Breaking Ethereum 😈

Lessons learnt from broken contracts

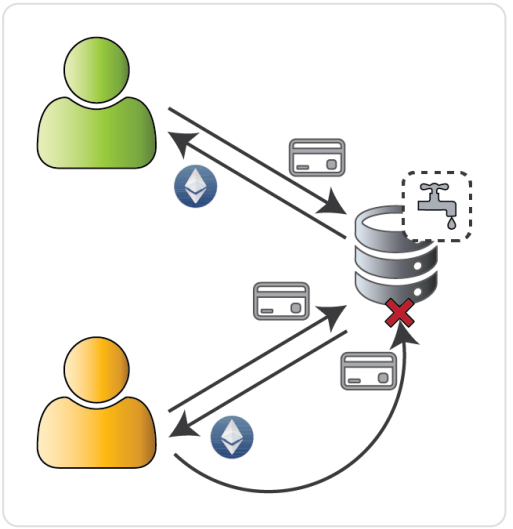
10 September 2016
Thomson Reuters Hackethon

Péter Szilágyi
Ethereum Core Developer

If you don't know the chain dynamics...

Ether faucet – 0x793ae8c1b1a160bfc07bfb0d04f85eab1a71f4f2

```
contract Faucet {
    uint amount = 0.01 ether;
    uint freq   = 5760;

    mapping (address => uint) prev;

    function request() {
        if (address(this).balance < amount) {
            return;
        }
        if(block.number < prev[msg.sender] + freq) {
            return;
        }
        msg.sender.send(amount);
        prev[msg.sender] = block.number;
    }
}
```

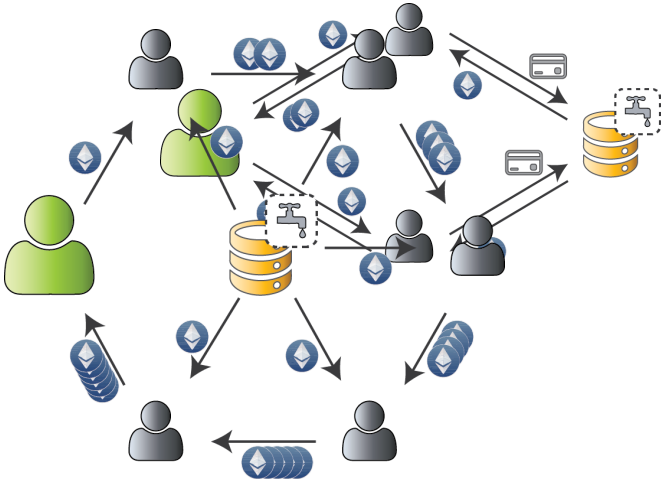


Give away 0.01 Ether to anyone, once per 24 hours... what could go wrong? 😇

Ether faucet – pwned

Payout (0.01 Ether) is a nice amount

- 4.1x withdrawals (48748 gas * 50 gwei)
- 9.5x transactions (21000 gas * 50 gwei)



Faucet security ⇔ Account uniqueness

- 24h restriction applies per account
- No global withdrawal throttling

Lesson: Accounts are free, instantaneous and infinite!

Roulette – 0x5fe5b7546d1628f7348b023a0393de1fc8

Sizeable implementation (https://github.com/retotrinkler/solidity1/blob/master/alpha/roulette.sol) of a rou

- Users bet on various outcomes of a "spin"
- After each bet the contract spins the wheel

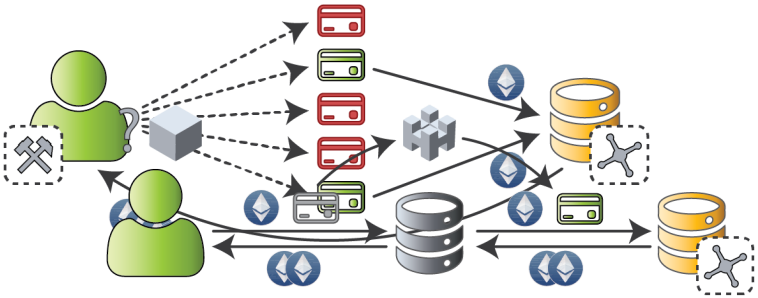
```
contract Roulette {
    uint seed = 1;

    function rand() private returns (uint) {
        seed = ((seed*3 + 1)/2 % 10**9);

        return (seed + block.number + block.difficulty + block.timestamp + block.gaslimit) % 37;
    }
    [...]
}
```

