AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

VPF

DATED : 7 June 23'

AUDIT SUMMARY

Project name – VPF

Date: 7 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	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/0x59CbA4F3b0A40809fB7b94B9c233f5834a6843AF

Token Information

Token Name : Vision Pro Finance

Token Symbol: VPF

Decimals: 18

Token Supply: 100,000,000

Token Address: ---

Checksum:
a8e59ab43e0fc1e6fca77e8fd58d2caa5828157c

Owner:
---**(at time of writing the audit)**

Deployer:

TOKEN OVERVIEW

Fees:

Buy Fees: 0-3%

Sell Fees: 0-3%

Transfer Fees: 0-3%

Fees Privilege: Owner

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: - changing swap threshold

- changing fees

- initial distribution of the tokens

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

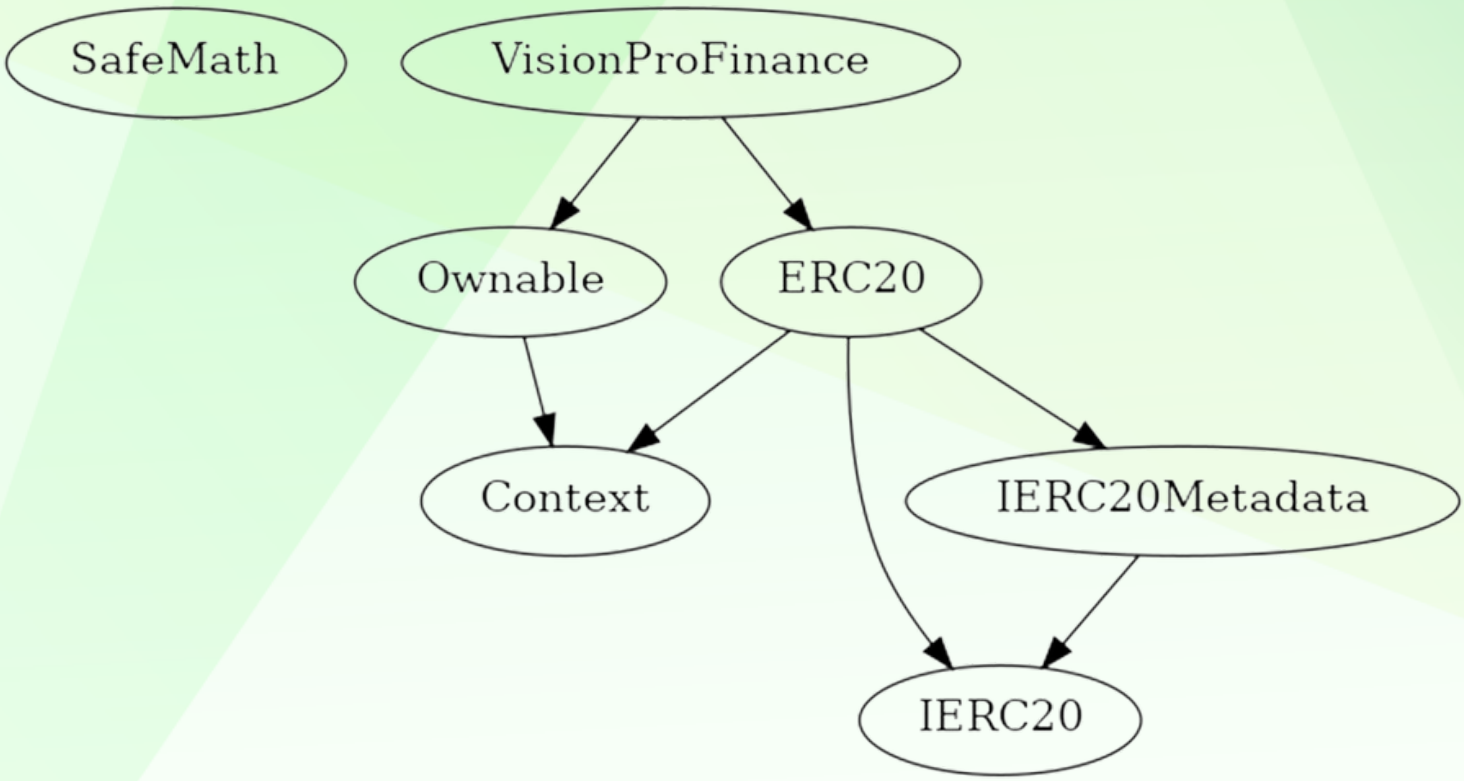
CLASSIFICATION OF RISK

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 fees more than 3% each
- owner is not able to blacklist an arbitrary wallet
- owner is not able to set limit for buy/sell/transfer/holding amounts
- owner is not able to mint new tokens
- owner is not able to disable trades
- owner can exclude/include an address from fees
- owner can update buy/sell/transfer fees
- owner can claim stuck tokens
- owner can transfer ownership
- owner can renounce ownership

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	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 ❗

CONTRACT ASSESMENT

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 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
VisionProFinance	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	setTaxWallet	External ❗	🛑	onlyOwner
└	setWalletsExcludedFromFee	External ❗	🛑	onlyOwner
└	unsetWalletsExcludedFromFee	External ❗	🛑	onlyOwner
└	setTax	External ❗	🛑	onlyOwner
└	recoverTokensFromContract	External ❗	🛑	onlyOwner
└	recoverEthFromContract	External ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reentrancy in VisionProFinance.recoverTokensFromContract(address) (contracts/Token.sol#843-847):
        External calls:
        - IERC20(_tokenAddress).transfer(msg.sender,balance) (contracts/Token.sol#845)
        Event emitted after the call(s):
