



CYSRO

Cyber Security



SMART CONTRACT SECURITY AUDIT

Wojak

Table of Contents

Project Overview

Project Summary	2
Introduction	3
Scope of work	3

Audit Overview

Vulnerability severity information	4
Findings	5
Security Score	5
Severity chart	6
Function overview	8
Functional Flow diagram	12
Inheritance graph	13
Liquidity lock	13
Token Ownership renounced	13
Deployers actions	13
SWC Attacks	15
Test Results	16

Conclusion	18
-------------------------	-----------

Disclaimer	19
-------------------------	-----------

About.....	20
-------------------	-----------

Project Summary

Project name	WOJAK
Platform	Ethereum
Language	Solidity
Contract address	https://bscscan.com/address/0x62dd11e6a799eca6664827d2417c31dd7278b1c2#code
Repository	NA
Contract owner address	0xD72164B132CF19A75F65ecf137E0a8b5d5EaD62A
Deployers contract address	0xD72164B132CF19A75F65ecf137E0a8b5d5EaD62A
Decimal	9
Total supply	5000000000000000000000
Website	NA
Social media	NA
Audit methodology	Whitebox Testing
Delivery Date	July 20, 2021

Introduction

Given the opportunity to review WOJAK Project's smart contract source code, we in the report outline our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to launch after resolving the mentioned issues, there are no critical or high issues found related to business logic, security or performance.

Scope of work

The files that needed to be evaluated for the security assessment were given to us by the Team. The files listed below were used for this audit. Other files and contracts that are not listed here are not audited by us hence we will not be responsible for any security issues caused by those contracts.

	File	Checksum
1	0x62dD11e6a799ECa6664827D2417c31Dd7278b1c2	NA

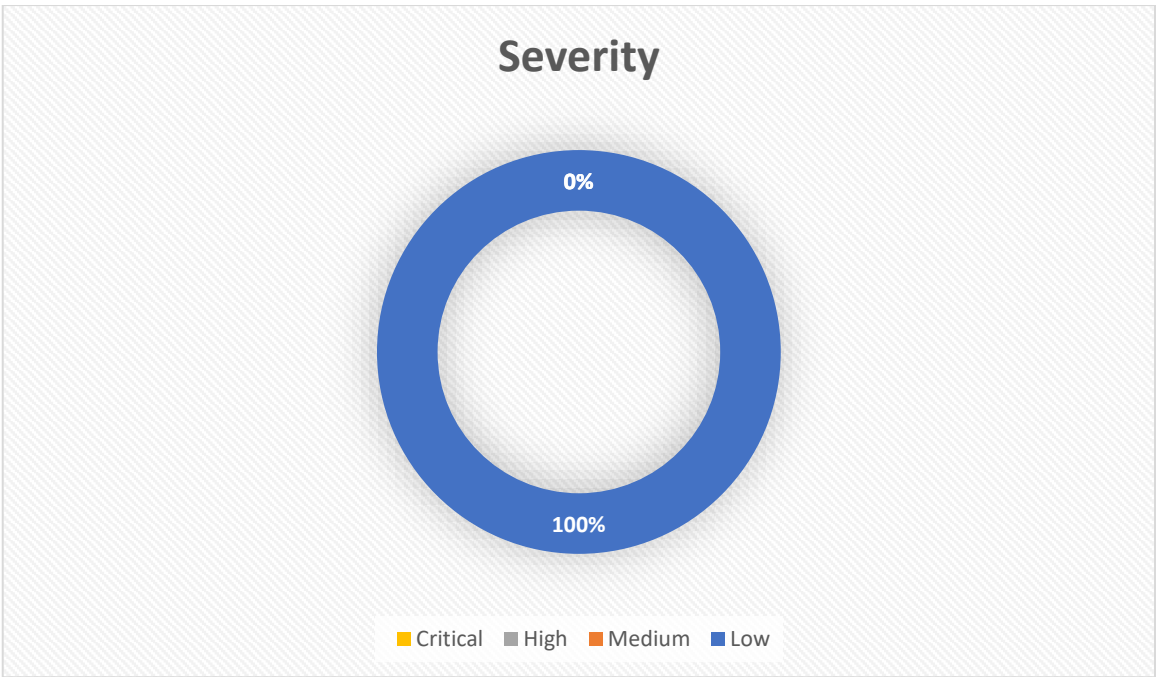
Vulnerability severity information



Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Informational	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Findings

