AuditBlock

ContractsAudit.com

DRTP Token

v0.6.12+commit.27d51765

Low-Risk	Medium-Risk	High-Risk
low-risk code	medium-risk code	high-risk code

Contract Address

DRTP Token Deployed On Bscscan.com

0x72e8021aB884786b5EA176c84d0A9CfEdcd2e116

Disclaimer

AuditBlock is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

AuditBlock is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

AuditBlock can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

- BNB Chain Network

Source Code

- AudiTBlock was comissioned by ContrcatsAudit to perform an audit based on the following smart contract:

- https://bscscan.com/address/0x72e8021aB884786b5EA176c84d0A9CfEdcd2e116#code

Global

```
 mapping (address => uint256) private
_rOwned;    mapping (address => uint256)
private _tOwned;    mapping (address =>
mapping (address => uint256)) private
_allowances;        mapping (address =>
bool)    private    _isExcludedFromFee;
mapping  (address  =>  bool)  private
_isExcluded;          address[]  private
_excluded;              uint256  private
constant MAX = ~uint256(0);    uint256
private _tTotal = 100000000000 * 10**9;
uint256 private _rTotal = (MAX - (MAX %
_tTotal));              uint256  private
_tFeeTotal;     string private _name =
"DRTP Token";    string private _symbol
= "DRTP";    uint8 private _decimals =
9;          uint256 public _taxFee = 4;
uint256   private   _previousTaxFee   =
_taxFee;
```

Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise,
after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed
condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5
Contract/ DRTPToken.sol	eeb48fee8f62fb6a50f53b2f228060f4

Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source Sha1 Hash
Contracts	553fbd01798d95ce18de846b295071e7cb77f45d

0.2 Info Found

TOP HOLDERS INFO

- address 0xd0a3...c has **49.95% tokens supply** !
- address 0x8a52...4 has **39.31% tokens supply** !
- address 0xa6b4...0 has **2.89% tokens supply**
- address 0xeee4...e has **2.07% tokens supply**
- address 0xe683...e has **1.41% tokens supply**