        - TokensRecovered(balance) (contracts/Token.sol#846)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/Token.sol#238-240) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#700-716) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/Token.sol#95-97) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#189-194) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#153-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#211-216) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Token.sol#109-111) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/Token.sol#170-175) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Token.sol#24-30) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#66-71) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#78-83) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#49-59) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#37-42) 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

Parameter VisionProFinance.setTax(uint256)._tax (contracts/Token.sol#837) is not in mixedCase
Parameter VisionProFinance.recoverTokensFromContract(address)._tokenAddress (contracts/Token.sol#843) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in VisionProFinance.recoverEthFromContract() (contracts/Token.sol#849-853):
        External calls:
        - address(owner()).transfer(balance) (contracts/Token.sol#851)
        Event emitted after the call(s):
        - ETHRecovered(balance) (contracts/Token.sol#852)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

VisionProFinance.constructor() (contracts/Token.sol#810-814) uses literals with too many digits:
        - _mint(msg.sender,100000000 * 10 ** decimals()) (contracts/Token.sol#811)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

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

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x146704a1715ec724e6a049861f9efcfe8b099c1dd4aaf68ce2fa078c0dcfd494

2- Buying (0% tax) when excluded from fees (passed):
https://testnet.bscscan.com/tx/0xa831949ade40a8d8628b670788fe8350678f518afc1c3bbc303008e25414bb71

3- Selling (0% tax) when excluded from fees (passed):
https://testnet.bscscan.com/tx/0x1ed76b7d1bfa765a8aa1f8b34fc9ba747ee70ab3039b96ba487eb0243be40372

4- Transferring (0% tax) when excluded from fees (passed):
https://testnet.bscscan.com/tx/0x30196c0d1ccb62cedc438217912ac597a793edd74fe039e1c9f4fbfdd86e0d88

5- Buying when not excluded from fees (0-3% tax) (passed):
https://testnet.bscscan.com/tx/0xa72ac917ab34c606f8387e0708c813dbf7f164bb1ef03043be5e042452f3047f

6- Selling when not excluded from fees (0-3% tax) (passed):
https://testnet.bscscan.com/tx/0x7a234487c8ecfeee03f29e95aed47538ebaa7174bd2396ca207045d7d0e9dcf0

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-3% tax) (passed):
https://testnet.bscscan.com/tx/0xf4c0a8251a7a02cef0b03328126faef475fff57cd95fa35ba0c19cb898d1b6ed

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