AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Mushee

DATED : 8 APR 23'

AUDIT SUMMARY

Project name – Mushee

Date: 8 April, 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	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1.Manual Review: The code has undergone a line-by-line review by the 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.

Token Information

Name : Mushee

Symbol : MSH

Decimals: 18

Netowrk: BSC

Token Type: BEP20

Owner: Not ownable

Deployer:
0xB158DD1bAF461D930a9aa0C2eD003E913A5AAEFC

Token Address :
0x6e937Ec2a209060278F591B026b386dB0c7b88D0

Token Information

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilige: None

Ownership : None

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: ---

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

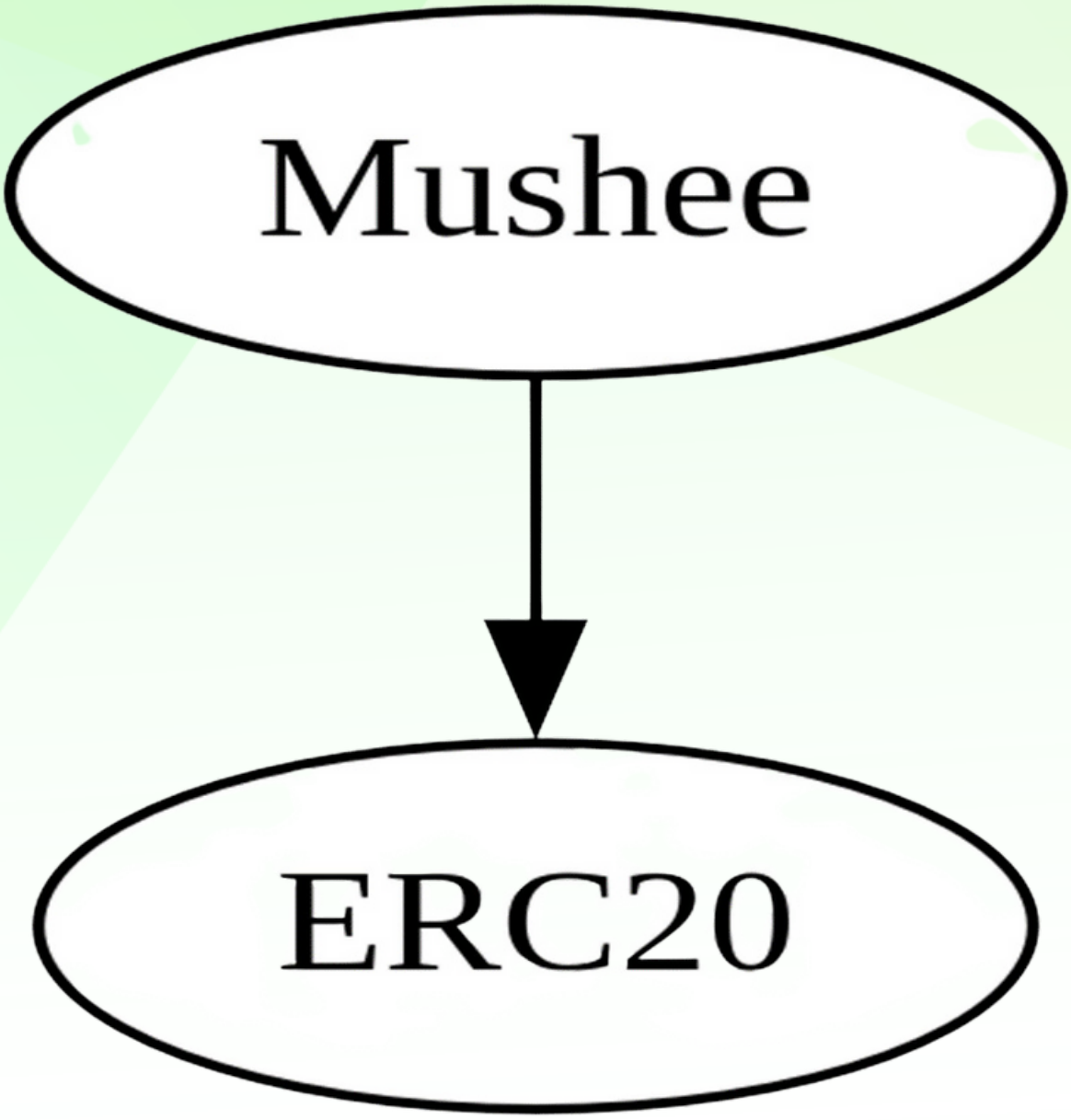
CLASSIFICATION OF RISK

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	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set buy/sell/transfer taxes
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens

TOKEN DISTRIBUTION

it should be noted that the deployer wallet currently holds 100% of the total supply. However, information about the distribution of these tokens is not available, and it is recommended that investors exercise caution when considering this aspect.

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
ERC20	Implementation				
Mushee	Implementation	ERC20			
└	<Constructor>	Public ❗	🛑	NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	approve	Public ❗	🛑	NO ❗

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Pragma version^0.8.2 (contracts/Token.sol#6) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Mushee.slitherConstructorVariables() (contracts/Token.sol#9-54) uses literals with too many digits:
        - totalSupply = 1100000 * 10 ** 18 (contracts/Token.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Mushee.decimals (contracts/Token.sol#15) should be constant
Mushee.name (contracts/Token.sol#13) should be constant
Mushee.symbol (contracts/Token.sol#14) should be constant
Mushee.totalSupply (contracts/Token.sol#12) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

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

No issues found

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x7bd48d027220411560fc5da9e64a8591eda23b8bcf75a7f50982bd0e88dd9533

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x02704bdc8f86b12ae3faebfd07cc1aa16cefa219048e57973f6b7de228c0d251

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xcfb456f7c24237fc935e624c01e64a93192af14a6f3ff5f066cf6fe1f017a4a0

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf638e3e3e8b2aadb7ade8bc29addfbcb0c228cb90ae32718946b8f2ef4776937

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