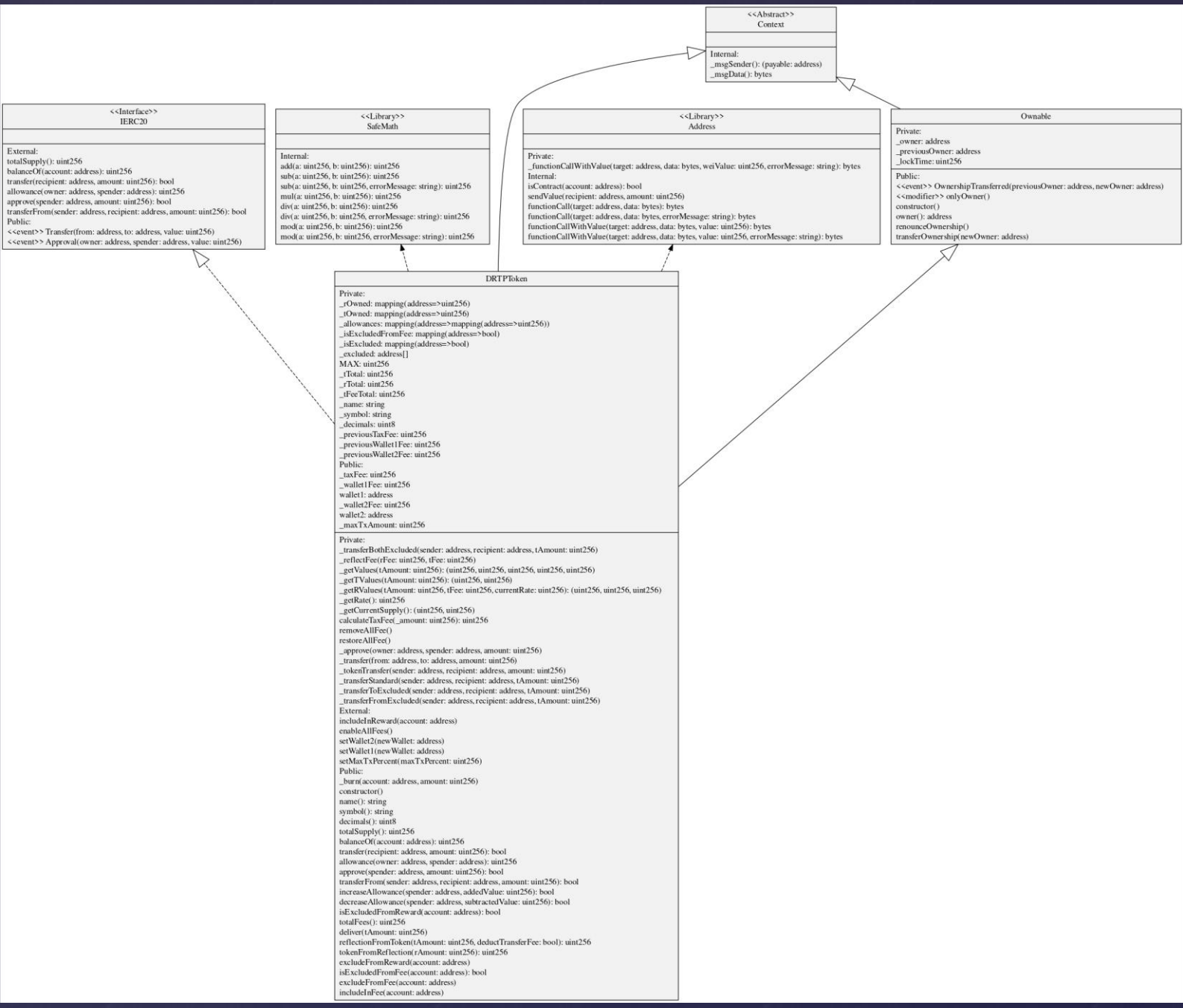
LP INFO

Cannot find LP infos

Warning Advise

The low-level functions , Owner and Found return 0.2 true as their first return value if the account called is non-existent, as part of the design of the EVM. Account existence must be checked prior to calling if needed.