Uses an onchain random number generator... what could go wrong? 😇

Roulette – pwned

Miners make the chain

- Block parameters are defined by miners
- Included transactions are chosen by miners



Transactions are aware of the chain

- Block parameters are shared between them

Lesson: Blockchain state is free for all to use and abuse!

- Contracts decide runtime how to invoke others

Etherdice – 0x2faa316fc4624ec39adc2ef7b5301124cfb68777

Fairly involved (https://etherdice.io/#contract) dice game

- Owner seeds round with hidden number
- Players bet on outcomes with own numbers
- Owner reveals the number, evaluating the round

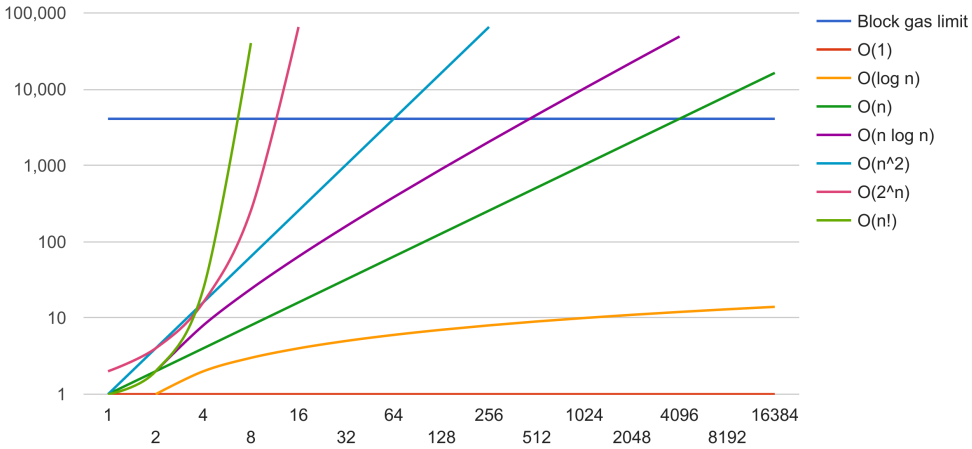
```
contract Dice {
    function evaluate(bytes32 seed) {
        // [...] verify the seed
        for (uint i = 0; i < bets.length; i++) {
            // [...] evaluate bets and pay winners
        }
    }
    [...]
}
```

Iterate over all accumulated bets in one go... what could go wrong? 😇

Etherdice – self pwned 😬

Blocks have limited gas allowances

- Limits the transactions in a block
- Limits the gas of a single transaction



Etherdice iterated all bets when closing a round

- Gas usage increased linearly with popularity 😕
  Lesson: Operations above O(1) will eventually exceed the gas limit!*
- Reaching critical mass, the contract locked up 😬

If you don't know the language dynamics...

GovernMental – 0xf45717552f12ef7cb65e95476f217ea008167ae3

Twisted (https://github.com/GovernMental/GovernMental) Ponzi scheme with smart contra

- Newcomers invest money to become members
- Members earn returns from newcomer investments

