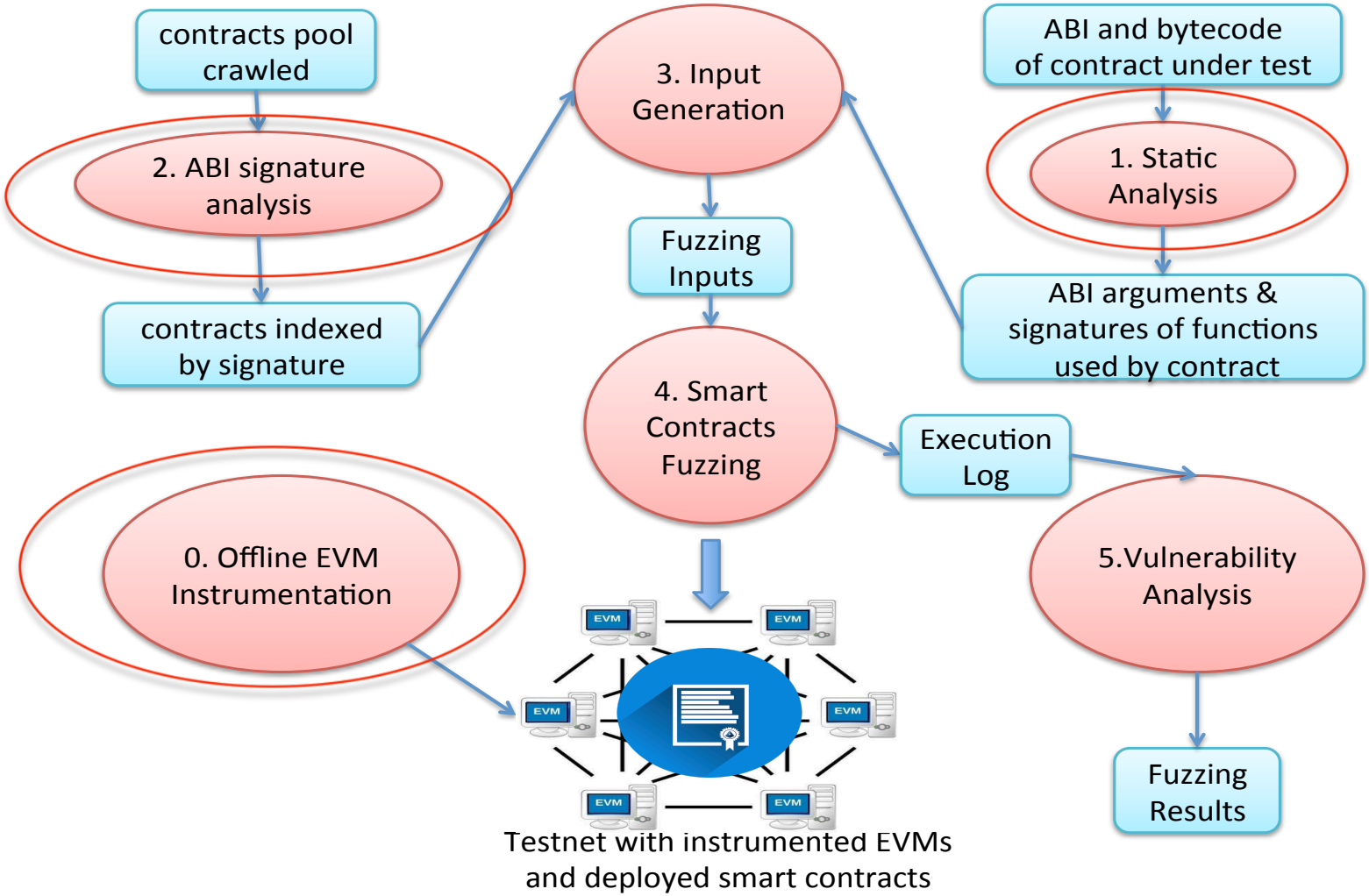
Contents

Typical Smart Contract Vulnerabilities

Vulnerability Type	Description
Gasless Send	send() triggers out of gas exception due to expensive fallback & sender keep ether wrongfully
Exception Disorder	Inconsistent error propagation of low-level calls
Reentrancy	Nonreentrant function invoked in reentrant manner
Timestamp Dependency	Relying timestamp to decide ether transfer
Block Number Dependency	Relying Block Number to decide ether transfer
Dangerous DelegateCall	The argument of delegatecall can be provide by msg.data
Freezing Ether	Smart contracts can receive ether but cannot send ether except through delegatecall

