AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Flash

DATED : 20 June 23'

FUNCTIONAL TESTING

Centralization – Excessive fees

Severity: High

Status: Open

Overview:

There are no limits for maximum amount of fees, meaning that Ower is able to change buy/sell/transfer fees in range 0-100%.

```
function setTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    taxes = Taxes(_marketing, _liquidity, _burn);
}

function setSellTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    sellTaxes = Taxes(_marketing, _liquidity, _burn);
}
```

Suggestion

Limit maximum amount of fees to 10% buy/sell/transfer

```
function setTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    require(_marketing + _liquidity + _burn <= 10, "Can' set buy/transfer fees more than 10%");
    taxes = Taxes(_marketing, _liquidity, _burn);
}

function setSellTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    require(_marketing + _liquidity + _burn <= 10, "Can' set buy/transfer fees more than 10%");
    sellTaxes = Taxes(_marketing, _liquidity, _burn);
}
```

FUNCTIONAL TESTING

Logical – Upgradeable router

Severity: High

Status: Open

Overview:

Owner is able to update router which is used for performing internal swap.
If router is set to a non contract address or a contract that doesn't support pancakeswap v2 interface (or a malicious contract in general), all interal swap transactions would be failed potentially leading to disabled sell/transfers.

```
function updateRouterAndPair(IRouter _router, address _pair) external onlyOwner {
    router = _router;
    pair = _pair;
}
```

Suggestion

Ensure that router is immutable. This can be achieved by removing updateRouterAndPair function.

FUNCTIONAL TESTING

Logical – zero swap threshold may disable sells

Severity: High

Status: Open

Overview:

setting swapThreshold to zero may revert the sell/transfer transaction if contract has 0 tokens in it, this is because contract performs internal swap regardless of whether contract has tokens or not.

```
function setSwapThreshold(uint256 new_amount) external onlyOwner {
    swapThreshold = new_amount;
}
```

Suggestion

Ensure that swapThreshold is within a reasonable range (ussualy between 0.001% - 1% of total supply)

```
function setSwapThreshold(uint256 new_amount) external onlyOwner {
    require(new_amount >= totalSupply() / 100000, "Can' swapThreshold less than 0.001% of supply");
    require(new_amount <= totalSupply() / 100, "Can' swapThreshold more than 1% of supply");
    swapThreshold = new_amount;
}
```

AUDIT SUMMARY

Project name – Flash

Date: 20 June, 2023

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	3	2	0	0
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/token/0x57bcce62058c383999cd314475cdbf44019b3853

Token Information

Token Name : The Flash

Token Symbol: Flash

Decimals: 9

Token Supply: 420,690,000,000,000

Token Address:
0xd9d7B8B32975c5135F7d31B0dDDD76beee9DF507

Checksum:
642b967271eeff7b8c766465e6d86f888883021a

Owner:
0xA654D2eD88aFD0E1391cC11016260Cd1eE6A3D9c
(at time of writing the audit)

Deployer:
0x60DBce870c3C45405820d751677d56B415885ab7

TOKEN OVERVIEW

Fees:

Buy Fees: 0-100%

Sell Fees: 0-100%

Transfer Fees: 0-100%

Fees Privilege: Owner

Ownership:

0xA654D2eD88aFD0E1391cC11016260Cd1eE6A3D9c

Minting: none

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges:- Initial distribution of the tokens

