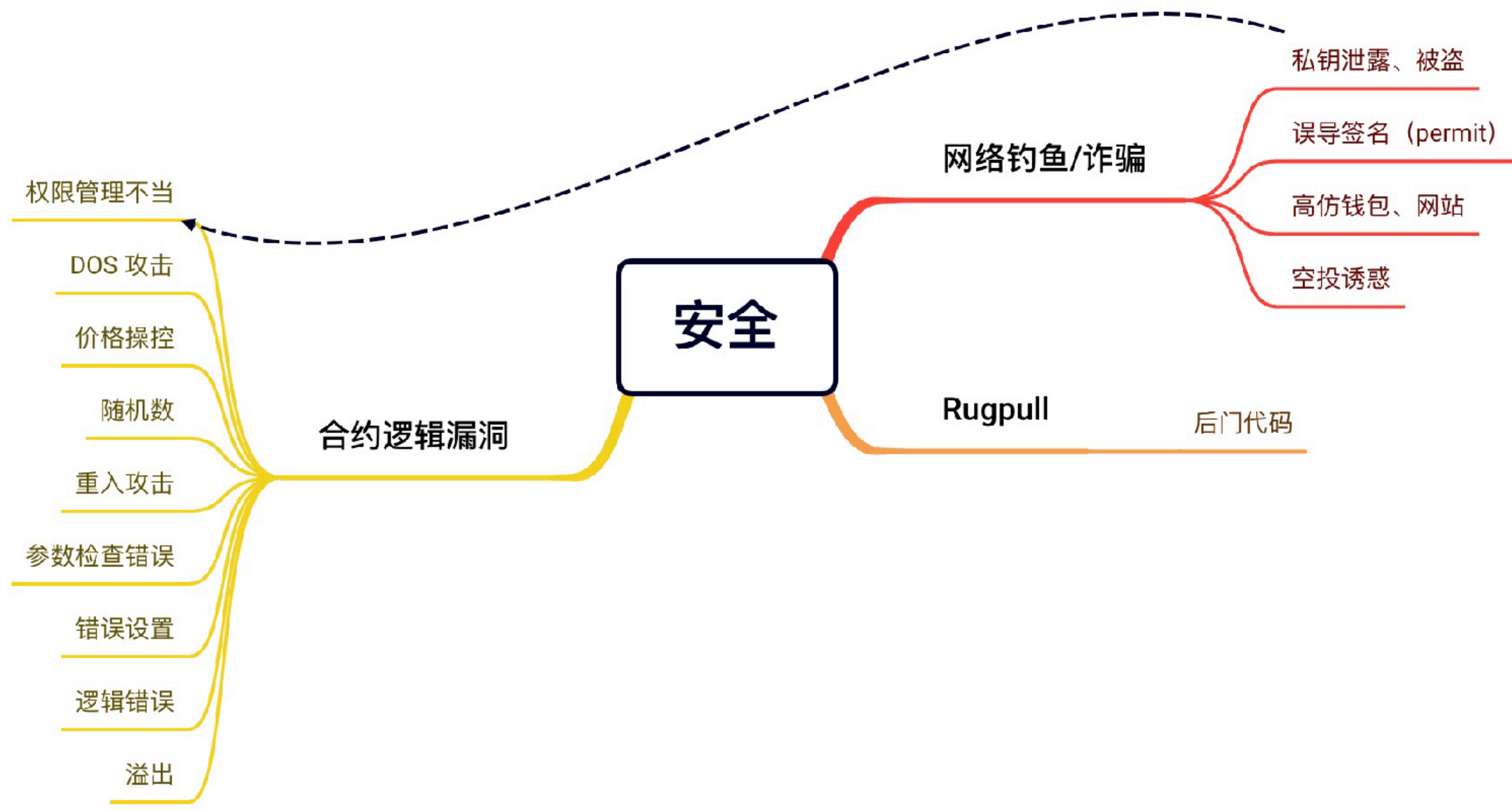


Solidity 安全

安全的重要性

- 合约不可修改 + 直接处理资金
- 每年因安全问题，导致数十亿的美元的资金损失
- 让区块链安全比 Web2 安全 成为一个更突出的问题

常见安全问题





常见漏洞

- 重入攻击
- Dos 拒绝服务
- 签名重用
- 溢出 (Solidity < 0.8) 、精度损失
- 合约账户控制

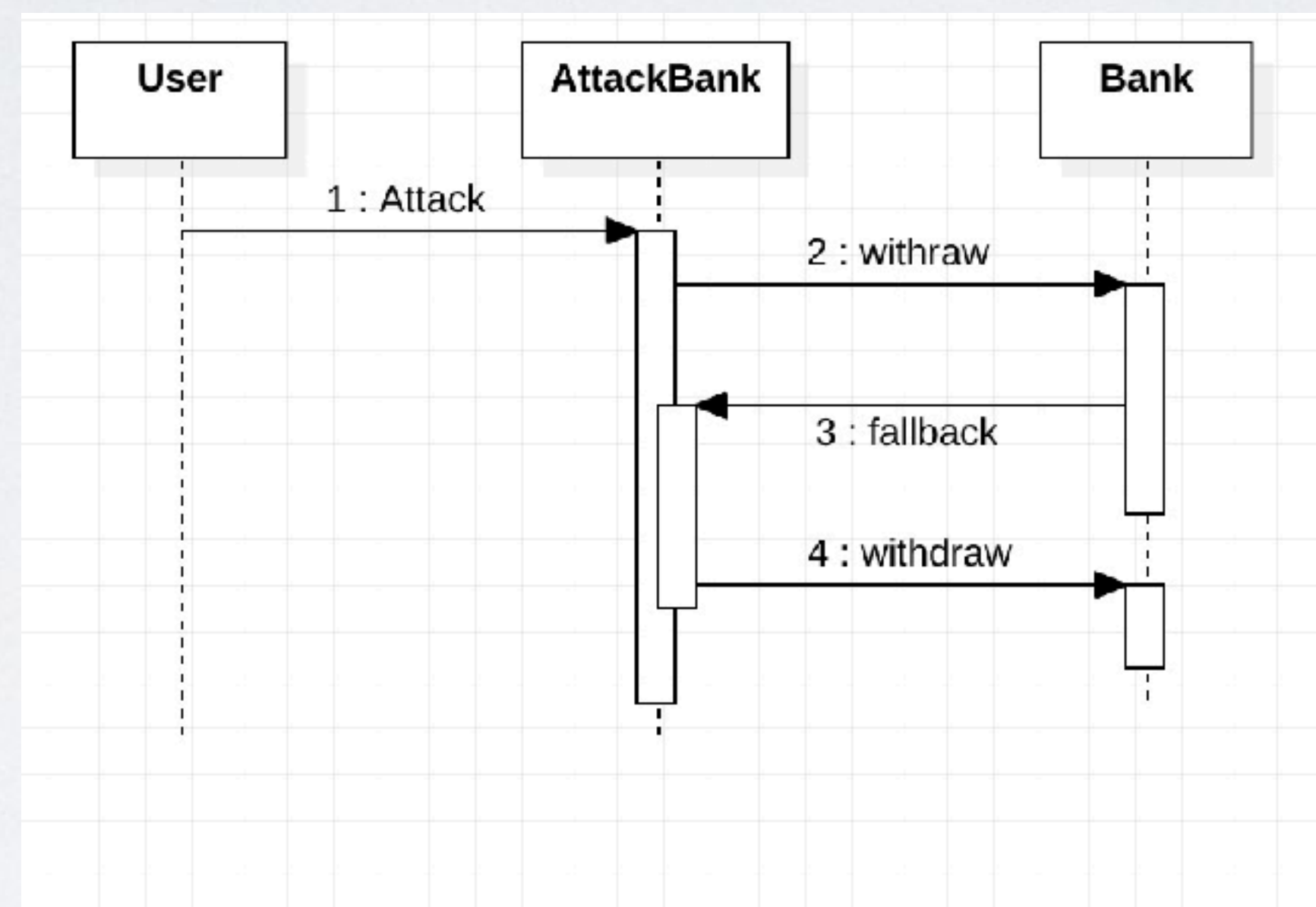
找问题

```
function withdraw() public {  
    (bool success, ) = msg.sender.call{value: deposits[msg.sender]}("");  
    deposits[msg.sender] = 0;  
  
    require(success, "Failed to send Ether");  
}
```


重入攻击

- 调用外部函数时，要时刻注意重入问题：

- 重入



`solidity_sample_code/testReplay.sol`

防范重入攻击问题

- 先检查 - 再修改 - 最后交互 (checks-effect-interaction)
- 重入锁控制

testReplay.sol

找问题

```
function enter() public {  
    // Check for duplicate entrants  
    for (uint256 i; i < entrants.length; i++) {  
        if (entrants[i] == msg.sender) {  
            revert("You've already entered!");  
        }  
    }  
    entrants.push(msg.sender);  
}
```


找问题

```
uint256 public totalDeposits;
mapping(address => uint256) public deposits;

function deposit() external payable {
    deposits[msg.sender] += msg.value;
    totalDeposits += msg.value;
}

function withdraw() external {
    assert(address(this).balance == totalDeposits);

    uint256 amount = deposits[msg.sender];
    totalDeposits -= amount;
    deposits[msg.sender] = 0;

    payable(msg.sender).transfer(amount);    // 0
}
```


找问题

```
bytes32 public constant TYPEHASH = keccak256("withdrawBySig(uint256 amount)");

function withdrawBySig(uint8 v, bytes32 r, bytes32 s, uint256 amount) external payable {
