



智能合约安全入门

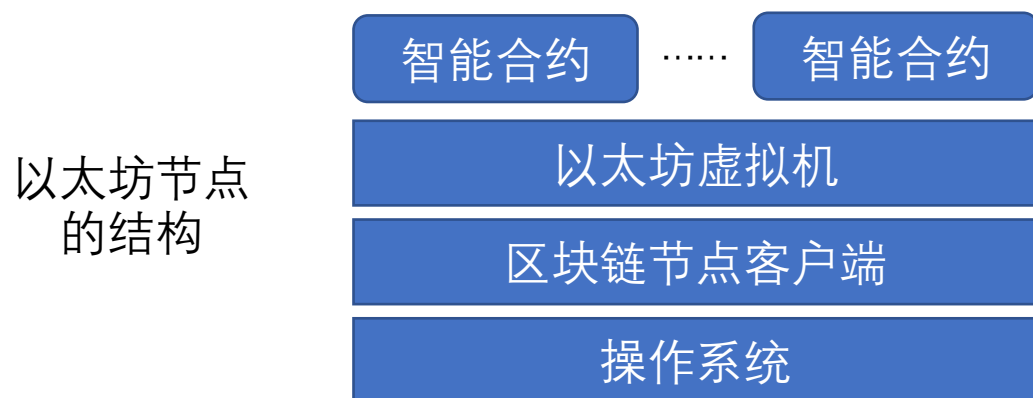


Fernando

智能合约是什么？

- 智能合约是根据事先任意制定的规则来自动转移数字资产的系统。
- 智能合约是运行在区块链的具备图灵完备性的分布式应用程序。

在股权众筹、游戏、保险、供应链、物联网等领域都有应用。



[1]Ethereum W. Ethereum Whitepaper[J]. Ethereum. URL: <https://ethereum.org> [accessed 2020-07-07], 2014.

[2]倪远东, 张超, 殷婷婷. 智能合约安全漏洞研究综述[J]. 信息安全学报, 2020, 5(3): 78-99

理解智能合约

- 狭义的智能合约 = 以太坊上的合约账户，与solidity程序相关
- 生命周期：



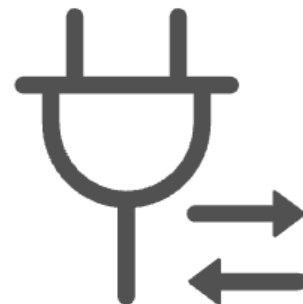
开发



编译



部署



调用



销毁

合约漏洞分类

高级语言层面

整数溢出

变量覆盖

未检验返回值

资产冻结

拒绝服务

任意地址写入

未初始化变量

影子变量

虚拟机层面

重入

代码注入

不一致性攻击

短地址攻击

区块链层面

时间戳依赖

条件竞争

随机性不足

未检验返回值

```
function withdraw(uint256 _amount) public {  
    require(balances[msg.sender] >= _amount);  
    msg.sender.send(_amount);  
    balances[msg.sender] -= _amount;  
    etherLeft -= _amount;  
    emit Withdraw(msg.sender, _amount);  
}
```

返回

整数溢出

0000	1000
0001	1001
0010	1010
0011	1011
0100	1100
0101	1101
0110	1110
0111	1111

变量覆盖

```
pragma solidity ^0.4.21;
contract DVPgame {
    ERC20 public token;
    uint256[] map;
    using SafeERC20 for ERC20;
    using SafeMath for uint256;
    ...
}
```

代码注入

```
contract B{
    function info(bytes data){
        this.call(data) ;
    }
    function secret() public{
        require(this ==msg.sender);
        // secret operations
    }
}
```

返回

短地址攻击

```
0x90b98a11
000000000000000000000000062bec9abe373123b9b635b75608f94eb86441600
0000000000000000000000000000000000000000000000000000000000000002
```

```
0x90b98a11
000000000000000000000000062bec9abe373123b9b635b75608f94eb864416
0000000000000000000000000000000000000000000000000000000000000002
```

```
0x90b98a11
000000000000000000000000062bec9abe373123b9b635b75608f94eb86441600
0000000000000000000000000000000000000000000000000000000000000002
```

[返回](#)

重入

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract EtherStore {
    mapping(address => uint) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw() public {
        uint bal = balances[msg.sender];
        require(bal > 0);

        (bool sent,) = msg.sender.call{value: bal}(""); // Vulnerability of
        require(sent, "Failed to send Ether");

        balances[msg.sender] = 0;
    }

    // Helper function to check the balance of this contract
    function getBalance() public view returns (uint) {
        return address(this).balance;
    }
}
```

返回

条件竞争

看到答案之后抢先提交