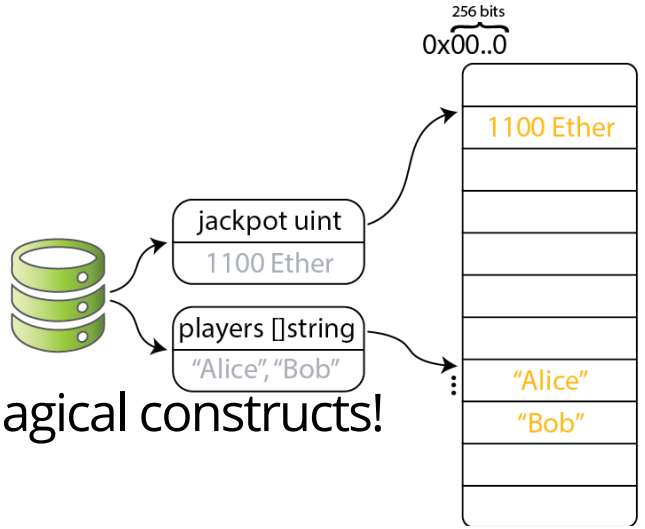
```
contract GovernMental {
    address[] creditors; uint[] credited;

    function invest() {
        if (block.timestamp - lastInvested < TWELVE_HOURS) { ... } else {
            creditors = new address[](0);
            credited  = new uint[](0);
        }
    }
    [...]
}
```

Casually reset the contract at round end... what could go wrong? 😇

GovernMental – self pwned

Contract storage in EVM is a single hash map

- All contract fields map into the same storage area
- Array elements map into the same storage space too



Freeing up a field ⇔ zeroing out a storage entry

Lesson: Understand and avoid magical constructs!

- Freeing up an array ⇔ freeing all associated entries

King of the Ether – 0xb336a86e2feb1e87a328fcb7dd4d04de3df254d0

Game of Thrones pyramid [contract](https://github.com/kieranelby/KingOfTheEtherThrone/tree/v0.4.0/)

- Usurpers pay the ether-price for the throne
- The ruler is paid and mysteriously disappears

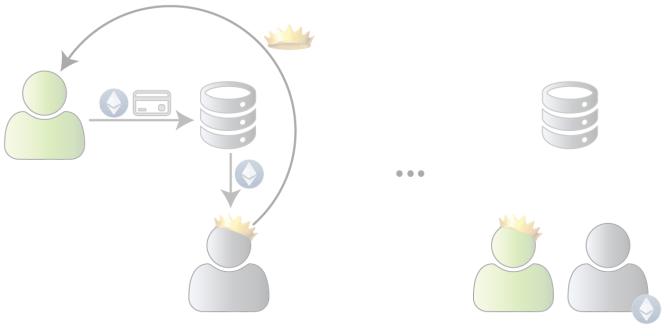
```
contract KingOfTheEtherThrone {
    address monarch;

    function claim() {
        // [...] calculate the ruler's compensation

        monarch.send(compensation);
        monarch = msg.sender;
    }
    [...]
}
```

Send blindly to compensate the previous ruler... what could go wrong? 😇

King of the Ether – broken

Sending funds is an external CALL operation

- Recipient execution limited to 2300 gas
- Returns whether the send succeeded



But what if the ether transfer failed?

- Ignore it? Bail out? Work around?
- Not caring easily breaks invariants!

Lesson: Anything that can go wrong, will go wrong!

If you don't know the EVM dyamics...

MakerDAO – 0xe02640be68df835aa3327ea6473c02c8f6c3815a

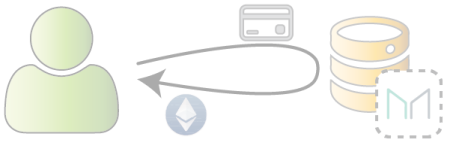
Contracts (https://github.com/makerdao) and frameworks (https://github.com/nexusdev) for an ge

- Users deposit and trade various tokens (and Ether)
- Users are free to withdraw coins at any point

```
contract MakerEthToken {
    function withdraw(uint amount) {
        if (balances[msg.sender] >= amount) {
            if (msg.sender.call.value(amount)()) {
                balances[msg.sender] -= amount;
            }
        }
    }
    [...]
}
```