- Updating fees

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

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

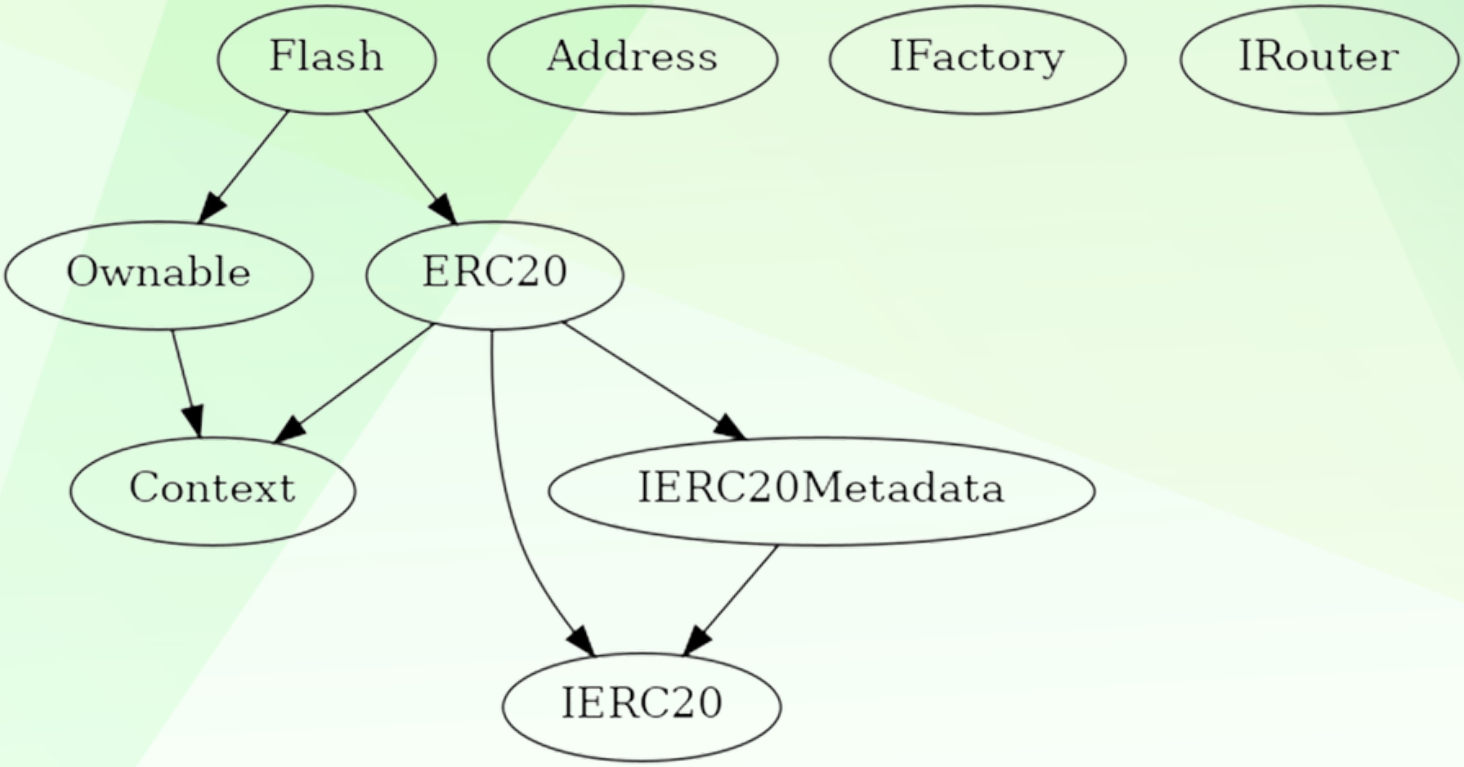
CLASSIFICATION OF RISK

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	3
Medium-Risk	2
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- owner is able to set buy/sell/transfer tax up to 100%
- owner is not able to set max buy/sell/transfer/wallet limits
- owner is not able to blacklist an arbitrary wallet
- owner is not able to mint new tokens
- owner is not able to disable trades

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
Address	Library			
└	sendValue	Internal 🔒	🛑	

CONTRACT ASSESMENT

Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IFactory	Interface			
└	createPair	External ❗	🛑	NO ❗
IRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💸	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
Flash	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	decimals	Public ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	swapForFees	Private 🔐	🛑	inSwap
└	swapTokensForBnb	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	setSwapEnabled	External ❗	🛑	onlyOwner
└	setSwapThreshold	External ❗	🛑	onlyOwner
└	setTaxes	External ❗	🛑	onlyOwner
└	setSellTaxes	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateRouterAndPair	External ❗	🛑	onlyOwner
└	updateExcludedFromFees	External ❗	🛑	onlyOwner
└	rescueBEP20	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💸	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💸	Function is payable