```
contract FindThisHash {  
    bytes32 constant public hash = 0xb5b5b97fafd9855eec9b41f74dfb6c38f5951  
  
    constructor() public payable {} // load with ether  
  
    function solve(string solution) public {  
        // If you can find the pre image of the hash, receive 1000 ether  
        require(hash == sha3(solution));  
        msg.sender.transfer(1000 ether);  
    }  
}
```

返回

案例：Akutar



 Contract 0xF42c318dbfBaab0EEE040279C6a2588Fa01a961d   

Buy ▾

Exchange ▾

Earn ▾




Gaming ▾

Featured: Build Precise & Reliable Apps with **Etherscan APIs**. [Learn More!](#)

Contract Overview

Balance: 11,539.5 Ether

Ether Value: \$15,525,012.51 (@ \$1,345.38/ETH)

Token: \$0.00  1  

More Info

 More ▾

My Name Tag: Not Available, [Update?](#)

Contract Creator: [Akutar: Deployer](#) at txn [0x40917b4bad2f8c2f308...](#)

link : <https://etherscan.io/address/0xf42c318dbfbaab0eee040279c6a2588fa01a961d#code>

结构体定义

```
struct bids {  
    address bidder;  
    uint80 price;  
    uint8 bidsPlaced;  
    uint8 finalProcess; //0: Not processed, 1: refunded, 2: withdrawn  
}
```

错误维护状态

```
function processRefunds() external {  
    require(block.timestamp > expiresAt, "Auction still in progress");  
    uint256 _refundProgress = refundProgress;  
    uint256 _bidIndex = bidIndex;  
    require(_refundProgress < _bidIndex, "Refunds already processed");  
  
    uint256 gasUsed;  
    uint256 gasLeft = gasleft();  
    uint256 price = getPrice();  
  
    for (uint256 i=_refundProgress; gasUsed < 5000000 && i < _bidIndex; i++) {  
        bids memory bidData = allBids[i];  
        if (bidData.finalProcess == 0) {  
            uint256 refund = (bidData.price - price) * bidData.bidsPlaced;  
            uint256 passes = mintPassOwner[bidData.bidder];  
            if (passes > 0) {  
                refund += mintPassDiscount * (bidData.bidsPlaced < passes ? bidData.bidsPlaced : passes);  
            }  
            allBids[i].finalProcess = 1;  
            if (refund > 0) {  
                (bool sent, ) = bidData.bidder.call{value: refund}("");  
                require(sent, "Failed to refund bidder");  
            }  
        }  
  
        gasUsed += gasLeft - gasleft();  
        gasLeft = gasleft();  
        _refundProgress++;  
    }  
  
    refundProgress = _refundProgress;  
}
```

用户无法紧急退款

```
function emergencyWithdraw() external {
    require(block.timestamp > expiresAt + 3 days, "Please wait for airdrop period.");

    bids memory bidData = allBids[personalBids[msg.sender]];
    require(bidData.bidsPlaced > 0, "No bids placed");
    require(bidData.finalProcess == 0, "Refund already processed");

    allBids[personalBids[msg.sender]].finalProcess = 2;
    (bool sent, ) = bidData.bidder.call{value: bidData.price * bidData.bidsPlaced}("");
    require(sent, "Failed to refund bidder");
}
```

项目方无法提款

```
function claimProjectFunds() external onlyOwner {
    require(block.timestamp > expiresAt, "Auction still in progress");
    require(refundProgress >= totalBids, "Refunds not yet processed");
    require(akuNFTs.airdropProgress() >= totalBids, "Airdrop not complete");

    (bool sent, ) = project.call{value: address(this).balance}("");
    require(sent, "Failed to withdraw");
}
```

其他案例

- The DAO 攻击（重入）
- BEC（整数溢出）
- KotET（拒绝服务）

应对

- 漏洞挖掘
- 智能合约安全防御
 - 安全编程
 - 热升级
 - 攻击检测和阻断

推荐

- Blocksec : <https://blocksec.com/>
- 漫雾科技 : <https://www.slowmist.com/zh/>
- Seebug : <https://paper.seebug.org/>
- 相关学术会议列表 : <https://zhuanlan.zhihu.com/p/27853093>
- Smart-Contracts-Security-CheckList :
<https://github.com/BookMangeTeam/Ethereum-Smart-Contracts-Security-CheckList>