auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

YO-YO

DATED : 21 June 23'

AUDIT SUMMARY

Project name – YO-YO

Date: 21 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/0xc8E9417A09198aad959e75E5497E96d59a3772E1

Token Information

Token Name : YO-YO

Token Symbol: YOYO

Decimals: 18

Token Supply:1,866,000,000,000

Token Address:
0xC99D53f342659F4B20D096A20169f295744032E7

Checksum:
bb6b15334497db62b0cfed28a618605054b1b671

Owner:
0x66409AbeC0c54a6858334263d058B41D569F25FC

TOKEN OVERVIEW

Fees:

Buy Fees: 0-25%

Sell Fees: 0-25%

Transfer Fees: 0-25%

Fees Privilige: Owner

Ownership : 0x66409AbeC0c54a6858334263d058B41D569F25FC

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.

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

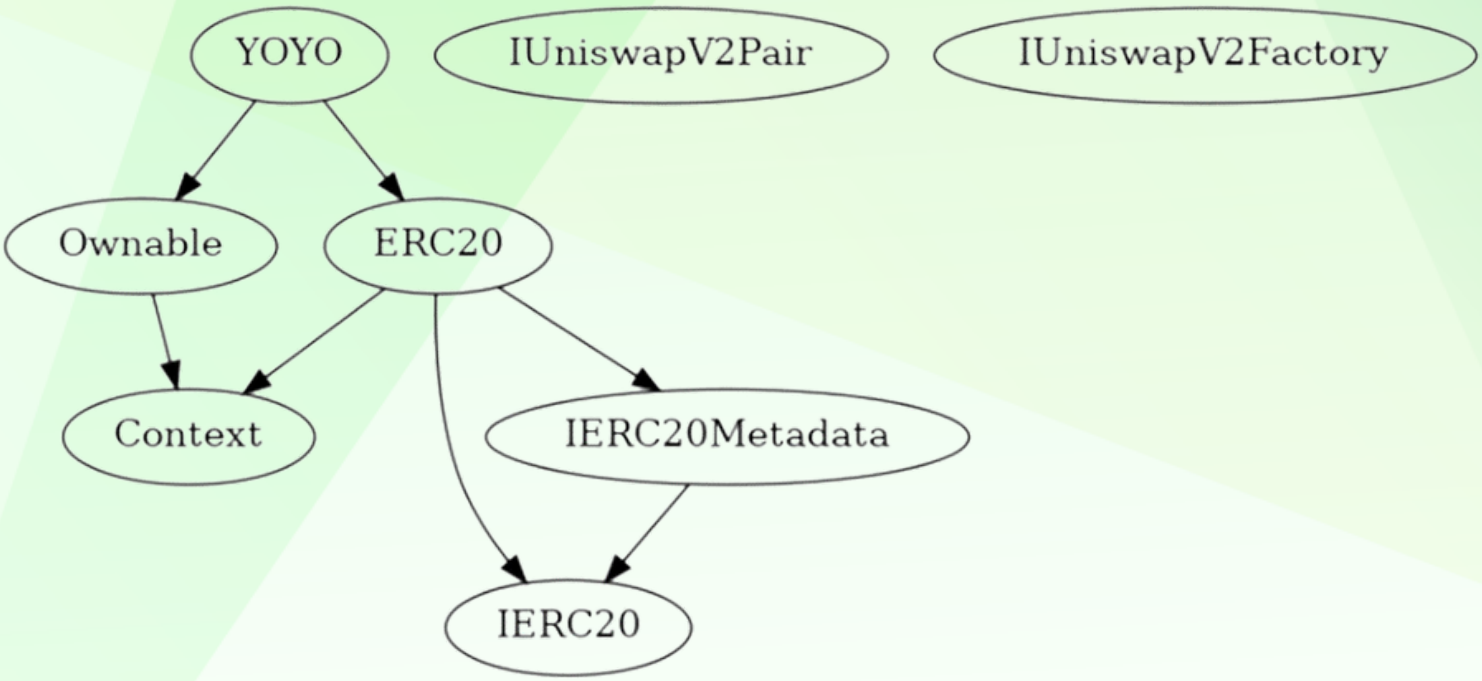
CLASSIFICATION OF RISK

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	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**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IUniswapV2Pair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗

CONTRACT ASSESMENT

└	allPairsLength	External ❗			NO❗
└	createPair	External ❗	🛑		NO❗
└	setFeeTo	External ❗	🛑		NO❗
└	setFeeToSetter	External ❗	🛑		NO❗
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❗
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 🔒	🛑		
└	_beforeTokenTransfer	Internal 🔒	🛑		
SafeMath	Library			
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			

CONTRACT ASSESMENT

└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
SafeMathInt	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	Library			
└	toInt256Safe	Internal 🔒		
IUniswapV2Router01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityETH	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapTokensForExactTokens	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗

CONTRACT ASSESMENT

IUniswapV2Router02	Interface	IUniswapV2Router01	
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑 |NO❗ |
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑 |NO❗ |
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑 |NO❗ |
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵 |NO❗ |
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑 |NO❗ |
YOYO	Implementation	ERC20, Ownable	
└	<Constructor>	Public ❗	🛑 | ERC20 |
└	<Receive Ether>	External ❗	💵 |NO❗ |
└	updateSwapTokensAtAmount	External ❗	🛑 | onlyOwner |
└	updateSwapEnabled	External ❗	🛑 | onlyOwner |
└	updateFees	External ❗	🛑 | onlyOwner |
└	excludeFromFees	Public ❗	🛑 | onlyOwner |
└	setAutomatedMarketMakerPair	Public ❗	🛑 | onlyOwner |
└	_setAutomatedMarketMakerPair	Private 🔐	🛑 | |
└	updateMarketingWallet	External ❗	🛑 | onlyOwner |
└	isExcludedFromFees	Public ❗	|NO❗ |
└	_transfer	Internal 🔒	🛑 | |
└	swapTokensForEth	Private 🔐	🛑 | |
└	addLiquidity	Private 🔐	🛑 | |
└	swapBack	Private 🔐	🛑 | |

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reentrancy in YOYO.swapBack() (contracts/Token.sol#1092-1120):
        External calls:
        - address(marketingWallet).transfer(ethForMarketing) (contracts/Token.sol#1115)
        External calls sending eth:
        - address(marketingWallet).transfer(ethForMarketing) (contracts/Token.sol#1115)
        - addLiquidity(liquidityTokens,ethForLiquidity) (contracts/Token.sol#1117)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadAddress,block.timestamp) (contracts/Token.sol#1082-1089)
        State variables written after the call(s):
        - addLiquidity(liquidityTokens,ethForLiquidity) (contracts/Token.sol#1117)
                - _allowances[owner][spender] = amount (contracts/Token.sol#427)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#428)
                - addLiquidity(liquidityTokens,ethForLiquidity) (contracts/Token.sol#1117)
        - SwapAndLiquify(amountToSwapForETH,ethForLiquidity,tokensForLiquidity) (contracts/Token.sol#1118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#716) is too similar to IUniswapV2Router01.addLiquid
ity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#717)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

YOYO.constructor() (contracts/Token.sol#922-960) uses literals with too many digits:
        - totalSupply = 1866000000000 * 1e18 (contracts/Token.sol#939)
YOYO.constructor() (contracts/Token.sol#922-960) uses literals with too many digits:
        - swapTokensAtAmount = 1000000 * 1e18 (contracts/Token.sol#941)
YOYO.updateSwapTokensAtAmount(uint256) (contracts/Token.sol#965-970) uses literals with too many digits:
        - require(bool,string)(newAmount >= totalSupply() * 1 / 100000,Swap amount cannot be lower than 0.001% total supply.) (contracts/Token.sol#966)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SafeMathInt.MAX_INT256 (contracts/Token.sol#644) is never used in SafeMathInt (contracts/Token.sol#642-699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

YOYO.maxWallet (contracts/Token.sol#883) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

YOYO.sellLiquidityFee (contracts/Token.sol#894) should be immutable
YOYO.sellMarketingFee (contracts/Token.sol#893) should be immutable
YOYO.sellTotalFees (contracts/Token.sol#892) 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/0x5ec3b0bab59018d449364b4044004fc4cc886af51c006d93754ed10f940550f6

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x9a19fe973ed86adb8a0de0a176bd7033efd4c1f56fd6c6c75a8fe2a4d2d17ce0

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x516107e47f395b85a5123dda670da1a61649974c0b36a4cf5f3e43b53cbf527e

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x513349278fd03e8c564344644f80c1c640cfbdef788972bb4b58f2c04b065895

5- Buying when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x9d4f4e5593b8bc7bc7ee42bf265bc68fd2acd098f060946b464771a28fa19a8c

6- Selling when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x9b545f17c49351d893974d2f3dee9df3cec71ab1c2b374a58dc8d7694aea354c

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-25% tax) (passed):
https://testnet.bscscan.com/tx/0x30247d371ec5ee8feaa83201daa0b82d3d7d871d45131d316fa7760e799c8253

8-Internal swap (passed):
- BNB fee sent to marketing wallet
- Auto liquidity sent to dead wallet
https://testnet.bscscan.com/address/0xf8cb87a074d8af61bc4f3dcda8c636e52e38347c#internaltx

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 updateFees(uint256 _marketingFee, uint256 _liquidityFee) external  onlyOwner {
    MarketingFee = _marketingFee;
    LiquidityFee = _liquidityFee;
    TotalFees = MarketingFee + LiquidityFee;
    require(TotalFees <= 250, "Must keep fees at 25% or less");
}
```

Suggestion

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

```
function updateFees(uint256 _marketingFee, uint256 _liquidityFee) external  onlyOwner {
    MarketingFee = _marketingFee;
    LiquidityFee = _liquidityFee;
    TotalFees = MarketingFee + LiquidityFee;
    require(TotalFees <= 100, "Must keep fees at 25% or less");
}
```

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 highly 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