Blockchain is secure

The End
But really, it isn’t

- Blockchain offers a secure infrastructure
- How you use it is your problem
- There’s no central authority backing you
  - It’s just you (and sometime the community)
- Billions ($) lost in about 5 years
Basic Transactions

Let’s get started with some basic activity on Ethereum
Basic Transaction

Alice
ACCOUNT A
100
BALANCE

Transaction
VALUE 10
GAS FEE 0.01

Bob
ACCOUNT B
0
BALANCE
Basic Transaction

ETH TRANSACTION

Alice
ACCOUNT A
89.99
BALANCE

Bob
ACCOUNT B
10
BALANCE

Transaction
VALUE 10
GAS FEE 0.01
Basic Transaction

- Signed by the sender
- Both sender and receiver are identified using public keys
- Ether balance is maintained by platform
- Ether is used to pay for transactions
ETH TRANSACTION

Token Transfer

Alice
ACCOUNT A
89.99 USDT
BALANCE

Transaction
TRANSFER (A, B, 90)
VALUE 0
GAS FEE 0.01

ACCOUNTS

BALANCE

ACCOUNT A 100.00
ACCOUNT B 0.00
ETH TRANSACTION

Token Transfer

Alice
ACCOUNT A
89.98 ETH
BALANCE

Transaction
TRANSFER (A, B, 90)
VALUE 0
GAS FEE 0.01

ACCOUNTS
BALANCE
ACCOUNT A 10.00
ACCOUNT B 90.00

USDT
TOKEN TRACKER
Token Transfer

- Contracts are used to power non-native currencies and assets
- A contract tracks the balances of each account (using their public keys as identifiers/addresses)
- The tracker is implemented in code, executes as part of the consensus mechanism
- Code will always execute exactly as intended written.
- Data cannot be changed except through mechanisms implemented in the contract
Some attacks

Looking at some malicious activity on Ethereum
Self Transfer Attack

Transaction

Transfer(address from, address to, uint256 value) {
    // Validation checks...
    uint256 balancesFrom = balances[from];
    uint256 balancesTo = balances[to];
    uint256 balancesFromNew = balancesFrom.sub(value);
    balances[from] = balancesFromNew;
    uint256 balancesToNew = balancesTo.add(value);
    balances[to] = balancesToNew;
}

Simulator
https://etherscan.io/tx/0x260f3e2d54cefbeb06979d75a83d27f20424ed3dbb4397f5a68916c4e1f513

Token Tracker
0X32E4C68B3A4A813B710595AEBA7F6B204AB9C15

ACCOUNTS (HOLDERS ADDRESSES) BALANCES
ACCOUNT A 200.00
ACCOUNT B 0.00

Attacker
ACCOUNT A

Transaction
TRANSFER (A, A, 100)
VALUE 0
GAS FEE 0.01

Token Tracker
0X...9C15

balancesFrom 100.00
balancesTo 100.00
balancesFromNew 0.00
balancesToNew 200.00
Batch Transfer Attack

function batchTransfer(address[] receivers, uint256 value) {
    uint cnt = receivers.length;
    uint256 amount = uint256(cnt) * value;
    require(cnt > 0 && cnt <= 20);
    require(value > 0 && balances[msg.sender] >= amount);
    balances[msg.sender] = balances[msg.sender].sub(amount);
    for (uint i = 0; i < cnt; i++) {
        balances[receivers[i]] = balances[receivers[i]].add(value);
    }
}

Transaction
TRANSFER ([B,C] 0x800000)
VALUE 0
GAS FEE 0.01

attacker ACCOUNT A

Token Tracker
0x793D

Account Tracker

| ACCOUNT A | 0.00 |
| ACCOUNT B | 0x800000 |
| ACCOUNT C | 0x800000 |
| ACCOUNT D | 0.00 |
Self/Batch Transfer Attack

- Error in the contract violates intended business rules
- Contract is final, no real way to recover

- Can be (easily) identified
  - But also by attackers
RedeemGift(uint256 value, uint256 data, byte256 secret) {
    uint256 hash = SHA3(ENCODE(
        value, data, secret));
    if (expectedHashes.contains(hash) &&
        usedHashes[hash] == false) {
        msg.sender.transfer(value);
        usedHashes[expectedHash] = true;
    }
}

