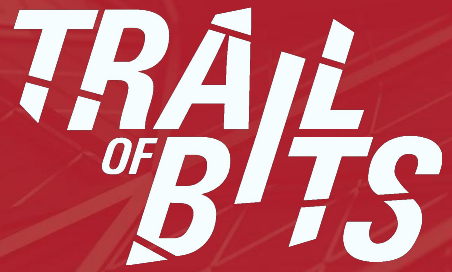


Introduction to Smart Contracts Vulnerabilities

GreHack 2018

Who Am I?

- **Josselin Feist, josselin@trailofbits.com**

- **Trail of Bits: trailofbits.com**
  - We help organizations build safer software
  - R&D focused: we use the latest program analysis techniques

Goals

- What is a Blockchain?
- What is a smart contract?
- How to break it?
- Hands on

Before Starting

- https://github.com/trailofbits/grehack18

The Ethereum Blockchain

TRAIL OF BITS

Blockchain

- Ledger: Growling list of records

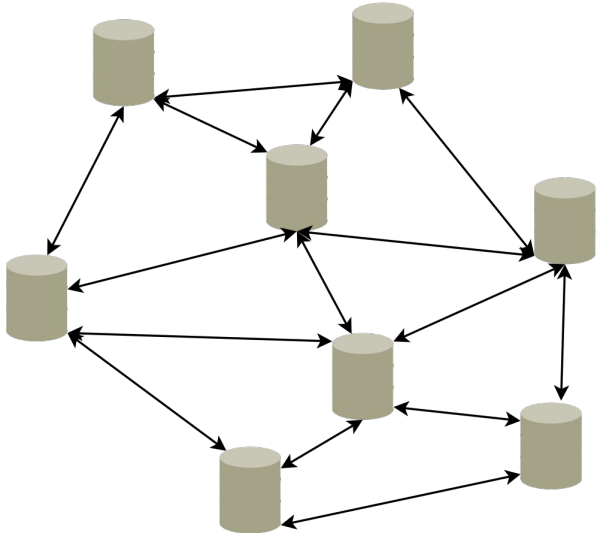
Alice + 100;

Alice - 50; Bob + 50;

...

Blockchain

- Distributed ledger: All participants store all the data
- Decentralized consensus: Everyone agrees on the data



Blockchain Application

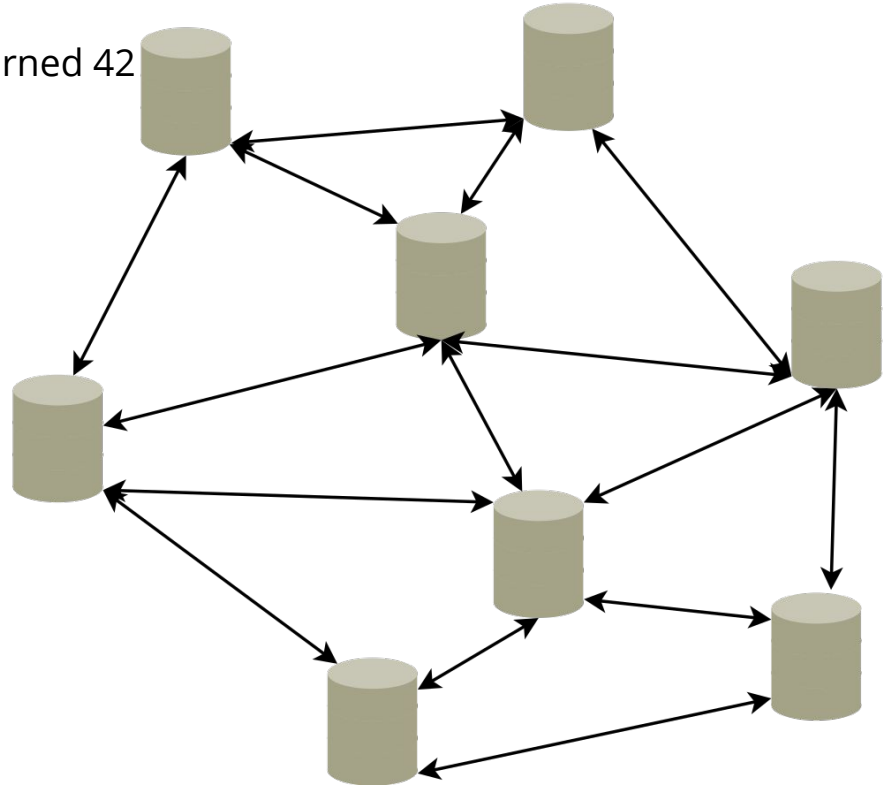
- **Bitcoin (2009): First digital currency using blockchain**
  - Solved the double spending problem

