auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Phoenix Chain

DATED : 26 Mar 23'

AUDIT SUMMARY

Project name – Phoenix Chain

Date: 26 March, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed (Contract is developed by Pinksale safu dev)**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	1	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	1	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0x00401808da1325d6a5fd2d76a7bc0ccb17a13bc8

Token Information

Token Name : Phoenix Chain

Token Symbol: PHX

Decimals: 18

Token Supply: 1,000,000,000

Token Address:
0x9776191F4ebBBa7f358C1663bF82C0a0906c77Fa
(Not deployed on chain)

Checksum:
8fac9d9cc46bb381e5fbd6dab5b11f761e4fdb0e

Owner:
0xE275535538dB0C5d5eC244aE736b675e91C080f8

TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 1%

Fees Privilige: None

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: yes

Other Priviliges: updating liquidity threshold - excluding from fees - including in 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

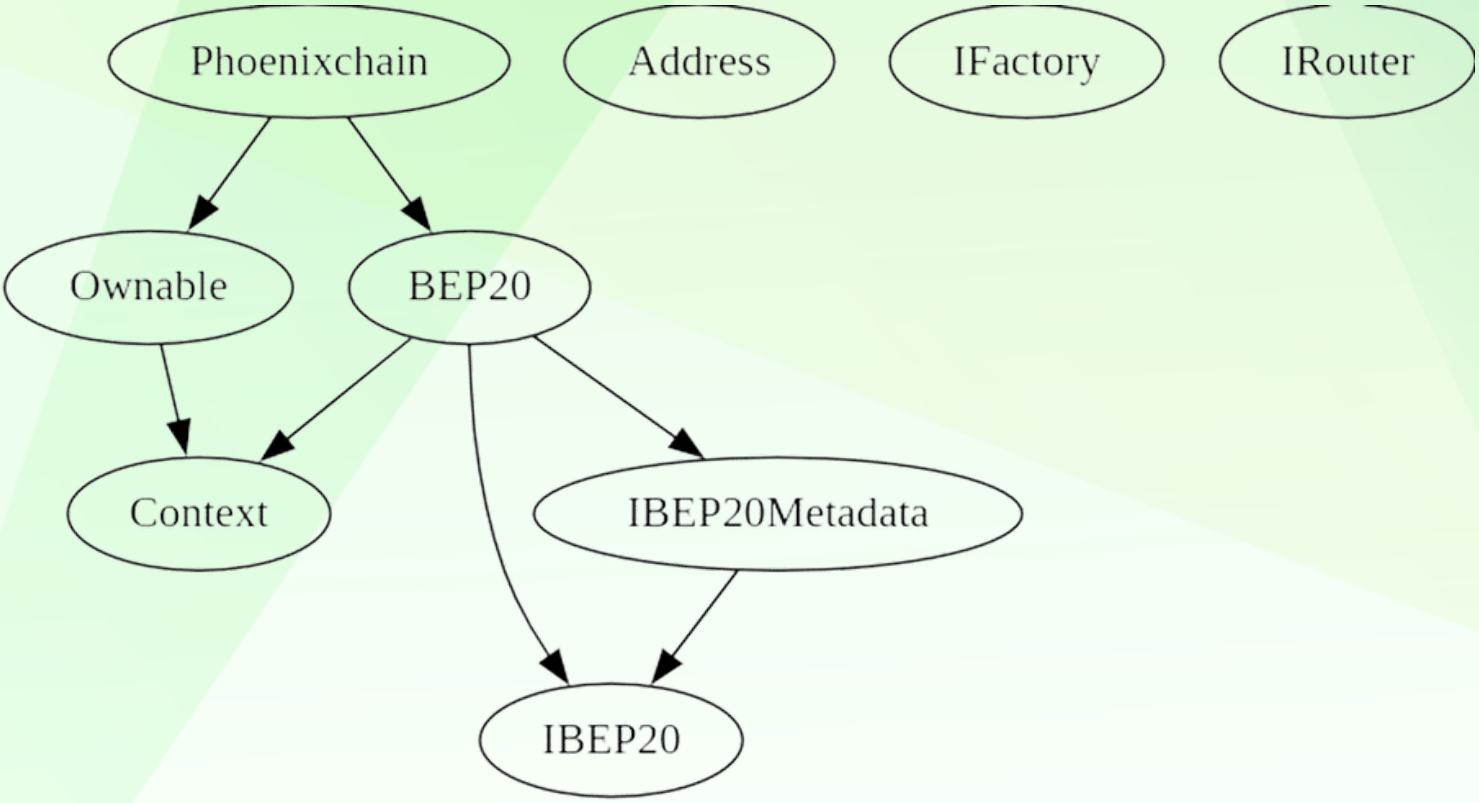
CLASSIFICATION OF RISK

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	1(Resolved)
High-Risk	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to modify fees (1% for buy/sell/transfer)
- Owner must enable trading for investors to be able to trade
- Owner is not able to disable trades
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to mint new tokens

Token Distribution

it should be noted that the owner 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

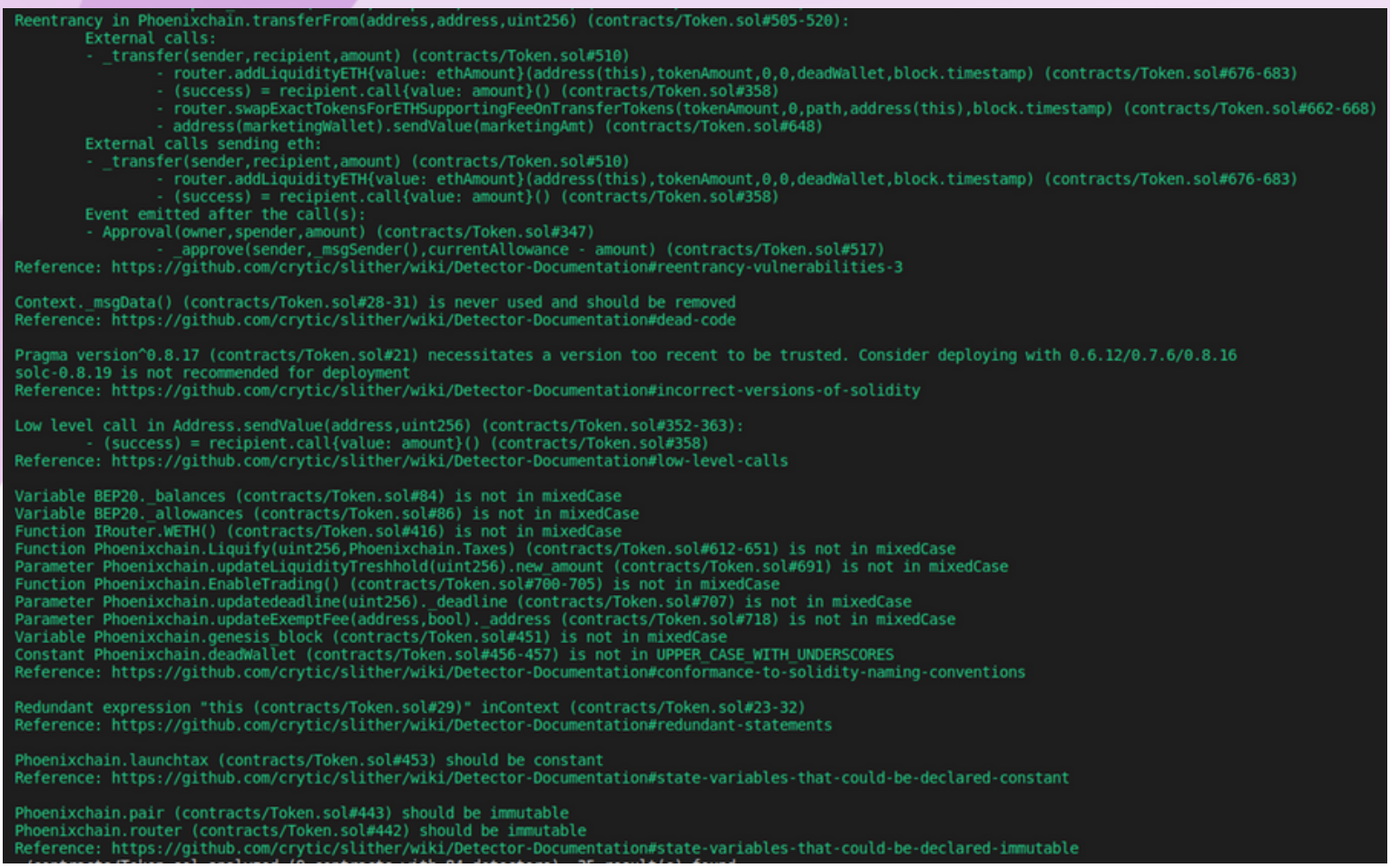
Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IBEP20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	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		