**Simulator**

**Transactions Pool**

Alice  
**ACCOUNT A1**

Transaction
REDEEMGIFT (100, SECRET, DATA)
VALUE 0  
GAS FEE 0.01

Miner

Attacker  
**ACCOUNT A2**

Transaction
REDEEMGIFT (100, SECRET, DATA)
VALUE 0  
GAS FEE 0.10

Contract

Pending Transactions Pool

<table>
<thead>
<tr>
<th>ACCOUNTS</th>
<th>ACTION</th>
<th>GAS PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT A1</td>
<td>REDEEMGIFT</td>
<td>0.01</td>
</tr>
<tr>
<td>ACCOUNT A2</td>
<td>REDEEMGIFT</td>
<td>0.10</td>
</tr>
</tbody>
</table>

**MINER**

**ACCOUNT A1**

100 ETH

**ACCOUNT A2**

Simulator
Frontrunning

- A normal part of how a blockchain operates
  - Visibility comes with its costs
- Not necessarily “illegal”, mostly unethical
- Need to be very careful with “secrets” in the blockchain
- Instruments in place to facilitate recovery
- Ability to update/upgrade the contract
Lack of Access Control

Parity Multi-Sig Wallet
0X...07B4

Create Contract

Deploy
Lack of Access Control

Attacker
ACCOUNT A

Transaction

INITIALIZE(ACCOUNT A)
VALUE 0
GAS FEE 0.01

Parity Multi-Sig Wallet
0X...07B4

Deployer forgot to initialize the contract
The contract has no owner

Self Destroyed
Parity multi-sig wallet

- Deployment/configuration issue
- Single point of failure for DoS
Stolen Private Key

Attacker
ACCOUNT A

Transaction
VALUE 8,709.0332
GAS FEE 0.01

KuCoin Wallets Balances (Account K)

0x2B5634C42055806A59E9107ED44D43C426E585258

$19,554,043.98
8,709.0332 Ether

Hack to the web server of KuCoin and stole the Private Key
Stolen Private Key

- Basically, allows anyone to impersonate the owner of the key
- Can usually easily transfer assets to another account, owned only by the attacker

- Protect your keys
- Recovery
Advanced Transactions

As things get more complex, they become more interesting
ETH TRANSACTION
Token Swap