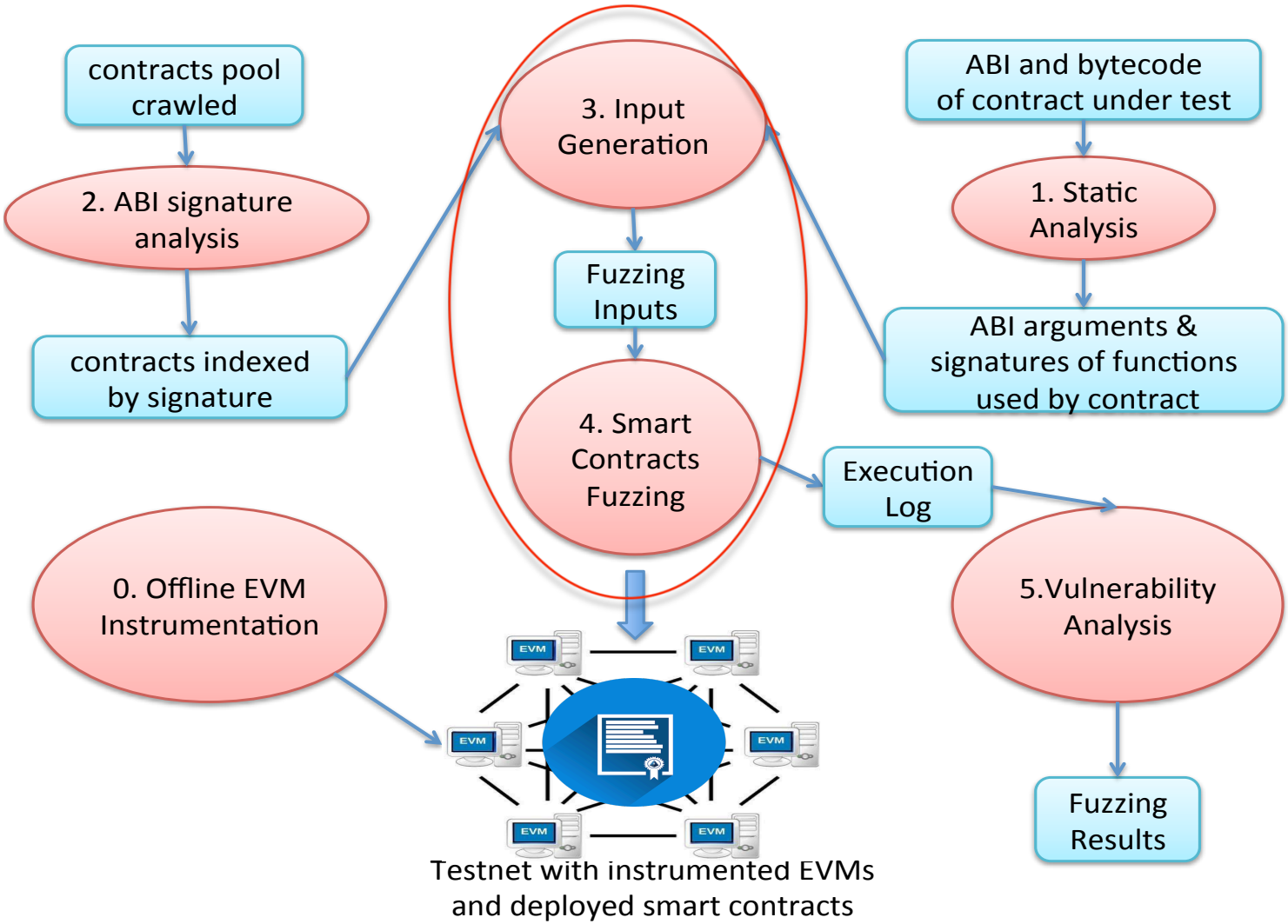
Contents

Overview of ContractFuzzer



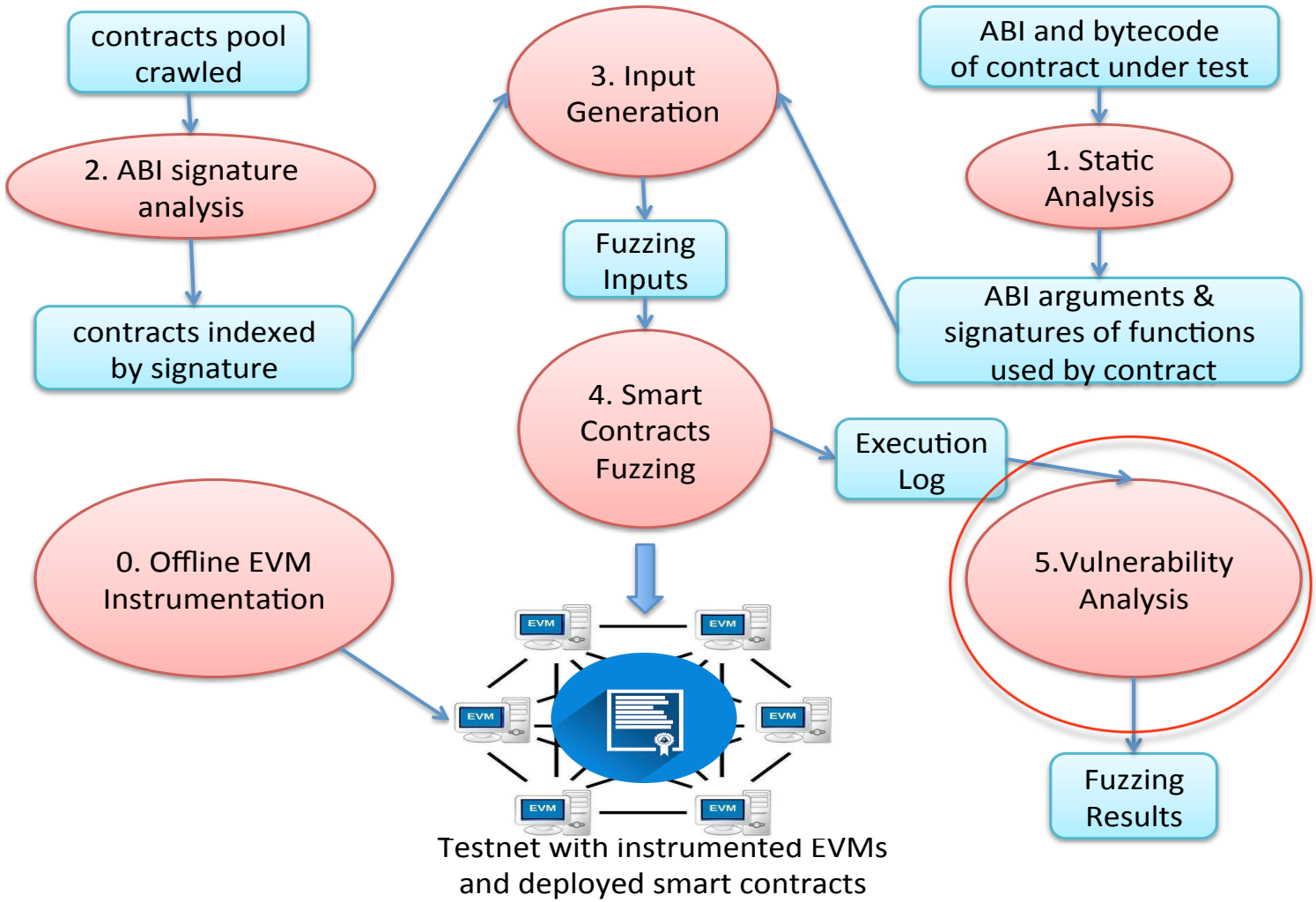
Open Sourced: