AUDIT HUNT

SECURITY REVIEW

X

Counting SVG NFT

Prepared for	Counting SVG NFT
Prepared by	Audit Hunt
Dates Audited	September 2 - September 10, 2023

Gas Limit Issues: The Hidden Vulnerabilities in Smart Contracts

Description:

When deploying or interacting with smart contracts, especially those with complex logic or large data structures, users may encounter gas limit issues. These issues can prevent transactions from being mined and can expose hidden vulnerabilities in the smart contract's design.

Steps to Reproduce:

1. Deploy a complex smart contract with multiple functions and large data structures.
2. Try to execute a function that requires a significant amount of computational work or interacts with large data.
3. Observe that the transaction fails due to exceeding the gas limit.

Expected Behavior:

The smart contract should handle large computations and data interactions efficiently, without exceeding the gas limit.

Actual Behavior:

The transaction fails, and an error message indicating that the gas limit has been exceeded is displayed.

Potential Solutions:

- Optimize the smart contract code to reduce computational complexity.
- Break down large functions into smaller, more manageable ones.
- Consider using off-chain computations where possible and only storing essential data on-chain.
- Increase the gas limit for the transaction (though this is not always feasible or recommended).

Additional Information:

- **Smart Contract Address:** 0x1234...abcd
- **Transaction Hash:** 0x5678...efgh
- **Environment:** Ethereum Mainnet
- **Tools Used:** Remix, Metamask

The Dangers of External Calls: Finding Pitfalls in Contract Security

Description:

External calls in smart contracts can introduce a range of security vulnerabilities if not handled correctly. These calls can be hijacked, lead to reentrancy attacks, or cause unexpected behaviors, compromising the integrity of the contract.

Steps to Reproduce:

1. Deploy a smart contract that makes an external call to another contract or address.
2. Interact with the contract in a way that triggers the external call.
3. Observe potential vulnerabilities or unexpected behaviors as a result of the external call.

Expected Behavior:

The smart contract should safely execute external calls without exposing itself to potential attacks or unintended side effects.

Actual Behavior:

The contract's external call can be exploited, leading to potential security breaches such as reentrancy attacks, loss of funds, or other unintended behaviors.

Potential Solutions:

- Use the `checks-effects-interactions` pattern to ensure state changes happen before external calls.
- Avoid using `.call()` with user-supplied data.
- Implement reentrancy guards to prevent recursive calls.
- Always be cautious and aware of the potential side effects of external calls, especially when interacting with untrusted contracts.

Additional Information:

- **Smart Contract Address:** 0xabcd...1234
- **Transaction Hash:** 0xefgh...5678
- **Environment:** Ethereum Mainnet
- **Tools Used:** Truffle, Ganache

Screenshots:

Note: Replace placeholders like `URL_to_screenshot_image`, smart contract address, and transaction hash with actual values when creating the issue on GitHub.

Uncovering Critical Exploits in Ethereum Smart Contracts