auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

The Walrus

DATED : 30 June 23'

AUDIT SUMMARY

Project name – The Walrus

Date: 30 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	1	0	1
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 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/0x6C3a03797378eAE216845CBc13a0b9F585f43F89

Token Information

Token Name : The Walrus

Token Symbol: Walrus

Decimals: 18

Token Supply:420,689,899,999,994

Token Address:
0x7D979670C525aE59630C09c903be0A0f493aBBa1

Checksum:
a51e880d2c55aa03361bb325ea5bc070f6d83e13

Owner:
0xb2aFf7Eb7fBfaf2d9d339139B70Fd63778b7cc60

Network: Ethereum

TOKEN OVERVIEW

Fees:

Buy Fees: 0-25%

Sell Fees: 0-25%

Transfer Fees: 0-25%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: none

Blacklist: No

Other Priviliges:

- Initial distribution of the tokens
- Modifying 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.

auditace.tech

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

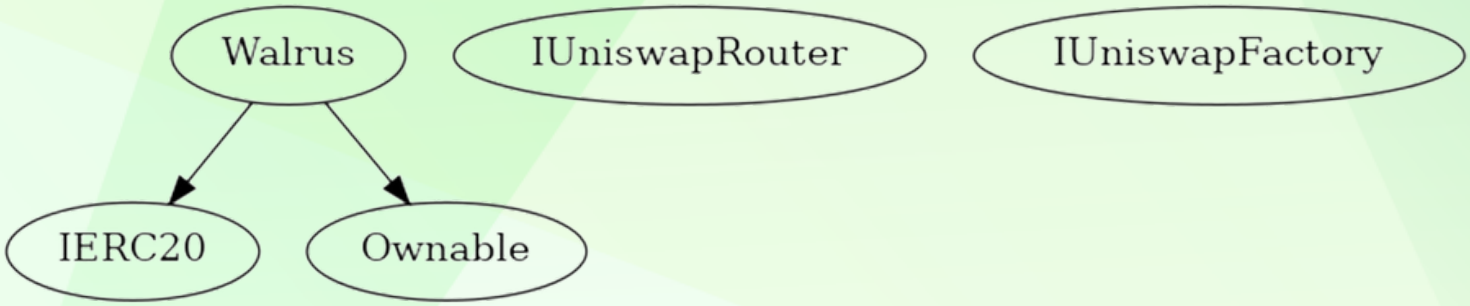
CLASSIFICATION OF RISK

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	1
Low-Risk	0
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to set buy/sell/transfer fees up to 25%
- 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

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IUniswapRouter	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
IUniswapFactory	Interface			
└	createPair	External ❗	🛑	NO❗
Ownable	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
Walrus	Implementation	IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	setFundAddr	Public ❗	🛑	onlyOwner
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	Design	Public ❗	🛑	onlyOwner