https://github.com/gongbell/ContractFuzzer

Overview of ContractFuzzer



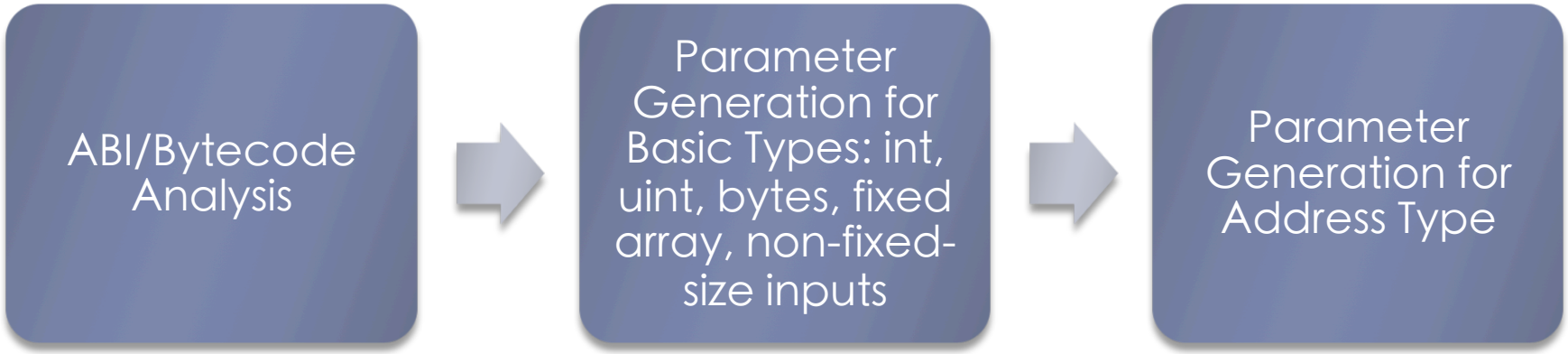
Open Sourced: https://github.com/gongbell/ContractFuzzer