    bytes32 structHash = keccak256(abi.encode(TYPEHASH, amount));
    bytes32 hash = _hashTypedDataV4(structHash);
    address signer = ECDSA.recover(hash, v, r, s);
    require(inWhitelist[signer], "error signer");
    _withdraw(signer, amount);
}

function _withdraw(address user, uint256 amount) internal {
    uint256 currentBalance = balances[user];
    if (currentBalance < amount) {
        revert SignatureReplay__InsufficientBalance(currentBalance, amount);
    }
    balances[user] = currentBalance - amount;
    payable(msg.sender).transfer(amount);
}
```


找问题

```
uint256 public moneyToSplitUp = 225;
uint256 public users = 4;
uint count;
function shareMoney() public view returns (uint256) {
    return moneyToSplitUp / users;
}

function decrement() public {
    unchecked {
        count--;
    }
}
```


找问题

```
function isContract(address account) public view returns (bool) {
    uint size;
    assembly {
        size := extcodesize(account)
    }
    return size > 0;
}

// 确保仅有 EOA 能调用
function protected() external {
    require(!isContract(msg.sender), "no contract allowed");
    .....
}
```


CTF

- <https://ethernaut.openzeppelin.com/>
- <https://capturetheether.com/>
- <https://www.damnvulnerabledefi.xyz/>
- <https://ciphershastra.com/Maya.html>

最佳实践

- <https://consensys.github.io/smart-contract-best-practices/>

作业

- 尝试盗取Vault 中的资金
- <https://decert.me/quests/b5368265-89b3-4058-8a57-a41bde625f5b>