STATIC ANALYSIS

```
Reentrancy in Flash.swapForFees() (contracts/Token.sol#474-500):
        External calls:
        - swapTokensForBnb(toSwap) (contracts/Token.sol#484)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#510)
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (contracts/Token.sol#492)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/Token.sol#518-525)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (contracts/Token.sol#492)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/Token.sol#518-525)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#308)
                - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (contracts/Token.sol#492)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/Token.sol#13-16) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#277-288) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#329-334):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#332)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable ERC20._balances (contracts/Token.sol#55) is not in mixedCase
Variable ERC20._allowances (contracts/Token.sol#57) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#377) is not in mixedCase
Parameter Flash.setSwapThreshold(uint256).new_amount (contracts/Token.sol#532) is not in mixedCase
Parameter Flash.setTaxes(uint256,uint256,uint256)._marketing (contracts/Token.sol#536) is not in mixedCase
Parameter Flash.setTaxes(uint256,uint256,uint256)._liquidity (contracts/Token.sol#536) is not in mixedCase
Parameter Flash.setTaxes(uint256,uint256,uint256)._burn (contracts/Token.sol#536) is not in mixedCase
Parameter Flash.setSellTaxes(uint256,uint256,uint256)._marketing (contracts/Token.sol#540) is not in mixedCase
Parameter Flash.setSellTaxes(uint256,uint256,uint256)._liquidity (contracts/Token.sol#540) is not in mixedCase
Parameter Flash.setSellTaxes(uint256,uint256,uint256)._burn (contracts/Token.sol#540) is not in mixedCase
Parameter Flash.updateRouterAndPair(IRouter,address)._router (contracts/Token.sol#548) is not in mixedCase
Parameter Flash.updateRouterAndPair(IRouter,address)._pair (contracts/Token.sol#548) is not in mixedCase
Parameter Flash.updateExcludedFromFees(address,bool)._address (contracts/Token.sol#553) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#14)" inContext (contracts/Token.sol#8-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x6a1785ad4ea4dbcff0a4ca1935fffcb9601b8d171c37b5ad0937e3c159a2fe2b

2- Buying when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x8e27bdebc58263c2e7a357938a6936c145da8f5cefc9e1601873d967a7bacffa

3- Selling when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x669317d84dc0442d71769be7e5bf3aa60ef18a68790798ce49a489648a46cb48

4- Transferring when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x8af66f27c43aaedfc9e88df4f185548ded1f00bf6758b46fbb41cf263121664e

5- Buying when not excluded from fees (0-100% tax) (passed):

https://testnet.bscscan.com/tx/0x5d0d28b582b26cee3ea349cd278f26e66a3f39b832fa1b85593866c30d94f74c

6- Selling when not excluded from fees (0-100% tax) (passed):

https://testnet.bscscan.com/tx/0x8b27f53021ece3473fc3eaa324aa8b2933dab1a3d631411970072e3346dde0a0

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-100% tax) (passed):

https://testnet.bscscan.com/tx/0x0c2f32ea5c936d61d35c0a50ff2a315d737708d8009b9aa7409b1e28e63ff2df

8- Internal swap (passed):

- BNB fee sent to marketing wallet
- Auto liquidity are sent to owner's wallet
- A portion of tokens burnt (sent to dead wallet)

https://testnet.bscscan.com/tx/0x8b27f53021ece3473fc3eaa324aa8b2933dab1a3d631411970072e3346dde0a0

FUNCTIONAL TESTING

Centralization – Excessive fees

Severity: High

Status: Open

Overview:

There are no limits for maximum amount of fees, meaning that Ower is able to change buy/sell/transfer fees in range 0-100%.