CONTRACT ASSESMENT

└	<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❗
Phoenixchain	Implementation	BEP20, Ownable		
└	<Constructor>	Public ❗	🛑	BEP20
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	transfer	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	Liquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForETH	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	updateLiquidityProvide	External ❗	🛑	onlyOwner
└	updateLiquidityTreshhold	External ❗	🛑	onlyOwner
└	EnableTrading	External ❗	🛑	onlyOwner
└	updatedeadline	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateExemptFee	External ❗	🛑	onlyOwner
└	bulkExemptFee	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueBSC20	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Reentrancy in Phoenixchain.transferFrom(address,address,uint256) (contracts/Token.sol#505-520):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#510)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/Token.sol#676-683)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#358)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#662-668)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/Token.sol#648)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#510)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/Token.sol#676-683)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#358)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#347)
                - _approve(sender, _msgSender(),currentAllowance - amount) (contracts/Token.sol#517)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/Token.sol#28-31) 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#21) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 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#352-363):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BEP20._balances (contracts/Token.sol#84) is not in mixedCase
Variable BEP20._allowances (contracts/Token.sol#86) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#416) is not in mixedCase
Function Phoenixchain.Liquify(uint256,Phoenixchain.Taxes) (contracts/Token.sol#612-651) is not in mixedCase
Parameter Phoenixchain.updateLiquidityTreshhold(uint256).new_amount (contracts/Token.sol#691) is not in mixedCase
Function Phoenixchain.EnableTrading() (contracts/Token.sol#700-705) is not in mixedCase
Parameter Phoenixchain.updatedeadline(uint256)._deadline (contracts/Token.sol#707) is not in mixedCase
Parameter Phoenixchain.updateExemptFee(address,bool)._address (contracts/Token.sol#718) is not in mixedCase
Variable Phoenixchain.genesis_block (contracts/Token.sol#451) is not in mixedCase
Constant Phoenixchain.deadWallet (contracts/Token.sol#456-457) 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#29)" inContext (contracts/Token.sol#23-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Phoenixchain.launchtax (contracts/Token.sol#453) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

Phoenixchain.pair (contracts/Token.sol#443) should be immutable
Phoenixchain.router (contracts/Token.sol#442) 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

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0xcc7323b612dfdaf6b97359830b86e97ed769d2573d70ca084dc2bdb3d24950c3

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x0a0fd11948542240d7eaf7e1d6140cf02768e3e2de2e3ef323667f5fbf40f960

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf5b33b673859b77ff7c9787aa00858ce7496bf718cfbf3993d448553a8361a17

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xc8c7f483a308bce33637f4c4c6c4e2528be614ff7b99f26cfc98ad81c3301735

5- Buying when not excluded (1% tax) (passed):
https://testnet.bscscan.com/tx/0x16aa896a05a88b62a4afc0835825cdcbac434e2d621b1eb4129c98bf16c0eb02

6- Selling when not excluded (1% tax) (passed):
https://testnet.bscscan.com/tx/0x32131458929a0a65dd8f4fe658c933fdcaab2afce40fab93211f970834aba662

FUNCTIONAL TESTING

7- Transferring when not excluded (1% tax) (passed):
https://testnet.bscscan.com/tx/0x2cc64244504d6efe669d824e5df0f10193647f57cb968c2dc1f1cf8642325cbf

8- Internal swap (passed):
Marketing wallet received ETH
https://testnet.bscscan.com/address/0x4d150dafe944ed0c917a739dcf5a0432f8791cbf#internaltx

MANUAL TESTING

Centralization - Owner must enable trading

Severity: High
Function: EnableTrading
Lines: 718
Status: Resolved
Overview:
The owner must activate trading for investors to buy, sell, or transfer tokens. If trading remains disabled, token holders will be unable to trade their tokens.

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

Since contract is owned by safu dev, enabling trades is guaranteed.

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