



# Smart Contract Audit

FOR  
**AMONGAI**

DATED : 08 January 2024



# AUDIT SUMMARY

**Project name - AMONGAI**

**Date:** 08 January, 2024

**Scope of Audit-** Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

**Audit Status:** Passed

## Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	2	3	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0



# USED TOOLS

---

## Tools:

### 1- Manual Review:

A line by line code review has been performed by audit ace team.

**2- BSC Test Network:** All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

### 3- Slither :

The code has undergone static analysis using Slither.

### Testnet version:

The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:

<https://testnet.bscscan.com/address/0xb99855985c0cf8e14c545c3c778c715d8d47ee5c#code>

---



# Token Information

---

**Token Address:**

0x9D1d562Bf89c138DfdD2D6E8C854490A983a20A9

**Name:** AMONGAI

**Symbol:** AMNG

**Decimals:** 18

**Network:** BscScan

**Token Type:** BEP-20

**Owner:**

0x60e06a65bCd2F0A16f236573A7056f6472FD2E27

**Deployer:**

0x60e06a65bCd2F0A16f236573A7056f6472FD2E27

**Token Supply:** 100000000

**Checksum:** f2032c616934aeb47e6039f76b20d2h5

**Testnet:**

<https://testnet.bscscan.com/address/0xb99855985c0cf8e14c545c3c778c715d8d47ee5c#code>

---



# TOKEN OVERVIEW

---

**Buy Fee:** 0-15%

---

**Sell Fee:** 0-15%

---

**Transfer Fee:** 0-0%

---

**Fee Privilege:** Owner

---

**Ownership:** Owned

---

**Minting:** None

---

**Max Tx:** Yes

---

**Blacklist:** No

---



# AUDIT METHODOLOGY

---

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

# VULNERABILITY CHECKLIST



Return values of low-level calls



**Gasless Send**



Private modifier



Using block.timestamp



Multiple Sends



Re-entrancy



Using Suicide



Tautology or contradiction



Gas Limit and Loops



Timestamp Dependence



Address hardcoded



Revert/require functions



Exception Disorder



Use of tx.origin



Using inline assembly



Integer overflow/underflow



Divide before multiply



Dangerous strict equalities



Missing Zero Address Validation



Using SHA3



Compiler version not fixed



Using throw

# CLASSIFICATION OF RISK

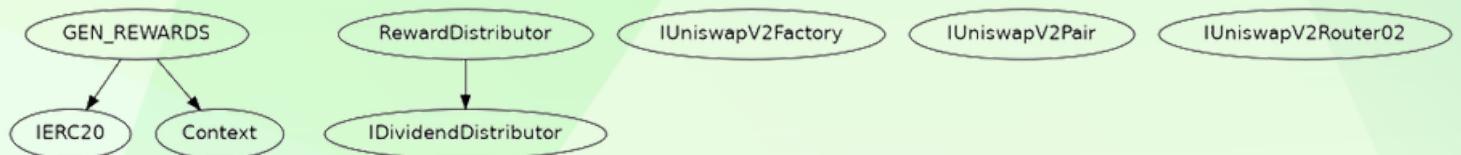
Severity	Description
◆ Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
◆ High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
◆ Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
◆ Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
◆ Gas Optimization / Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

## Findings

Severity	Found
◆ Critical	0
◆ High-Risk	0
◆ Medium-Risk	2
◆ Low-Risk	3
◆ Gas Optimization / Suggestions	1

# INHERITANCE TREE

---





## POINTS TO NOTE

---

- The owner can transfer ownership.
  - The owner can renounce ownership.
  - The owner can Enable trading.
  - The owner can set the fees not more than 15%.
  - The owner can set wallet limits.
  - The owner can rescue trapped tokens.
  - The owner can update telegram group/Lp locks URL/Website URL.
  - The owner can manually swap tokens.
  - The owner can include/exclude wallets from rewards.
-



# STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in RewardDistributor.process(uint256) (GEN_REWARDS.sol#1223-1248):
    External calls:
        - distributeDividend(shareholders[currentIndex]) (GEN_REWARDS.sol#1240)
            - IERC20(RWDTKN).transfer(sharerholder,amount) (GEN_REWARDS.sol#1273)
    State variables written after the call(s):
        - currentIndex = 0 (GEN_REWARDS.sol#1236)
RewardDistributor.currentIndex (GEN_REWARDS.sol#1159) can be used in cross function reentrancies:
    - RewardDistributor.process(uint256) (GEN_REWARDS.sol#1223-1248)
    - currentIndex ++ (GEN_REWARDS.sol#1245)
RewardDistributor.currentIndex (GEN_REWARDS.sol#1159) can be used in cross function reentrancies:
    - RewardDistributor.process(uint256) (GEN_REWARDS.sol#1223-1248)
Reentrancy in RewardDistributor.setShare(address,uint256) (GEN_REWARDS.sol#1186-1200):
    External calls:
        - distributeDividend(sharerholder) (GEN_REWARDS.sol#1188)
            - IERC20(RWDTKN).transfer(sharerholder,amount) (GEN_REWARDS.sol#1273)
    State variables written after the call(s):
        - shares[sharerholder].amount = amount (GEN_REWARDS.sol#1198)
RewardDistributor.shares (GEN_REWARDS.sol#1148) can be used in cross function reentrancies:
    - RewardDistributor.distributeDividend(address) (GEN_REWARDS.sol#1257-1275)
    - RewardDistributor.getUnpaidEarnings(address) (GEN_REWARDS.sol#1277-1288)
    - RewardDistributor.setShare(address,uint256) (GEN_REWARDS.sol#1186-1200)
    - RewardDistributor.shares (GEN_REWARDS.sol#1148)
    - shares[sharerholder].totalExcluded = getCumulativeDividends(shares[sharerholder].amount) (GEN_REWARDS.sol#1199)
RewardDistributor.shares (GEN_REWARDS.sol#1148) can be used in cross function reentrancies:
    - RewardDistributor.distributeDividend(address) (GEN_REWARDS.sol#1257-1275)
    - RewardDistributor.getUnpaidEarnings(address) (GEN_REWARDS.sol#1277-1288)
    - RewardDistributor.setShare(address,uint256) (GEN_REWARDS.sol#1186-1200)
    - RewardDistributor.shares (GEN_REWARDS.sol#1148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
GEN_REWARDS.addLiquidity(uint256,uint256) (GEN_REWARDS.sol#1027-1038) ignores return value by uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Liquidity,block.timestamp) (GEN_REWARDS.sol#1030-1037)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
INFO:Detectors:
GEN_REWARDS.allowance(address,address).owner (GEN_REWARDS.sol#784) shadows:
    - GEN_REWARDS.owner() (GEN_REWARDS.sol#760-762) (function)
GEN_REWARDS._approve(address,address,uint256).owner (GEN_REWARDS.sol#806) shadows:
    - GEN_REWARDS.owner() (GEN_REWARDS.sol#760-762) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GEN_REWARDS.swapTriggerCount(uint256) (GEN_REWARDS.sol#495-498) should emit an event for:
    - swapTrigger = Transaction_Count + 1 (GEN_REWARDS.sol#497)
GEN_REWARDS.setDistributionGas(uint256) (GEN_REWARDS.sol#651-656) should emit an event for:
    - distributorGas = Gas_Amount (GEN_REWARDS.sol#654)
RewardDistributor.setDistributionCriteria(uint256,uint256) (GEN_REWARDS.sol#1180-1184) should emit an event for:
    - minPeriod = _minPeriod (GEN_REWARDS.sol#1182)
    - minDistribution = _minDistribution (GEN_REWARDS.sol#1183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
GEN_REWARDS.constructor(string,string,uint256,uint256,address)._OwnerWallet (GEN_REWARDS.sol#147) lacks a zero-check on :
    - _owner = _OwnerWallet (GEN_REWARDS.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
RewardDistributor.distributeDividend(address) (GEN_REWARDS.sol#1257-1275) has external calls inside a loop: IERC20(RWDTKN).transfer(sharerholder,amount) (GEN_REWARDS.sol#1273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop
INFO:Detectors:
Reentrancy in GEN_REWARDS.Open_Trade() (GEN_REWARDS.sol#381-424):
    External calls:
        - uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(this),uniswapV2Router.WETH()) (GEN_REWARDS.sol#405)
    State variables written after the call(s):
        - _isExcludedFromRewards[uniswapV2Pair] = true (GEN_REWARDS.sol#419)
        - _isExcludedFromRewards[address(this)] = true (GEN_REWARDS.sol#420)
        - _isExcludedFromRewards[_owner] = true (GEN_REWARDS.sol#421)
        - _isExcludedFromRewards[Wallet_Burn] = true (GEN_REWARDS.sol#422)
        - _isLimitExempt[uniswapV2Pair] = true (GEN_REWARDS.sol#417)
        - _isPair[uniswapV2Pair] = true (GEN_REWARDS.sol#416)
Reentrancy in GEN_REWARDS._transfer(address,address,uint256) (GEN_REWARDS.sol#850-941):
    External calls:
        - processFees(contractTokens) (GEN_REWARDS.sol#913)
            - (SendSuccess,None) = address(_to).call{value: _amount}() (GEN_REWARDS.sol#832)
            - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Liquidity,block.timestamp) (GEN_REWARDS.sol#1030-1037)
```



# STATIC ANALYSIS

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#timestamp
INFO:Detectors:
RewardDistributor.process(uint256) (GEN_REWARDS.sol#1223-1248) has costly operations inside a loop:
  - currentIndex = 0 (GEN_REWARDS.sol#1236)
RewardDistributor.distributeDividend(address) (GEN_REWARDS.sol#1257-1275) has costly operations inside a loop:
  - totalDistributed += amount (GEN_REWARDS.sol#1268)
RewardDistributor.process(uint256) (GEN_REWARDS.sol#1223-1248) has costly operations inside a loop:
  - currentIndex ++ (GEN_REWARDS.sol#1245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
GEN_REWARDS._transfer(address,address,uint256) (GEN_REWARDS.sol#850-941) has a high cyclomatic complexity (18).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Pragma version0.8.19 (GEN_REWARDS.sol#15) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in GEN_REWARDS.send_BNB(address,uint256) (GEN_REWARDS.sol#830-834):
  - (SendSuccess,None) = address(_to).call{value: _amount}() (GEN_REWARDS.sol#832)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Variable IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (GEN_REWARDS.sol#55) is too similar
  to IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (GEN_REWARDS.sol#56)
Variable GEN_REWARDS.Project_Information().Owner_Wallet (GEN_REWARDS.sol#236) is too similar to GEN_REWARDS.constructor(string,string,uint256,uint256,address)
  .OwnerWallet (GEN_REWARDS.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
GEN_REWARDS.setDistributionGas(uint256) (GEN_REWARDS.sol#651-656) uses literals with too many digits:
  - require(bool,string)(Gas_Amount <= 1000000000000000,RG) (GEN_REWARDS.sol#653)
GEN_REWARDS.slitherConstructorVariables() (GEN_REWARDS.sol#96-1124) uses literals with too many digits:
  - distributorGas = 500000 (GEN_REWARDS.sol#135)
RewardDistributor.slitherConstructorVariables() (GEN_REWARDS.sol#1129-1308) uses literals with too many digits:
  - minDistribution = 5000000000 (GEN_REWARDS.sol#1157)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
RewardDistributor.RWDTKN (GEN_REWARDS.sol#1140) should be constant
RewardDistributor.dividendsPerShareAccuracyFactor (GEN_REWARDS.sol#1154) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
GEN_REWARDS._decimals (GEN_REWARDS.sol#107) should be immutable
GEN_REWARDS.distributor (GEN_REWARDS.sol#216) should be immutable
RewardDistributor.DivRouter (GEN_REWARDS.sol#1142) should be immutable
RewardDistributor._token (GEN_REWARDS.sol#1131) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:GEN_REWARDS.sol analyzed (8 contracts with 93 detectors), 137 result(s) found
```