Overview of ContractFuzzer

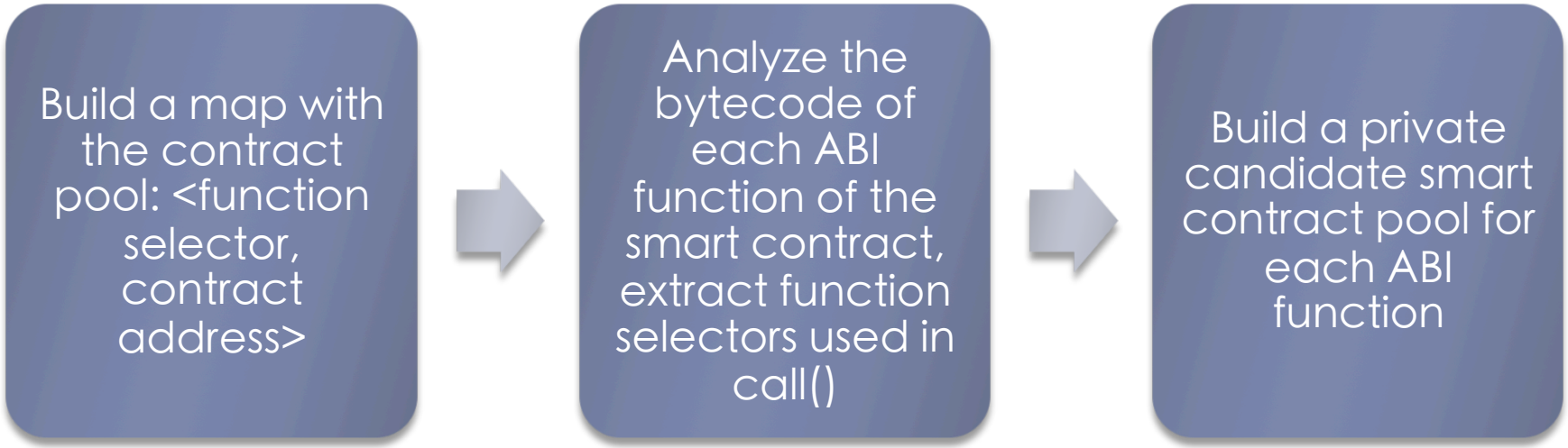


Open Sourced: https://github.com/gongbell/ContractFuzzer

Input Generation for ABI Signature



How to generate inputs for address type?



Input Generation for Reentrancy Vulnerability

```
contract BountyHunt{
…
function claimBounty() preventTheft {
  uint balance = bountyAmount[msg.sender];
  if (msg.sender.call.value(balance)()) {
      totalBountyAmount -= balance;
      bountyAmount[msg.sender] = 0;
    }
  }
}
contract AttackerAgent{
  ①  …                                  ②                    ③
    function AgentCall(address contract_addr,bytes msg_data){
      call_contract_addr  = contract_addr;
      call_msg_data = msg_data;
      contract_addr.call(msg_data);
    }
    function() payable{
      call_contract_addr.call(call_msg_data);
    }
}
```

Test Oracle Design and EVM Instrumentation

Vulnerability Type	Description
Gasless Send	*send()* call returns *ErrOutOfGas* during execution
Exception Disorder	For a chain of nested calls: the root call doesn't throw exception while at least one of its nested calls throws exception
Reentrancy	*ReentrancyCall* $\wedge$ *CallAgentWithValue*
Timestamp Dependency	*TimestampOp* $\wedge$ *(SendCall* $\vee$ *EtherTransfer)*
Block Number Dependency	*BlockNumOp* $\wedge$ *(SendCall* $\vee$ *EtherTransfer)*
Dangerous DelegateCall	*delegateCall* on function extracted from input
Freezing Ether	balance > 0 $\wedge$ no *transfer/send/call/suicide*