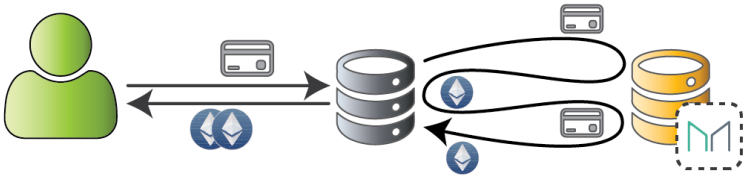
Send funds with full gas allowance... what could go wrong? 😇

MakerDAO – preventive pwned



Calling another contract relinquishes execution

- Arbitrary code may execute (different context)
- Entire granted gas allowance may be consumed



Recipient may have enough gas to call further

- Can update multiple related contracts (good)
- Can call back in to the original contract (hmmm)

Lesson: External calls will eventually loop back in!

Pre-homestead multisig wallet

Wallet contract (https://github.com/ethereum/meteor-dapp-wallet/blob/e0980c6905006456945b8b465df71ea1bf2...le authorizations

- To save on deploy: user → stub (multi) → code (single)
- Context needs to be forwarded down the call chain

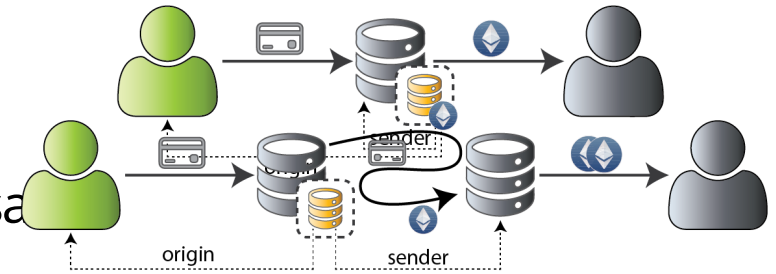
```
contract MultiOwned {
    mapping(uint => uint) owners;

    modifier onlyowner {
        if (owners[tx.origin] > 0) {
            _
        }
    }
    [...]
}
```

Simply use tx.origin for authentication... what could go wrong? 😇

Pre-homestead multisig wallet – swapped before pwn

Pre-homestead, libraries used CALLCODE

- Forwards runtime context, except msg.sender
- Libraries relied on tx.origin to authorize the transa



Internal transactions retain the same tx.origin

- My nested contracts can authorize me (good)
- Not my nested contracts can reenter as me (oops)

Lesson: Authorization forwarding is exceptionally risky!

TheDAO – 0xbb9bc244d798123fde783fcc1c72d3bb8c189413

Beer keg challenge 🍺

Beer keg – 0x629469c8db3a4d7bcc3a823effcf8900119ba

Untappable (http://etherscan.io/address/0x629469c8db3a4d7bcc3a823effcf8900119ba7e7#code) beer contract

- A round of beer inside (5 Ether)
- Crack it open? Have a round on me!

```
contract BeerKeg {
    bytes20 prev; // Nickname of the previous tap attempt

    function tap(bytes20 nickname) {
        prev = nickname;
        if (prev != nickname) {
            msg.sender.send(this.balance);
        }
    }
}
```

Lesson: You tell me! 😈

Legacy of the fallen ones... ฝ(ʘ*´ᴗ`)ʘ

- Accounts are free, instantaneous and infinite! (20 Ether)
- Blockchain state is free for all to use and abuse! (150 Ether)
- Operations above O(1) will exceed the gas limit! (5192 Ether)
- Understand and avoid magical constructs! (1100 Ether)
- Anything that can go wrong, will go wrong! (42 Ether)
- External calls will eventually loop back in! (5800 Ether)
- Authorization forwarding is exceptionally risky! (0 Ether)
- Never forget Mt. Gox! Never forget TheDAO! (3.6M Ether)
- Bonus: Compilers are written by mere mortals! (5 Ether)

Thank you

Péter Szilágyi
peter@ethereum.org (mailto:peter@ethereum.org)
@peter_szilagyi (http://twitter.com/peter_szilagyi)

Ethereum Core Developer
https://ethereum.org/ (https://ethereum.org/)