WEB3 CITADEL

Froggies Audit Report

Gas optimizations

Gas	Issue	Instances
Gas-1	Use selfbalance() instead of address(this).balance	2
Gas-2	Use assembly to check for address(0)	3
Gas-3	Using bools for storage incurs overhead	2
Gas-4	Cache array length outside of loop	1
Gas-5	Don't initialize variables with default value	2
Gas-6	++i costs less gas than i++ , especially when it's used in for -loops (--i /i-- too)	2
Gas-7	Use != 0 instead of > 0 for unsigned integer comparison	1

FROGGIES

Critical High Medium Low Informational

File name: deployed-mainnet.sol | MD5 checksum: 6eaf39c06217b1b4f5c8b11836ae2623

Line no : 1025, 1046

Issue Type

[GAS-1] Use selfbalance() instead of address(this).balance

Issue Details

Use assembly when getting a contract's balance of BNB.

You can use selfbalance() instead of address(this).balance when getting your contract's balance of BNB to save gas. Additionally, you can use balance(address) instead of address.balance() when getting an external contract's balance of BNB.

Saves 15 gas when checking internal balance, 6 for external

Instances (2):

```
File: deployed-mainnet.sol

1025:         uint256 initialBalance = address(this).balance;

1046:         return address(this).balance.sub(initialBalance);
```

Line no : 695, 696, 709

Issue Type

[GAS-2] Use assembly to check for address(0)

Issue Details

Saves 6 gas per instance

Instances (3):

```
File: deployed-mainnet.sol

695:         if (owner == address(0)) revert InvalidAddress();

696:         if (spender == address(0)) revert InvalidAddress();

709:             if (from == address(0) || to == address(0)) revert InvalidAddress();
```

Critical High Medium Low Informational

Line no : 501, 502

Issue Type

[GAS-3] Using bools for storage incurs overhead

Issue Details

Use uint256(1) and uint256(2) for true/false to avoid a Gwarmaccess (100 gas), and to avoid Gsset (20000 gas) when changing from ‘false’ to ‘true’, after having been ‘true’ in the past. See source.

Instances (2):

```
File: deployed-mainnet.sol

501:          mapping(address => bool) _isExcludedFromFee;

502:          mapping(address => bool) _isExcludedFromReward;
```

Line no : 651, 1448

Issue Type

[GAS-4] Cache array length outside of loop

Issue Details

If not cached, the solidity compiler will always read the length of the array during each iteration. That is, if it is a storage array, this is an extra sload operation (100 additional extra gas for each iteration except for the first) and if it is a memory array, this is an extra mload operation (3 additional gas for each iteration except for the first).

Instances (2):

```
File: deployed-mainnet.sol

651:          for (uint256 i = 0; i < state._excludedFromReward.length; i++) {
```

Line no : 651, 1448

Issue Type

[GAS-5] Don't initialize variables with default value

Instances (2):

```
File: deployed-mainnet.sol

651:          for (uint256 i = 0; i < state._excludedFromReward.length; i++) {

1448:          for (uint256 i = 0; i < excludedLength; i++) {
```

Line no : 651, 1448

Issue Type

[GAS-6] ++i costs less gas than i++ , especially when it's used in for -loops (--i /i-- too)

Issue Details

Saves 5 gas per loop

Instances (2):

```
File: deployed-mainnet.sol

651:          for (uint256 i = 0; i < state._excludedFromReward.length; i++) {

1448:          for (uint256 i = 0; i < excludedLength; i++) {
```

Line no : 1430

Issue Type

[GAS-7] Use != 0 instead of > 0 for unsigned integer comparison

Instances (2):

```
File: deployed-mainnet.sol

1430:          if (state._rOwned[_account] > 0) {
```

Critical Issues

No critical issues found

Non Critical Issues

No non critical issues found

Informational

[INFO-1] Centralization Risk in token supply

Top 10 addresses hold 34.07% of total supply at the time of audit that can potentially drive the price from a centralized control

[INFO-2] Centralization Risk in ownership

0×6790e946cd8E548e09b39DB42A9D4322B36Eef30 is admin of the contract at the time of audit and has control to modify various tax percentages,
Maximum tax that can be applied is 15%.