# FUNCTIONAL TESTING

---

1- **Approve (passed)**:

<https://testnet.bscscan.com/tx/0xebca8cfc6b3d0dda383d9dad739b8828f4158ca28775bb8e80a633a91ff84632>

2- **Increase Allowance (passed)**:

<https://testnet.bscscan.com/tx/0x8dd00d0feea13383c01e1e5ce3d2ff1725a10c69a5c9a47f6722c2bfddbd07ca>

3- **Decrease Allowance (passed)**:

<https://testnet.bscscan.com/tx/0x340d2b437201b410881795e6b53ee4e599f60ef7ef56e809185e814a00ff98ad>

4- **Rewards Exclude Wallet (passed)**:

<https://testnet.bscscan.com/tx/0x8875412e9efdc1af30675f2bc8d18bcf41e5a01d87f56fb402ba3b741c1b6807>

5- **Add Liquidity Pair (passed)**:

<https://testnet.bscscan.com/tx/0xdbf425e18eeb7793f1baff295f35a6229a06a906c21722e80ed5c4622b6e27a2>

---



# FUNCTIONAL TESTING

---

6- **Set Fees (passed)**:

<https://testnet.bscscan.com/tx/0x0ecc1d247ceb1986a3947763b09e42724386064e65f89d89ae16029bc9e9c4c7>

7- **Set Wallet Limits (passed)**:

<https://testnet.bscscan.com/tx/0xe51bf0c88557b1479e671acd80fbf50613639fca6c019960dfe101a475b37c2b>



# MANUAL TESTING

---

**Centralization** – Liquidity is added to EOA.

**Severity:** Medium

**Function:** addLiquidity

**Status:** Open

**Overview:**

Liquidity is added to EOA. It may be drained by the Wallet Liquidity.

```
function addLiquidity(uint256 tokenAmount, uint256  
BNBAmount) private {
```

```
    _approve(address(this), address(uniswapV2Router),  
tokenAmount);
```

```
    uniswapV2Router.addLiquidityETH{value: BNBAmount}(  
address(this),  
    tokenAmount,
```

```
0,  
0,
```

```
    Wallet_Liquidity,  
    block.timestamp
```

```
);  
}
```

**Suggestion:**

It is suggested that the address should be a contract address or a dead address.

---



# MANUAL TESTING

**Centralization – Missing Require Check.**

**Severity: Medium**

**Function: Update\_Wallet\_Marketing**

**Status: Open**

## **Overview:**

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set the address to the contract address, then the Eth will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function Update_Wallet_Marketing(  
    address payable Marketing_Wallet  
)  
    external onlyOwner {  
  
    // Update Marketing Wallet  
    require(Marketing_Wallet != address(0), "W"); // Enter a valid  
    BSC Address  
    Wallet_Marketing = payable(Marketing_Wallet);  
  
}
```

## **Suggestion:**

It is recommended that the address should not be able to set as a contract address.



# MANUAL TESTING

---

## Centralization – Missing Events

**Severity:** Low

**Subject:** Missing Events

**Status:** Open

### Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function swapTriggerCount(uint256 Transaction_Count) external  
onlyOwner {  
    swapTrigger = Transaction_Count + 1;  
}  
function burnFromTotalSupply(bool true_or_false) external  
onlyOwner {  
    burnFromSupply = true_or_false;  
}  
function noFeeWalletTransfers(bool true_or_false) external  
onlyOwner {  
    no_Fee_Transfers = true_or_false;  
}  
function Update_Wallet_Liquidity(  
address Liquidity_Collection_Wallet
```



# MANUAL TESTING

---

```
) external onlyOwner {  
  
// Update LP Collection Wallet  
require(Liquidity_Collection_Wallet != address(0), "W"); // Enter  
a valid BSC Address  
    Wallet_Liquidity = Liquidity_Collection_Wallet;  
  
}  
function Update_Wallet_Marketing(  
  
address payable Marketing_Wallet  
  
) external onlyOwner {  
  
// Update Marketing Wallet  
require(Marketing_Wallet != address(0), "W"); // Enter a valid  
BSC Address  
    Wallet_Marketing = payable(Marketing_Wallet);  
  
}
```



# MANUAL TESTING

---

## Centralization - Missing Zero Address

**Severity:** Low

**Subject:** Shadowing Local

**Status:** Open

### Overview:

functions can take a zero address as a parameter (0x0000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function addLiquidityPair(  
    address Wallet_Address,  
    bool true_or_false)  
  
external onlyOwner {  
  
    _isPair[Wallet_Address] = true_or_false;  
    _isLimitExempt[Wallet_Address] = true_or_false;  
}
```



# MANUAL TESTING

---

## Centralization - Local Variable Shadowing

**Severity:** Low

**Subject:** Shadowing Local

**Status:** Open

```
function allowance(address owner, address spender) public view
override returns (uint256) {
    return _allowances[owner][spender];
}
function _approve(address owner, address spender, uint256
amount) private {
    require(owner != address(0), "APO"); // BEP20: approve from the
zero address
    require(spender != address(0), "APS"); // BEP20: approve from
the zero address

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}
```

### Suggestion:

Rename the local variable that shadows another component.



# MANUAL TESTING

---

## Optimization

**Severity:** Optimization

**Subject:** Remove unused code.

**Status:** Open

### Overview:

Unused variables are allowed in Solidity, and they do not pose a direct security issue. It is the best practice, though to avoid them.

```
event LiquidityAdded(uint256 Tokens_Amount, uint256  
BNB_Amount);
```



# DISCLAIMER

---

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.



# ABOUT AUDITACE

---

We specialize in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.



**<https://auditace.tech/>**



**[https://t.me/Audit\\_Ace](https://t.me/Audit_Ace)**



**[https://twitter.com/auditace\\_](https://twitter.com/auditace_)**



**<https://github.com/Audit-Ace>**

---