```
function setTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    taxes = Taxes(_marketing, _liquidity, _burn);
}

function setSellTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    sellTaxes = Taxes(_marketing, _liquidity, _burn);
}
```

Suggestion

Limit maximum amount of fees to 10% buy/sell/transfer

```
function setTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    require(_marketing + _liquidity + _burn <= 10, "Can' set buy/transfer fees more than 10%");
    taxes = Taxes(_marketing, _liquidity, _burn);
}

function setSellTaxes(uint256 _marketing, uint256 _liquidity, uint256 _burn) external onlyOwner {
    require(_marketing + _liquidity + _burn <= 10, "Can' set buy/transfer fees more than 10%");
    sellTaxes = Taxes(_marketing, _liquidity, _burn);
}
```

FUNCTIONAL TESTING

Logical – Upgradeable router

Severity: High

Status: Open

Overview:

Owner is able to update router which is used for performing internal swap.
If router is set to a non contract address or a contract that doesn't support pancakeswap v2 interface (or a malicious contract in general), all interal swap transactions would be failed potentially leading to disabled sell/transfers.

```
function updateRouterAndPair(IRouter _router, address _pair) external onlyOwner {
    router = _router;
    pair = _pair;
}
```

Suggestion

Ensure that router is immutable. This can be achieved by removing updateRouterAndPair function.

FUNCTIONAL TESTING

Logical – zero swap threshold may disable sells

Severity: High

Status: Open

Overview:

setting swapThreshold to zero may revert the sell/transfer transaction if contract has 0 tokens in it, this is because contract performs internal swap regardless of whether contract has tokens or not.

```
function setSwapThreshold(uint256 new_amount) external onlyOwner {
    swapThreshold = new_amount;
}
```

Suggestion

Ensure that swapThreshold is within a reasonable range (ussualy between 0.001% - 1% of total supply)

```
function setSwapThreshold(uint256 new_amount) external onlyOwner {
    require(new_amount >= totalSupply() / 100000, "Can' swapThreshold less than 0.001% of supply");
    require(new_amount <= totalSupply() / 100, "Can' swapThreshold more than 1% of supply");
    swapThreshold = new_amount;
}
```

FUNCTIONAL TESTING

Centralization – Owner receiving LP tokens from auto liquidity

Severity: Medium

Status: Open

Overview:

Owner wallet receives LP tokens which are generated through auto – liquidity. This tokens can be used to remove a portion of liquidity pool

```
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(router), tokenAmount);

    // add the liquidity
    router.addLiquidityETH{value: bnbAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

Suggestion

Its suggested to either burn or lock those new LP tokens .

Locking:

```
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(router), tokenAmount);

    uitn256 beforeLP = IER20(pair).balanceOf(address(this));
    router.addLiquidityETH{value: bnbAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        address(this),
        block.timestamp
    );
    uitn256 afterLP = IER20(pair).balanceOf(address(this));
            SomeLockContract.lock(pair, afterLP – beforeLP); //just an example
}
```

Burning:

```
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(router), tokenAmount);
    router.addLiquidityETH{value: bnbAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        0xdead,
        block.timestamp
    );
}
```

FUNCTIONAL TESTING

Logical – Marketing wallet rejecting BNB

Severity: Medium

Status: Open

Overview:

If marketing wallet is set to a contract that doesn't accept BNB, sending BNB to this wallet during internal swap will be reverted potentially disabling sell/transfers.

Example of a contract that rejects BNB:

```
function sendValue(address payable recipient, uint256 amount) internal {
    require(address(this).balance >= amount, "Address: insufficient balance");

    (bool success,) = recipient.call{value: amount}("");
    require(success, "Address: unable to send value, recipient may have reverted");
}

contract Rejector {
  receive() external payable {
    revert("Can't receive BNB");
  }
}
```

Suggestion

Ignore checking return whether low-level call to marketing wallet was successful or not. In below code, even if marketing wallet is set to ta contract that doensn't accept BNB, the transaction wont be reverted.

```
function sendValue(address payable recipient, uint256 amount) internal {
    require(address(this).balance >= amount, "Address: insufficient balance");

    (bool success,) = recipient.call{value: amount}(""); //ignoring success
}
```

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