7 HIGH ISSUES

Vulnerability Description:

SOLIDITY INCORRECT ACCESS CONTROL

Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract.

Remedy:

It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

Line: 522 – 525

Line 531 – 534

Line 536 – 540

Line 542 – 545

Line 547 – 550

Line 560 – 570

Line 652

****

3 MEDIUM ISSUES

1. STATE MODIFYING FUNCTIONS

Vulnerability Description:

If you are migrating your smart contract from versions <0.5.0 and you used a constant function that is now marked as a pure or view, special care should be taken if these functions modify the state as it may trap a contract in solidity versions >=0.5.0

Remedy:

Do not modify the state of the contract inside functions that are marked as pure, view, or constant.

Line 101 -108

2. ACCOUNT EXISTENCE CHECK FOR LOW LEVEL CALLS

Vulnerability Description:

The low-level calls such as the delegatecall, call, or callcode, do not validate prior to the call if the destination account exists or not. They will always return true even if the account is non-existent, therefore, giving invalid output.

Remedy:

It is recommended to have an account existence check before making these low-level calls to confirm the presence of an external account with some valid code. Eg: using extcodesize.

Line 111 -117

3. INTEGER OVERFLOW/UNDERFLOW

Vulnerability Description:

An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum storage of a variable type. Integers overflow or underflow may prove fatal when during an arithmetic operation, the number goes over or under the designated limit. This may prove fatal during calculations related to ether or tokens.

Remedy:

Solidity compiler versions >=0.8.0 automatically handle overflow and underflow validations. If you're using a lower solidity version, it is recommended to use the SafeMath library to protect the arithmetic operations.

Line 205

****

9 LOW ISSUES

1. CUSTOM ERRORS TO SAVE GAS

Vulnerability Description:

The contract was found to be using revert() statements. Since Solidity v0.8.4, custom errors have been introduced which are a better alternative to the revert. This allows the developers to pass custom errors with dynamic data while reverting the transaction and also making the whole implementation a bit cheaper than using reverts.

Remedy:

It is recommended to replace all the instances of revert() statements with error() to save gas.

Line 155

2. CHEAPER INEQUALITIES IN REQUIRE()

Vulnerability Description:

The contract was found to be doing comparisons using inequalities inside the require statement. When inside the require statements, non-strict inequalities (>=, <=) are usually costlier than the strict equalities (>, <).

Remedy:

It is recommended to go through the code logic, and, if possible, modify the non-strict inequalities with the strict ones to save ~3 gas as long as the logic of the code is not affected.

Line 33, Line 45, Line 112, Line 136

3. CHEAPER INEQUALITIES IN IF()

Vulnerability Description:

The contract was found to be doing comparisons using inequalities inside the if statement. When inside the if statements, non-strict inequalities (>=, <=) are usually cheaper than the strict equalities (>, <).

Remedy:

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save ~3 gas as long as the logic of the code is not affected.

Line 148, Line 590, Line 702, Line 933