AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

MONEY

DATED : 20 June 23'

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: **High**

function: EnableTrading

Status: Resolved (owned by safu dev)

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

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

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.

2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.

3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

AUDIT SUMMARY

Project name – MONEY

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	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	1	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/0xb3CecC7030E98DD1e46b95F90Fc95b5f172a4a3b

Token Information

Token Name : MONEY

Token Symbol: $MONEY

Decimals: 18

Token Supply: 69,000,000

Token Address:
0x6203Ac5627BbC7F66C8c1957C6F83e417bFd6c17

Checksum:
17f7deaa6ca72c461a3c4445ab1653259608ff4a

Owner:
0xF6200c127b7D9B1FDb2F45C2F5E8e12AA4A2Cb50
(at time of writing the audit)

Deployer:
0xF6200c127b7D9B1FDb2F45C2F5E8e12AA4A2Cb50

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: No fees

Ownership: Not owned

Minting: none

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges:- Initial distribution of the tokens

- enabling trades

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
- ✅ **Gasless Send**
- ✅ Private modifier
- ✅ Using block.timestamp
- ✅ Multiple Sends
- ✅ Re-entrancy
- ✅ Using Suicide
- ✅ Tautology or contradiction
- ✅ Gas Limitand 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

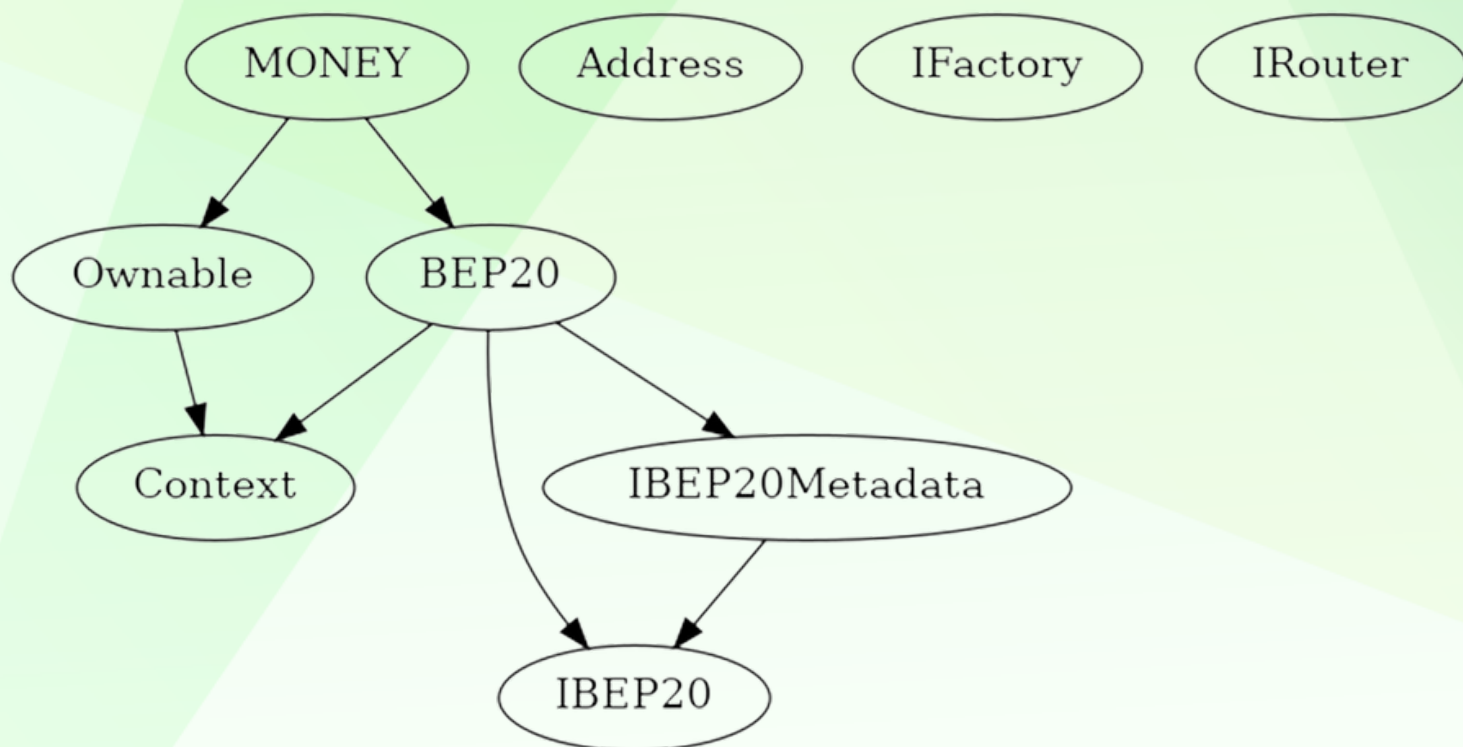| Severity	Found
◆ Critical	0
◆ High-Risk	1
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	0