UML


<<Abstract>>
Context
Internal:
_msgSender(): (payable: address)
_msgData(): bytes
<<Interface>>
IERC20
External:
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uint256): bool
Public:
<<event>> Transfer(from: address, to: address, value: uint256)
<<event>> Approval(owner: address, spender: address, value: uint256)
<<Library>>
SafeMath
Internal:
add(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256, errorMessage: string): uint256
mul(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256, errorMessage: string): uint256
mod(a: uint256, b: uint256): uint256
mod(a: uint256, b: uint256, errorMessage: string): uint256
<<Library>>
Address
Private:
_functionCallWithValue(target: address, data: bytes, weiValue: uint256, errorMessage: string): bytes
Internal:
isContract(account: address): bool
sendValue(recipient: address, amount: uint256)
functionCall(target: address, data: bytes): bytes
functionCall(target: address, data: bytes, errorMessage: string): bytes
functionCallWithValue(target: address, data: bytes, value: uint256): bytes
functionCallWithValue(target: address, data: bytes, value: uint256, errorMessage: string): bytes
Ownable
Private:
_owner: address
_previousOwner: address
_lockTime: uint256
Public:
<<event>> OwnershipTransferred(previousOwner: address, newOwner: address)
<<modifier>> onlyOwner()
constructor()
owner(): address
renounceOwnership()
transferOwnership(newOwner: address)
DRTPToken
Private:
_rOwned: mapping(address=>uint256)
_tOwned: mapping(address=>uint256)
_allowances: mapping(address=>mapping(address=>uint256))
_isExcludedFromFee: mapping(address=>bool)
_isExcluded: mapping(address=>bool)
_excluded: address[]
MAX: uint256
_tTotal: uint256
_rTotal: uint256
_tFeeTotal: uint256
_name: string
_symbol: string
_decimals: uint8
_previousTaxFee: uint256
_previousWallet1Fee: uint256
_previousWallet2Fee: uint256
Public:
_taxFee: uint256
_wallet1Fee: uint256
wallet1: address
_wallet2Fee: uint256
wallet2: address
_maxTxAmount: uint256
Private:
_transferBothExcluded(sender: address, recipient: address, tAmount: uint256)
_reflectFee(rFee: uint256, tFee: uint256)
_getValues(tAmount: uint256): (uint256, uint256, uint256, uint256, uint256)
_getTValues(tAmount: uint256): (uint256, uint256)
_getRValues(tAmount: uint256, tFee: uint256, currentRate: uint256): (uint256, uint256, uint256)
_getRate(): uint256
_getCurrentSupply(): (uint256, uint256)
calculateTaxFee(_amount: uint256): uint256
removeAllFee()
restoreAllFee()
_approve(owner: address, spender: address, amount: uint256)
_transfer(from: address, to: address, amount: uint256)
_tokenTransfer(sender: address, recipient: address, amount: uint256)
_transferStandard(sender: address, recipient: address, tAmount: uint256)
_transferToExcluded(sender: address, recipient: address, tAmount: uint256)
_transferFromExcluded(sender: address, recipient: address, tAmount: uint256)
External:
includeInReward(account: address)
enableAllFees()
setWallet2(newWallet: address)
setWallet1(newWallet: address)
setMaxTxPercent(maxTxPercent: uint256)
Public:
_burn(account: address, amount: uint256)
constructor()
name(): string
symbol(): string
decimals(): uint8
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uint256): bool
increaseAllowance(spender: address, addedValue: uint256): bool
decreaseAllowance(spender: address, subtractedValue: uint256): bool
isExcludedFromReward(account: address): bool
totalFees(): uint256
deliver(tAmount: uint256)
reflectionFromToken(tAmount: uint256, deductTransferFee: bool): uint256
tokenFromReflection(rAmount: uint256): uint256
excludeFromReward(account: address)
isExcludedFromFee(account: address): bool
excludeFromFee(account: address)
includeInFee(account: address)