- **Ethereum (2015): Extended blockchain to run apps**
  - Store & execute code

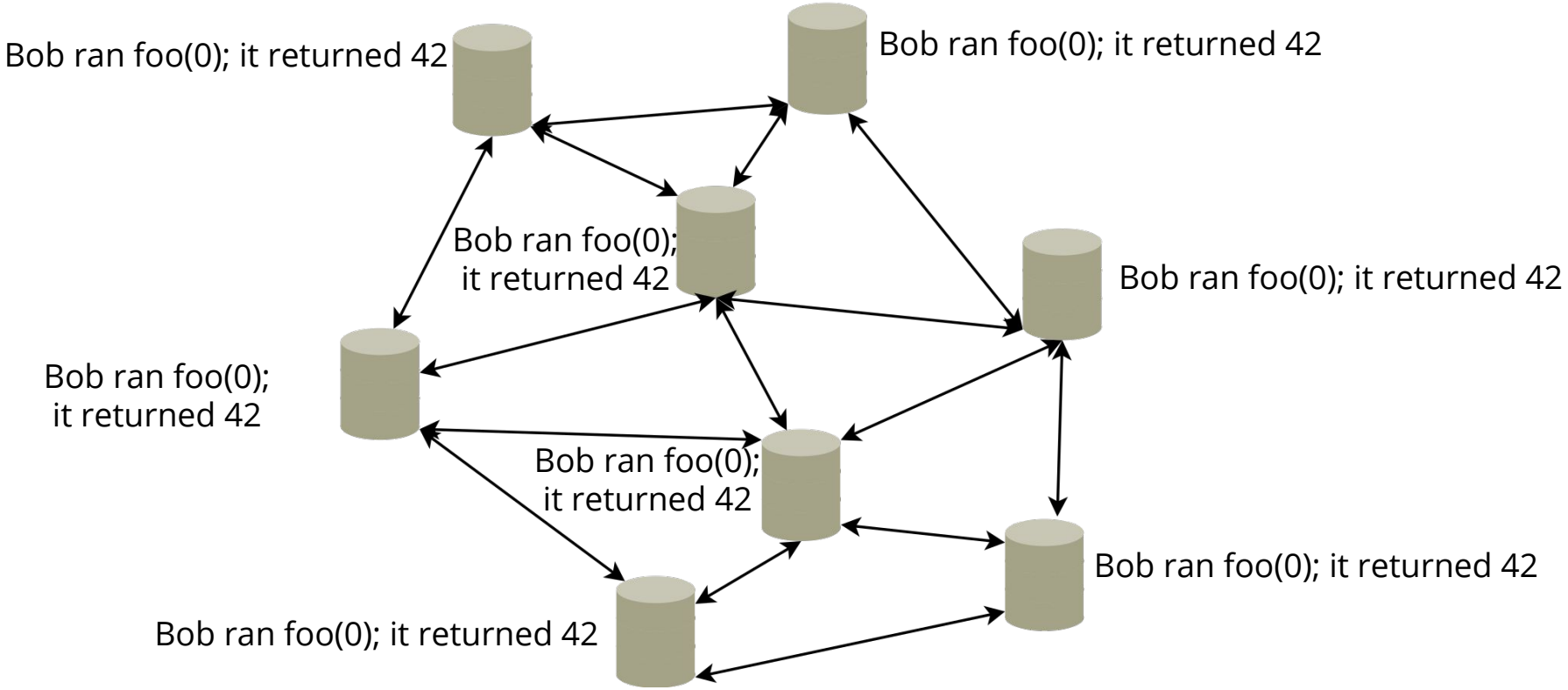
Bitcoin: distributed database => Ethereum: distributed VM

Decentralized Application

Bob ran foo(0); it returned 42



Decentralized Application



Smart Contracts

- **Smart Contracts: Applications that run on Ethereum**
  - Everyone executes and verifies it
  - Decentralized: nobody can stop or secretly modify data
  - => Ensures strong properties on your application

Smart Contract Usage

- **Digital currency is one example of an application**
  - ICO, Crowdfunding system
  - Game (ex: Poker, lotteries, ..)
  - ...