INHERITANCE TREE

POINTS TO NOTE

- owner is not able to set buy/sell/transfer fees (0% tax)
- 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
- **owner must enable trades manually for investors**

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IBEP20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	⬤	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	⬤	NO ❗
└	transferFrom	External ❗	⬤	NO ❗
IBEP20Metadata	Interface	IBEP20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
BEP20	Implementation	Context, IBEP20, IBEP20Metadata		
└	<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 🔒	⬤	
└	_tokengeneration	Internal 🔒	⬤	
└	_approve	Internal 🔒	⬤	
Address	Library			
└	sendValue	Internal 🔒	⬤	
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	⬤	NO ❗

CONTRACT ASSESMENT

└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	⬛	onlyOwner
└	transferOwnership	Public ❗	⬛	onlyOwner
└	_setOwner	Private 🔐	⬛	
IFactory	Interface			
└	createPair	External ❗	⬛	NO ❗
IRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
MONEY	Implementation	BEP20, Ownable		
└	<Constructor>	Public ❗	⬛	BEP20
└	approve	Public ❗	⬛	NO ❗
└	transferFrom	Public ❗	⬛	NO ❗
└	increaseAllowance	Public ❗	⬛	NO ❗
└	decreaseAllowance	Public ❗	⬛	NO ❗
└	transfer	Public ❗	⬛	NO ❗
└	_transfer	Internal 🔒	⬛	
└	EnableTrading	External ❗	⬛	onlyOwner
└	updateWhitelist	External ❗	⬛	onlyOwner
└	bulkWhitelist	External ❗	⬛	onlyOwner
└	rescueETH	External ❗	⬛	onlyOwner
└	rescueERC20	External ❗	⬛	onlyOwner
└	burnERC20	External ❗	⬛	onlyOwner
└	<Receive Ether>	External ❗	🆂🅿	NO ❗

Legend

Symbol	Meaning
⬛	Function can modify state
🆂🅿	Function is payable

STATIC ANALYSIS

```
Address.sendValue(address,uint256) (contracts/Token.sol#290-295) is never used and should be removed
Context._msgData() (contracts/Token.sol#24-27) 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#17) 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#290-295):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#293)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BEP20._balances (contracts/Token.sol#66) is not in mixedCase
Variable BEP20._allowances (contracts/Token.sol#68) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#338) is not in mixedCase
Function MONEY.EnableTrading() (contracts/Token.sol#412-415) is not in mixedCase
Parameter MONEY.updateWhitelist(address,bool)._address (contracts/Token.sol#417) is not in mixedCase
Constant MONEY.deadWallet (contracts/Token.sol#349) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#25)" inContext (contracts/Token.sol#19-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

MONEY.pair (contracts/Token.sol#345) should be immutable
MONEY.router (contracts/Token.sol#344) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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/0x1b3cb0da28521fde903fc97f47e4c64
7ac1f3e4445bdc24dd9595fd13031fcbf

2- Buying (0% tax) (passed):

https://testnet.bscscan.com/tx/0xe097f94c8c0d1605b87c5d0d4b5df4
f020befc49717c7de8678c5cf9bd2e5930

3- Selling (0% tax) (passed):

https://testnet.bscscan.com/tx/0xd8800a5e770e252fbef60ef12a23204
17013ad888044963a5143d241b7c425fc

4- Transferring (0% tax) (passed):

https://testnet.bscscan.com/tx/0x4fee1f80be68a5a41c562281c0b6e2f
a4762f6dba21f1f37eaa90a3b3eb5026a

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: **High**

function: EnableTrading

Status: **Resolved (owned by safu dev)**

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

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

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.

2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.

3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

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