AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

BnbRise

DATED : 28 September 2025

AUDIT SUMMARY

Project name – BnbRise

Date: 28 September 2025

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

Audit Status: **Passed With High Risk**

Issues Found

Status	Critical	High	Medium	Low	Informational
Open	0	1	1	4	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.

3- Slither :
The code has undergone static analysis using Slither.

Token Information

Token Address:
0xb7962589f2F58eD37296AC107B0dae9f30D32d4e

Name: BnbRise

Symbol: BNBRISE

Decimals: 9

Network: Bsc Scan

Token Type: BEP-20

Owner: 0x4d81a451B53a48D6a4e4D3F00287641ADF6A5659

Deployer:
0x4d81a451B53a48D6a4e4D3F00287641ADF6A5659

Token Supply: 72,000,000,000

Checksum: 809f98d5565a2ff955f24ddff42fd7b251bf30fc

TOKEN OVERVIEW

Buy Fee: 3-10%

Sell Fee: 3-10%

Transfer Fee: 3-10%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

AUDIT METHODOLOGY

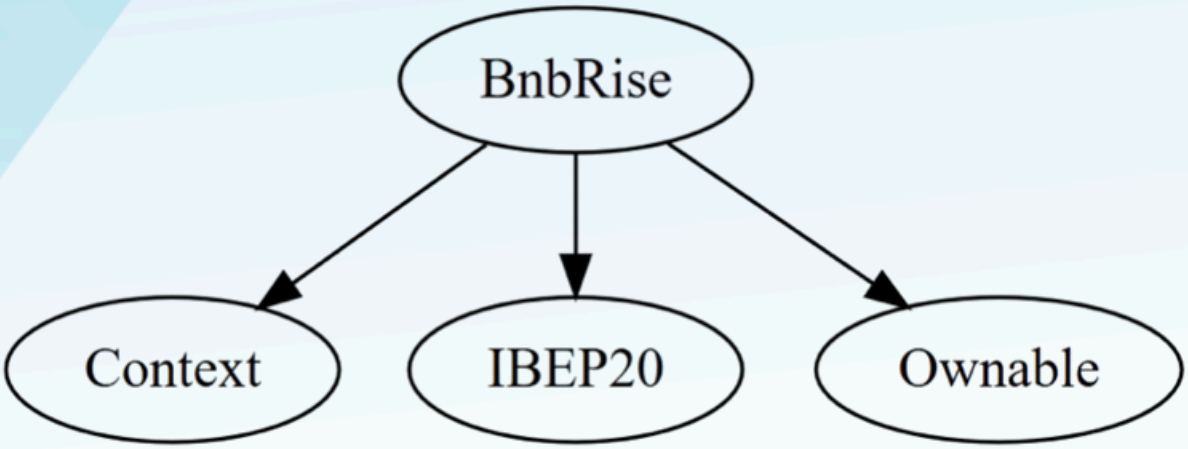
The auditing process will follow a routine as special considerations by Auditace:

- 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 isexercised 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
- Private modifier
- Multiple Sends
- Using Suicide
- Gas Limitand Loops
- Address hardcoded
- Exception Disorder
- Using inline assembly
- Divide before multiply
- Missing Zero Address Validation
- Compiler version not fixed
- **Gasless Send**
- Using block.timestamp
- Re-entrancy
- Tautology or contradiction
- Timestamp Dependence
- Revert/require functions
- Use of tx.origin
- Integer overflow/underflow
- Dangerous strict equalities
- Using SHA3
- Using throw

INHERITANCE TREE



POINTS TO NOTE

- The owner can exclude/includeFrom/InReward.
- The owner can update deadline.
- The owner can Enable trading.
- The owner can exclude/includeFrom/InFee.
- The owner can update Sell/Buy/Launch Taxes.
- The owner can set Sell Taxes not more than 10%.
- The owner can rescue BNB.
- The owner can bulkExcludeFee.
- The owner can update marketing wallet.
- The owner can update dev wallet.
- The owner can update marketing wallet.
- The owner can update Swap tokens at amount.
- The owner can update marketing wallet.
- The owner can rescue any BEP20 Tokens.

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	1
Medium-Risk	1
Low-Risk	4
Optimization/ Informational	1

MANUAL TESTING

Centralization –Enabling trades
Severity: High
Function: EnableTrading
Status: Open

Overview:

```
function EnableTrading() external onlyOwner {
require(!tradingEnabled, "Cannot re-enable trading");
tradingEnabled = true;
swapEnabled = true;
genesis_block = block.number;
}
```

Explanation:

The **EnableTrading** function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

Fix:

To reduce centralization and potential manipulation, consider one of the following approaches:

1. Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2. If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization –Potential Honeypot Risk.

Severity: Medium

Function: UpdateMarketingWallet

Status: Open

Overview:

```
function updateMarketingWallet(address newWallet)
externalonlyOwner {
require(newWallet != address(0), "Fee Address cannot be zero
address");
marketingWallet = newWallet;
}
```

Explanation:

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.

Fix:

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

MANUAL TESTING

Centralization – Unchecked Return Value

Severity: Low

Status: Open

Overview:

rescueAnyBEP20Tokens

IBEP20(_tokenAddr).transfer(msg.sender, _amount)

Explanation:

If the token does not return true, the transfer may silently fail.

Fix:

Check the return value and revert if false.

MANUAL TESTING

Centralization – No Zero address validation

Severity: Low

Status: Open

Overview:

```
function excludeFromFee(address account) public onlyOwner {
_isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
 _isExcludedFromFee[account] = false;
}
```

Explanation:
Allows zero address to be excluded/included, which is unnecessary and could be confusing.

Fix:
Add require(account != address(0), ...)

MANUAL TESTING

Centralization – Unused functions/events

Severity: Low

Status: Open

Overview:

• _msgData() in Context is unused.

• UpdatedRouter event is declared but never emitted.

```
function _msgData() internal view virtual returns (bytes calldata) {
this; // silence state mutability warning without generating bytecode -
see https://github.com/ethereum/solidity/issues/2691
return msg.data;
}
}
event UpdatedRouter(address oldRouter, address newRouter);
```

Explanation:

Unused code increases contract size and can be confusing.

MANUAL TESTING

Centralization – Local Variable Shadowing

Severity: Low

Status: Open

Location:

_approve

```
function _approve(address owner, address spender, uint256 amount)
private {
require(owner != address(0), "BEP20: approve from the zero address");
require(spender != address(0), "BEP20: approve to the zero address");
_allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
}
```

Explanation:

The parameter owner shadows the state variable in Ownable.
Not a bug, but can be confusing.

MANUAL TESTING

Centralization - Floating Pragma

Severity: Informational

Status: Open

Overview:

pragma solidity ^0.8.19;

Explanation:

Floating pragma can lead to unexpected compiler behavior if dependencies use different versions.

Fix:

Use a fixed pragma if possible.

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 specializes 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://twitter.com/auditace_

https://github.com/Audit-Ace