Contents

Experiment Setup

- Subject Programs: 6991 out of 9960 smart contracts
- Environment: Two dockers, one for contract fuzzer tool and one for testnet with geth client version 1.7.0
- Procedure:
  - 1. generate 6*k calls for each ABI function
    - 3 types of account: creator account, external account, *AttackAgent*
    - 2 call.value(): with&without value transfer
  - 2. merge all calls into a pool for a smart contract
  - 3. Randomly select calls from pool for fuzzing
  - 4. Analyzing and report results

Summary of Vulnerabilities Detected

Vulnerability Type	Number of Vulnerabilities	Percentage	True Positive Rate
Gasless Send	138	2.06%	100%
Exception Disorder	36	0.54%	100%
Reentrancy	14	0.21%	100%
Timestamp Dependency	152	2.27%	96.05%
Block Number Dependency	82	1.23%	96.34%
Dangerous Delegatecall	7	0.10%	100%
Freezing Ether	30	0.45%	100%
Total	459	/	/

Comparison of ContractFuzzer and Oyente

Vulnerability Type	ContractFuzzer			Oyente		
Statistics	No.	FP	FN	No.	FP	FN
Timestamp Dependency	152	6	95	273	70	44
Reentrancy	14	0	1	43	28	0

A False Negative Case of ContractFuzzer

```
contract BDSM_Crowdsale {
    ...
    uint public startICO_20_December = 1513728060; //2017.12.20
    uint public stopICO_20_March = 1521504060; //2018.3.20
    function () payable {
        if (now < startICO_20_December) {
            msg.sender.transfer(msg.value);
        }
    ...
```

A False Negative Case of Oyente

```
function buy(uint8 _bomb) public payable {
...
int _random =
uint(keccak256(block.blockhash(block.number-
1),msg.gas,tx.gasprice,block.timestamp))%bomb.chance + 1;
if(_random == 1){
    bomb.owner.transfer(...)
    ceoAddress.transfer(...)
}}
```

Cryptographic functions hard for SMT solver

Case Studies-ICO Wrongfully Holding Ether

```
contract CrowdSalePreICO {
    ...
 function() payable stopInEmergency onlyStarted notFinalized{
    ...
    uint contribution = msg.value;
if (safeAdd(totalDepositedEthers, msg.value) >hardCapAmount)
{
contribution = safeSub(hardCapAmount, totalDepositedEthers);
}
uint excess = safeSub(msg.value, contribution);
    if (excess > 0){
        msg.sender.send(excess);
}}}
```

Send may fail, error is hidden due to exception disorder, but send does not check for error

Case Studies – Timestamp Dependency

```
contract SlotMachine {
    ...
function(){
    uint nr = now; //now is the block timestamp
    uint y = nr & 3;
    ...
if(y==1) { wins[1]++; win = (msg.value * 2)  + (msg.value / 2);}
    earnings += int(msg.value);
if(win > 0) {
    bool res = msg.sender.send(win);
    earnings -= int(win);
} } }
```

Case Studies – The Wallet Smart Contracts Frozen by Parity Bug

```
contract Wallet is WalletEvents {
  ...
  function() payable {
    if (msg.value > 0)
        Deposit(msg.sender, msg.value);
    else if (msg.data.length > 0)
      _walletLibrary.delegatecall(msg.data);
    }
  function hasConfirmed(bytes32 _operation, address _owner)...{
      return _walletLibrary.delegatecall(msg.data);
     }
   address constant _walletLibrary =
0xcafecafecafecafecafecafecafecafecafecafecafe;
}
```

Contents

Related Work

- Oyente
  - Luu et al.@ ACM CCS, 2016.
  - Symbolic execution based verification
- MAIAN
  - Nikolic et al.@arxiv.org
  - Symbolic execution tool on execution traces
- Echidna Fuzzer
  - https://github.com/trailofbits/echidna
  - Fuzzing based on Unit testing framework
  - No support for security testing of smart contracts

Contents

Conclusion

- In this work, we proposed ContractFuzzer, the first comprehensive fuzzing framework to detect smart contract vulnerabilities.
- We proposes the test oracles for detecting 7 typical smart contract security vulnerabilities.
- Our experiment with 6991 real world smart contracts shows ContractFuzzer are effective to detect security vulnerabilities with very high precision.
- In total, 459 vulnerabilities detected during Fuzzing.

Open Sourced: https://github.com/gongbell/ContractFuzzer

Future Work

- Explore more effective input generation strategies
- Evaluate the tool in terms of coverage
- Reduce false negatives

Thank you!

Thank you!

Thank you!