CONTRACT ASSESMENT

└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_transferToken	Private 🔐	🛑	
└	swapTokenForETH	Private 🔐	🛑	lockTheSwap
└	setIsExcludeFromFees	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Walrus.setFundAddr(address).newAddr (contracts/Token.sol#143) lacks a zero-check on :
                - fundAddress = newAddr (contracts/Token.sol#144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Walrus.transferFrom(address,address,uint256) (contracts/Token.sol#181-187):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#182)
                - _uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(fundAddress),block.timestamp) (contracts/Token
.sol#258-262)
        State variables written after the call(s):
        - _allowances[sender][msg.sender] = _allowances[sender][msg.sender] - amount (contracts/Token.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Walrus._transfer(address,address,uint256) (contracts/Token.sol#202-226):
        External calls:
        - swapTokenForETH(numTokensSellToFund) (contracts/Token.sol#215)
                - _uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(fundAddress),block.timestamp) (contracts/Token
.sol#258-262)
        Event emitted after the call(s):
        - Transfer(sender,address(this),swapAmount) (contracts/Token.sol#244)
                - _transferToken(from,to,amount,takeFee,sellFlag) (contracts/Token.sol#225)
        - Transfer(sender,recipient,tAmount - feeAmount) (contracts/Token.sol#249)
                - _transferToken(from,to,amount,takeFee,sellFlag) (contracts/Token.sol#225)
Reentrancy in Walrus.swapTokenForETH(uint256) (contracts/Token.sol#254-263):
        External calls:
        - _uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(fundAddress),block.timestamp) (contracts/Token.sol#258
-262)
        Event emitted after the call(s):
        - catchEvent(0) (contracts/Token.sol#261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Pragma version^0.8.0 (contracts/Token.sol#7) allows old versions
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapRouter.WETH() (contracts/Token.sol#33) is not in mixedCase
Variable Ownable._owner (contracts/Token.sol#49) is not in mixedCase
Event WalruscatchEvent(uint8) (contracts/Token.sol#252) is not in CapWords
Function Walrus.Design(uint256,uint256) (contracts/Token.sol#189-195) is not in mixedCase
Variable Walrus._isExcludeFromFee (contracts/Token.sol#90) is not in mixedCase
Variable Walrus._uniswapRouter (contracts/Token.sol#94) is not in mixedCase
Variable Walrus._buyFundFee (contracts/Token.sol#101) is not in mixedCase
Variable Walrus._sellFundFee (contracts/Token.sol#102) is not in mixedCase
Variable Walrus._uniswapPair (contracts/Token.sol#104) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Walrus._decimals (contracts/Token.sol#88) should be immutable
Walrus._name (contracts/Token.sol#86) should be immutable
Walrus._symbol (contracts/Token.sol#87) should be immutable
Walrus._totalSupply (contracts/Token.sol#92) should be immutable
Walrus._uniswapPair (contracts/Token.sol#104) should be immutable
Walrus._uniswapRouter (contracts/Token.sol#94) 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/0x066265bd147f5885f1acc4eb2086d70c9aa7cc566430b6a49457a61f04561875

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x7316aff6c9deefa6cb65e067214d266470a1f820ff042522fa7b82639dff308e

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x5033fec2a4cd57c4af2490f81249fa989144c6376266e778715f22f3651ae175

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf554bf9290d02bc3dc676c269573ed7fe076166e4152a9c0b1ee4d60c36e08f8

5- Buying when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0xf6be7c11b5951b3513e492d7779e71fd7760557ae2d5b2df32fff6c45cdbf7e3

6- Selling when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x7172b3f0fb4cfe9d7c8216087b591e6a14f968da9eaeafa99ea703dc6d05aed3

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0xc8554a3a4f2437e0dcc53229c9195862635cd2faa97c72ce94e7cdd93b34847a

8-Internal swap (passed):
- BNB fee sent to fund address
https://testnet.bscscan.com/tx/0x7172b3f0fb4cfe9d7c8216087b591e6a14f968da9eaeafa99ea703dc6d05aed3

ISSUES FOUND

Centralization – Excessive fees

Severity: Medium

Status: Open

Overview:

Owner is able to set buy/sell/transfer fees each up to 25%

.

0 <= buy total fees <= 25%
0 <= sell total fees <= 25%
0 <= transfer total fees <= 25%

```
function Design(uint256 newbFee, uint256 newsFee) public onlyOwner {
    _buyFundFee = newbFee;
    _sellFundFee = newsFee;

    require(_buyFundFee <= 25, "too high");
    require(_sellFundFee <= 25, "too high");
}
```

Suggestion

Its suggested to keep buy/sell/transfer fees less than 10% each (according to pinksale safu criteria)

```
function Design(uint256 newbFee, uint256 newsFee) public onlyOwner {

    _buyFundFee = newbFee;

    _sellFundFee = newsFee;

    require(_buyFundFee <= 10, "too high");

    require(_sellFundFee <= 10, "too high");

}
```

Safu criteria:

https://docs.pinksale.finance/important/safu-contract

ISSUES FOUND

Missing logic – Stuck Tokens And ETH

Severity: Informational

Status: Open

Overview:

There are no functions to withdraw stuck ETH or ERC20 tokens from the contract. If tokens (ETH/ERC20) were sent to contract by mistake there wont be anyways to withdraw those tokens .

Suggestion

Its higly recommended to create a function for withdrawing ERC20 tokens and ETH from the contract

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