- **Already a lot of money invested into smart contracts**
  - ~ $6 billion raised in 2017 by ICOs
  - ~ $7 billion in 2018

Ethereum Internals

TRAIL OF BITS

Smart Contract Usage

- **Ethereum runs EVM bytecode**
  - VM with <150 opcodes
  - 1 register (PC)
  - Stack-based

```
00000000  PUSH1   0x60
00000002  PUSH1   0x40
00000004  MSTORE
00000005  CALLDATASIZE
00000006  ISZERO
00000007  PUSH2   0x131
0000000a  JUMPI
```

- **Calling a function = making a transaction**
  - It has a cost: gas, paid in ethers
- **Bytecode cannot be updated (!)**

Solidity

- **Smart contracts are typically written in Solidity**
  - High-level language in “Javascript style”
  - Contracts organized as a set of methods
  - State = contract variables + balance (# ethers)

Solidity: Example

```
pragma solidity 0.4.24; // Compiler version
contract Bank{                          // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    constructor(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.24; // Compiler version
contract Bank{                          // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    constructor(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.24; // Compiler version
contract Bank{                          // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    constructor(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.24; // Compiler version
contract Bank{                          // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    constructor(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.24; // Compiler version
contract Bank{                          // There are bugs, don't use this contract
    mapping(address => uint) private balances;

     constructor(uint initial_supply) public {
        balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.24; // Compiler version
contract Bank{                          // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    constructor(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

State variable

Constructor

Public function

Constant function (gas-free)

Ethereum Transaction

- **From/To: caller/destination**
- **Data: Function name and parameters**
  - Function name: 4 bytes of `keccak256(signature)`
    - Ex: 'transfer(address,uint256)' => 0xa9059cbb
  - Parameters can be padded with 0 bytes according the size

transfer(0x41414141, 0x42) =

0xa9059cbb000000000000000000000000000000000000000000000000000000000041414141000000000000000000000000000000000000000000000000000000000000000042

Demo

- Geth transaction
- Ethersplay

Smart Contracts Vulnerabilities

TRAIL OF BITS

Vulnerabilities Impact

- **Vulnerabilities in smart contracts have already cost a lot**

- **Parity Wallet (2017)**
  - $30 million (could have been a lot worse)
- **DAO Hack (2016)**
  - $150 million
  - Hard fork

Smart Contract Vulnerabilities

- **“Classic” vulnerabilities:**
  - Integer overflow/underflow
  - Race condition

- **Logic vulnerabilities / errors in the design**
  - Harder to find, but deadly

Improperly restricted functions

- Parity Wallet
  - Widely used library for storing ethers
  - Built by Gavin Wood, formerly CTO of Ethereum Foundation

- **Key function was public instead of private**
  - Anyone can become the owner of the contract

Reentrancy

- The DAO ($$$)

```
if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
    throw;
}

userBalance[msg.sender] = 0;
```

- **Use of the fallback function to call the caller**
  - Call the fallback function of the malicious contract
  - The fallback function calls a second time the original contract
  - Repeat n times => withdraws n times the original deposit

Logic vulnerabilities are hard to find

- **What is a vulnerability in a contract?**
  - It depends on the contract purpose!

- **A user ends with more ethers than invested, is it a bug?**
  - Yes, if the contract is a paid service
  - No, if the contract is a lottery

Hands on

Hands-on

- **/home/grehack/Desktop/grehack18/exercises.pdf**
  - Or https://github.com/trailofbits/grehack18/blob/master/exercises.pdf

Exercise 1 Solution

Hands-on

- **/home/grehack/Desktop/grehack18/exercises.pdf**
  - Or https://github.com/trailofbits/grehack18/blob/master/exercises.pdf

Exercise 2 Solution

Exercise 3 Solution

Exercise 4 Solution

Workshop Conclusion

Conclusion

- **Smart contracts is a new technology**
  - Already a lot of money = good target for attackers
  - Developer tooling is not mature
- **We will see other large hacks in a near future**
- **Automating bugs finders:**
  - Slither: https://github.com/trailofbits/slither/
  - Echidna: https://github.com/trailofbits/echidna/
  - Manticore: https://github.com/trailofbits/manticore/

How to Learn More?

https://github.com/trailofbits/awesome-ethereum-security

- List of Ethereum security references

https://github.com/trailofbits/not-so-smart-contracts

- Examples of real world vulnerabilities