Alice
ACCOUNT A

89.98
BALANCE

Transaction
SWAP (USDT, BNB, 100)
VALUE 0
GAS FEE 0.03

Uniswap

Transaction
TRANSFER (A, U, 100)
VALUE 0
GAS FEE 0.01

Uniswap

Token Swap

ACCOUNTS
BALANCE

ACCOUNT A
100.00

ACCOUNT U
1.2K

USDT
TOKEN TRACKER
ETH TRANSACTION

Token Swap

Alice
ACCOUNT A
BALANCE 89.98

Uniswap

Transaction
SWAP (USDT, BNB, 100)
VALUE 0
GAS FEE 0.03

Transaction
TRANSFER (A, U, 100)
VALUE 0
GAS FEE 0.01

ACCOUNT A
BALANCE 0.00

ACCOUNT U
BALANCE 1.3K

USDT
TOKEN TRACKER
ETH TRANSACTION

Token Swap

Transaction
SWAP (USDT, BNB, 100)
VALUE 0
GAS FEE 0.03

Transaction
TRANSFER (A, U, 300)
VALUE 0
GAS FEE 0.01

Transaction
TRANSFER (A, U, 100)
VALUE 0
GAS FEE 0.01

Alice
ACCOUNT A
89.98
BALANCE

Uniswap

USDT
TOKEN TRACKER

ACCOUNTS
BALANCE
ACCOUNT A 0.00
ACCOUNT U 1.3K

BNB
TOKEN TRACKER

ACCOUNTS
BALANCE
ACCOUNT A 0.00
ACCOUNT U 3.4K

Transaction
TRANSFER (U, A, 100)
VALUE 0
GAS FEE 0.01

Transaction
SWAP (USDT, BNB, 100)
VALUE 0
GAS FEE 0.03

Transaction
TRANSFER (U, A, 300)
VALUE 0
GAS FEE 0.01
ETH TRANSACTION

Token Swap

Alice
ACCOUNT A
89.95
BALANCE

Transaction
SWAP (USDT, BNB, 100)
VALUE 0
GAS FEE 0.03

Uniswap

Transaction
TRANSFER (U, A, 100)
VALUE 0
GAS FEE 0.01

Transaction
TRANSFER (A, U, 100)
VALUE 0
GAS FEE 0.01

ACCOUNTS
BALANCE

ACCOUNT A
0.00

ACCOUNT U
1.3K

USDT
TOKEN TRACKER

ACCOUNTS
BALANCE

ACCOUNT A
300.00

ACCOUNT U
3.3K

BNB
TOKEN TRACKER

Transaction
TRANSFER (A, U, 300)
VALUE 0
GAS FEE 0.01

Transaction
TRANSFER (U, A, 100)
VALUE 0
GAS FEE 0.03
Token Swap

- Automated code that does swap pricing
- Susceptible to manipulation in various ways
ETH TRANSACTION

Flash Loan

Alice
ACCOUNT A

ACCOUNT A
BALANCE
89.95

Transaction
LOAN (USDT, 100)
VALUE
0
GAS FEE
0.04

SMART CONTRACT
ALICE

Borrow

Aave
AAVE

FLASHLOAN

Transfer

Defi Trader
TOKEN TRACKER

ACCOUNTS
BALANCE
AAVE
1,000
SMART CONTRACT

PAYOUT
Receive
Withdraw

WITHDRAW
Check Balance
payOut
Update Balance

USDT
TOKEN TRACKER

ACCOUNTS
BALANCE
SMART CONTRACT

payOut
Update Balance

Check Balance

Withdraw

payOut
Update Balance

withdraw

Check Balance

Flash Loan

- Access to nearly unlimited funds for the scope of a single transaction
- Very safe for the loaner
- Relatively low interest rates
- Can be used to multiply attack impact significantly
- Quite unique to blockchain
Some more attacks

Looking at some more malicious activity on Ethereum
Reentrancy

function withdrawRewardFor(address _account) noEther internal returns (bool _success) {
    if ((balanceOf(_account) *
        rewardAccount.accumulatedInput()) /
        totalSupply < paidOut[_account]) {
        throw;
    }
    uint reward = (balanceOf(_account) *
        rewardAccount.accumulatedInput()) /
        totalSupply - paidOut[_account];
    if (!rewardAccount.payOut(_account,
        reward)) {
        throw;
    }
    paidOut[_account] += reward;
    return true;
}

Simulator - The DAO Contract

https://etherscan.io/tx/0x0ec3f2488a93839524add10ea229e773f6bc891b4eb4794c3337d4495263790b

Simulator - Attacker Contract

function receive() public payable {
    address theDao = 0x...9B2D;
    theDao.withdrawRewardFor(address(this));
}
Reentrancy

function deposit(address protocol, address[] memory tokens, uint256[] memory dnAmounts) {
// ...
uint256 nBalanceBefore = distributeYieldInternal(protocol);
depositToProtocol(protocol, tokens, dnAmounts);
uint256 nBalanceAfter = updateProtocolBalance(protocol);
// ...
}

function receive() public payable {
address akropolis = 0x…9B2D;
akropolis.deposit(address(this));
}

Simulator - AKROPOLIS
https://etherscan.io/tx/0xe1f375a47172b5612d96496a4599247049f07c9a7d5f832f8be296b0c281e04d

Simulator - Attacker Contract

function receive() public payable {
address akropolis = 0x…9B2D;
akropolis.deposit(address(this));
}
Reentrancy

- Relatively easy to detect when it occurs
- Difficult to detect ahead of time
- Easy to mitigate at the cost of flexibility
  - Complexity <> Simplicity is a tradeoff for Flexibility <> Security
Oracle Manipulation

- Flashloan: 100 ETH
- Tarde 1: 50 ETH → 2K BAT
- Tarde 2: 50 ETH → 1K BAT
- Supply BZX: 500 BAT
- Loan on BZX: 25 ETH
- Reverse Trade 2: 50 ETH → 2K BAT
- Reverse Trade 1: 50 ETH → 2K BAT
- Repay Flashloan: 101 ETH
Oracle Manipulation

- Manipulation can sometimes be difficult to detect
- Legality is unknown, morality is clear
Key Takeaways

• Be prepared
  • Act to eliminate vulnerabilities
  • Reduce attack surface