高级话题类6-10

面试问答题（中英文）

高级话题类 Advanced Topics（中英文，6-10）

6：访问控制是什么，为什么重要?

What is access control and why is it important?

答：

访问控制是一种重要的机制，用于限制对智能合约的访问。通过使用访问控制，您可以确保只有经过授权的用户才能执行特定操作或访问敏感信息。这可以帮助保护您的智能合约免受未经授权的访问和攻击。

Solidity提供了几种访问控制修饰符，例如public、private、internal和external。这些修饰符用于控制函数和状态变量的可见性和访问权限。

Access control is an important mechanism for restricting access to smart contracts. By using access control, you can ensure that only authorized users can perform specific actions or access sensitive information. This can help protect your smart contracts from unauthorized access and attacks.

Solidity provides several access control modifiers, such as public, private, internal, and external, which are used to control visibility and access to functions and state variables.

7：什么是抢跑（front running）？

What is front running?

答：

抢跑（front running）是一种攻击行为，指在一笔正常交易等待打包的过程中，抢跑机器人通过设置更高 Gas 费用抢先完成攻击交易。在所有 Front-Running 中，最典型最具危害性的就是针对 AMM 交易的 Sandwich Attacks （三明治攻击）

注意：夹子交易有时也成三明治攻击，但是它是有矿工或验证节点端完成的，能把被攻击的那笔交易夹在中间打包区块。

Front running is a type of attack behavior, which means that while a normal transaction is waiting to be packaged, a front-running bot completes the attack transaction first by setting a higher Gas fee. The most typical and damaging of all Front-Running is Sandwich Attacks on AMM transactions.

Note: Sandwich Attacks are sometimes called Sandwich Attacks, but they are done on the miner's or validator's end, and can pack blocks with the attacked transaction in the middle.

8：什么是提交-揭示方案，何时使用它?

What is Commit-Reveal Scheme and when to use it?

答：

提交-揭示方案（Commit-Reveal Scheme）是一种用于在区块链上进行投票或竞标的协议。该协议的目的是防止参与者在提交投票或竞标之前查看其他参与者的提交，从而保护投票或竞标的公正性。

具体：

1.提交-揭示方案的基本思想是将投票或竞标分为两个阶段：提交阶段和揭示阶段。在提交阶段，参与者将加密的投票或竞标提交到智能合约中。在揭示阶段，参与者揭示他们的加密投票或竞标，并将其与提交阶段中的哈希值进行比较。如果哈希值匹配，则投票或竞标被接受。否则，投票或竞标将被拒绝。

2.可以防止参与者在提交投票或竞标之前查看其他参与者的提交，从而保护投票或竞标的公正性。它还可以防止恶意参与者在提交阶段提交虚假的投票或竞标，因为他们无法预测其他参与者的投票或竞标。

3.通常用于加密货币中的投票或竞标，例如DAO（去中心化自治组织）的投票。它也可以用于其他需要保护公正性的场景，例如拍卖和投标。

The Commit-Reveal Scheme (CRS) is a protocol used to conduct voting or bidding on the blockchain. The purpose of the protocol is to protect the integrity of a vote or bid by preventing participants from viewing other participants' submissions before submitting a vote or bid.

Specifically:

1.the basic idea of the submit-reveal scheme is to divide the voting or bidding into two phases: the submission phase and the reveal phase. In the submission phase, participants submit encrypted votes or bids into the smart contract. In the reveal phase, participants reveal their encrypted vote or bid and compare it to the hash value in the submit phase. If the hashes match, the vote or bid is accepted. Otherwise, the vote or bid is rejected.

2.It can protect the fairness of a vote or bid by preventing participants from viewing other participants' submissions before submitting a vote or bid. It also prevents malicious participants from submitting false votes or bids at the submission stage, as they cannot predict the votes or bids of other participants.

3.It is commonly used for voting or bidding in cryptocurrencies, such as voting in DAOs (Decentralized Autonomous Organizations). It can also be used in other scenarios where fairness needs to be protected, such as auctions and bids.

9：在什么情况下，abi.encodePacked 可能会产生漏洞?

Under what circumstances might abi.encodePacked create a vulnerability?

答：

abi.encodePacked可能会产生漏洞，因为它不会在参数之间添加填充，而是将所有参数拼接在一起。这可能会导致哈希碰撞，从而使攻击者能够伪造交易或执行其他恶意操作。例如，如果攻击者知道您使用abi.encodePacked来编码交易数据，他们可以构造一个具有相同哈希值的交易，从而欺骗您的智能合约。

为了避免这种情况，用abi.encode来编码交易数据，因为它会在每个参数之间添加填充，以确保每个参数占用32个字节。这可以防止哈希碰撞，并提高智能合约的安全性。

Abi.encodePacked may create vulnerabilities because it does not add padding between parameters, but instead stitches all parameters together. This could lead to hash collisions, which could allow an attacker to forge transactions or perform other malicious operations. For example, if an attacker knows that you use abi.encodePacked to encode transaction data, they can construct a transaction with the same hash value to spoof your smart contract.

To avoid this, encode transaction data with abi.encode as it adds padding between each parameter to ensure that each parameter takes up 32 bytes. This prevents hash collisions and improves the security of your smart contract.

10、什么是 gas griefing?

What is gas griefing?

答：

Gas griefing是一种智能合约攻击，攻击者通过发送恰好足够的gas来执行主要智能合约，但未为其子调用或外部通信提供足够的gas，从而导致未受控制的行为并在某些情况下对合约的业务逻辑造成

严重破坏。这种攻击可能会导致合约的不可预测行为，例如无限循环或拒绝服务攻击。为了防止gas griefing攻击，您可以使用以下方法：

1.在智能合约中检查子调用所需的gas量，并确保为其提供足够的gas。

2.使用安全的编程实践来编写智能合约，例如避免使用循环和递归等高消耗操作。

3.使用Solidity的require和assert语句来确保智能合约的正确性和安全性。

Gas griefing is a smart contract attack in which an attacker executes the main smart contract by sending just enough gas to execute it, but does not provide enough gas for its sub-calls or external communications, leading to uncontrolled behavior and in some cases causing severe damage to the contract's business logic. Such attacks can lead to unpredictable behavior of the contract, such as infinite loops or denial-of-service attacks. To prevent a gas griefing attack, you can use the following methods:

1.check the amount of gas required for subcalls in the smart contract and ensure that sufficient gas is provided for them.

2.Use safe programming practices to write smart contracts, such as avoiding high-consumption operations such as loops and recursions.

3.Use Solidity's require and assert statements to ensure the correctness and security of smart contracts.