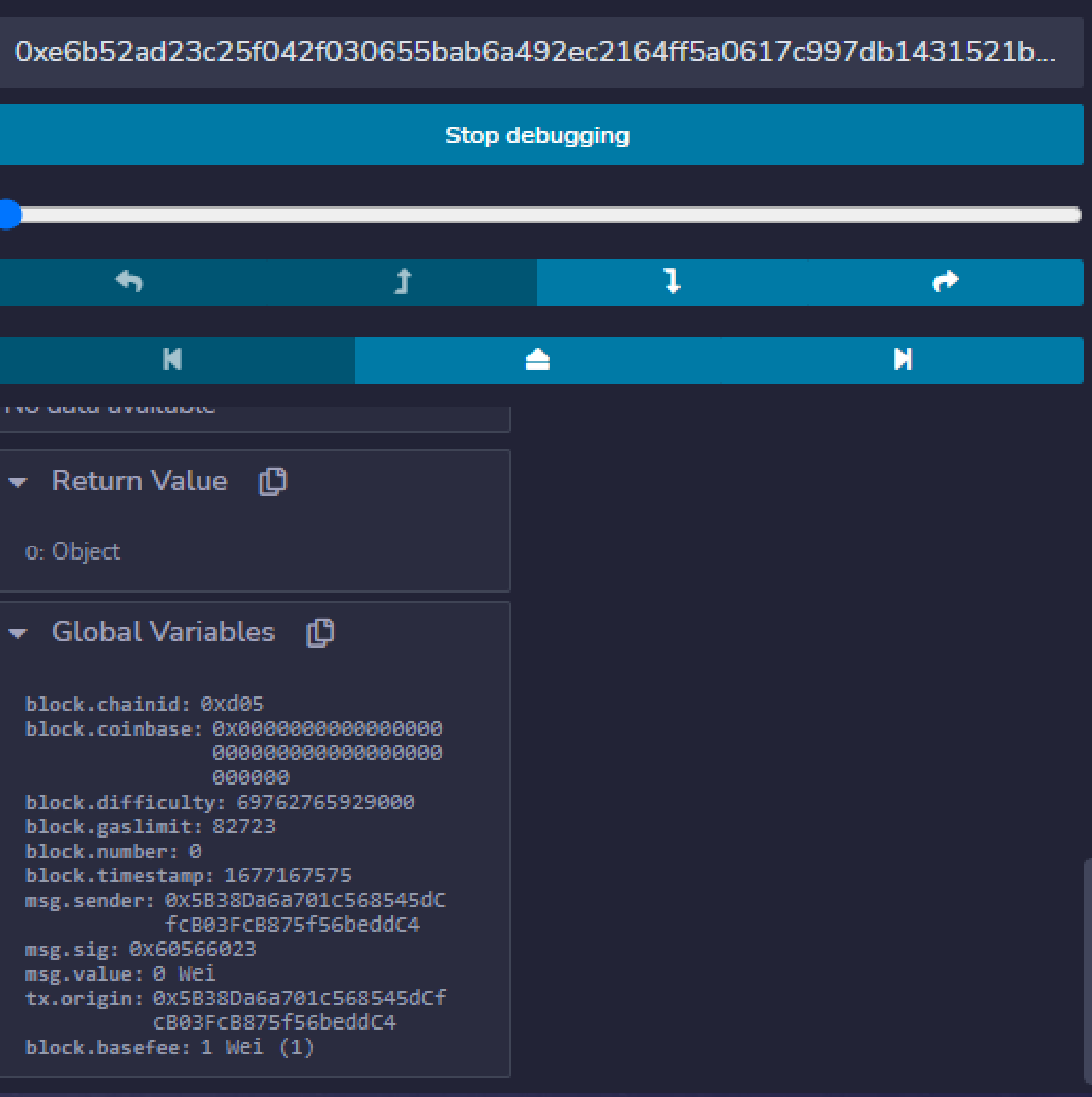
Contract Gas Snapshot

Gas costs:
Gas requirement of function that many is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

For loop over dynamic array: Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to

```
    function includeInReward(address account) external onlyOwner() {
        require(_isExcluded[account], "Account is already excluded");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                _excluded[i] = _excluded[_excluded.length - 1];
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded.pop();
                break;
            }
        }
    }


    function _getCurrentSupply() private view returns(uint256, uint256) {
        uint256 rSupply = _rTotal;
        uint256 tSupply = _tTotal;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] >
tSupply) return (_rTotal, _tTotal);
            rSupply = rSupply.sub(_rOwned[_excluded[i]]);
            tSupply = tSupply.sub(_tOwned[_excluded[i]]);
        }
        if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
        return (rSupply, tSupply);
    }
```



https://docs.soliditylang.org/en/v0.8.17/control-structures.html#error-handling-assert-require-revert-and-exceptions

Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, AudiTBlock experts found **0 big Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, AudiTBlock experts found **1 High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, AudiTBlock experts found **2 Medium issues** in the code of the smart contract.

LOW ISSUES

During the audit, AudiTBlock experts found **1 Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

During the audit, AuditBlock experts found **no Informational issues** in the code of the smart contract.

SWC Attacks

ID	Title		Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓

ID	Title		Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

Owner privileges

- Verify Claims
  The contract block difficulty 69762765929000
  Status: tested and verified

- Owner/Deployer
  Status: tested and verified

- Owner/Deployer _wallet1
  Status: tested and verified

- Owner/Deployer _wallet2
  Status: tested and verified

- Owner/Deployer _enableAllFee
  Status: tested and verified

- Owner/Deployer _excludeFromFee
  Status: tested and verified

Executive Summary

Two (2) independent AuditBlock experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs
The overall code quality is good and not overloaded with unnecessary functions, these is greatly

benefiting the security of the contract. It correctly implemented widely used and reviewed contracts from OpenZeppelin. he main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work.

During the audit, no issues were found after the manual and automated security testing.

Deployed On BNB Mainnet Binance Chain

VERIFIED

https://bscscan.com/token/0x72e8021aB884786b5EA176c84d0A9CfEdcd2e116