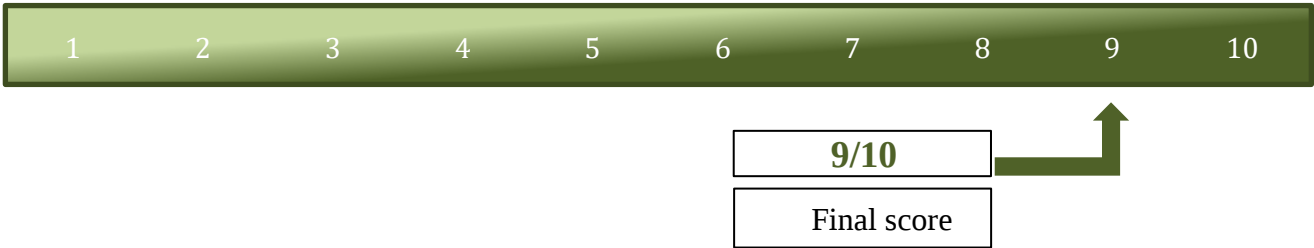
Total issues: 1



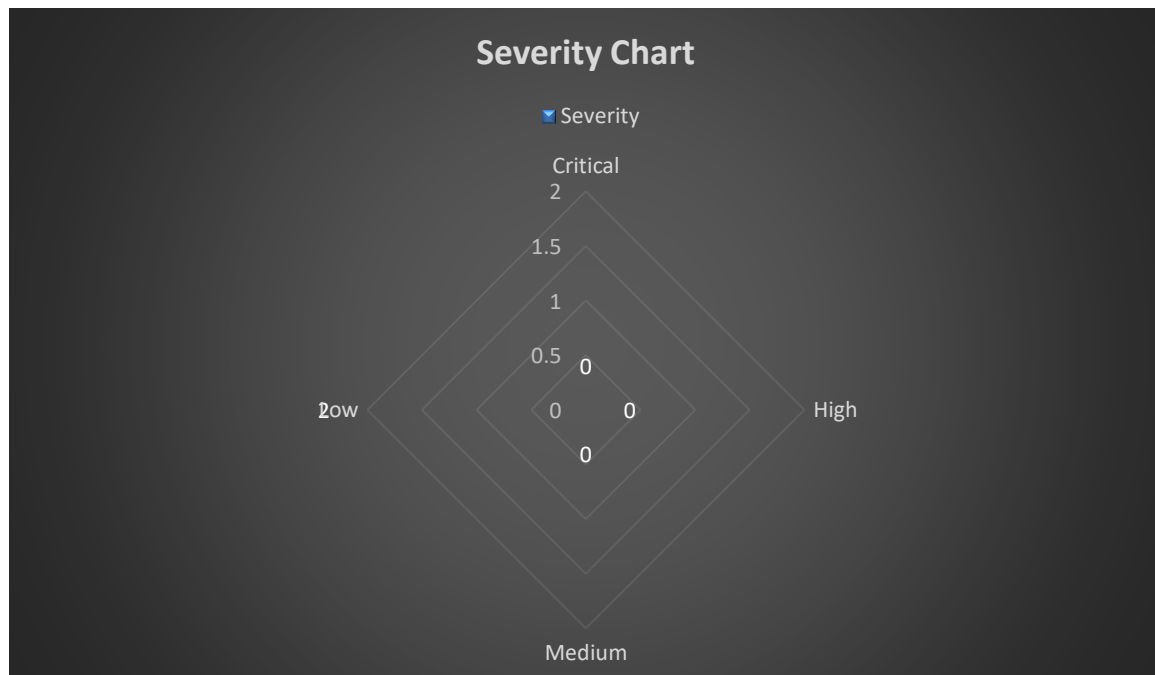
Critical	High	Medium	Low	Informational
0	0	0	1	0

Security Score

As a result of the audit, the code contains no issues. Therefore, the security score is 9/10.



Severity chart



We have so far identified that there are potential issues with severity of 0 Critical, 0 High, 0 Medium, and 1 Low. Overall, these smart contracts are well-designed and engineered.

1. Message call with hardcoded gas amount

Severity	Location	Classification	Status
Low	https://bscscan.com/address/0x62dd11e6a799eca6664827d2417c31dd7278b1c2#readContract		Open

Description

Call with hardcoded gas amount. The highlighted function call forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions. If this was done to prevent reentrancy attacks, consider alternative methods such as the checks-effects-interactions pattern or reentrancy locks instead.

```

545     function sendETHToDAO(uint256 amount) private {
546         _DaoWalletAddress.transfer(amount);
547     }

```

Relationships

CWE-655: Improper Initialization

Remediations

Avoid the use of `transfer()` and `send()` and do not otherwise specify a fixed amount of gas when performing calls. Use `.call.value(...)("")` instead. Use the checks-effects-interactions pattern and/or reentrancy locks to prevent reentrancy attacks.

Function overview

Contract	Type	Bases	Visibility	Mutability	Modifiers
Context	Function Name				
	Implementation				
	_msgSender	Internal			
IBEP20	_msgData	Internal			
	Interface				
	totalSupply	External			NO
	balanceOf	External			NO
	transfer	External		●	NO
	allowance	External			NO
	approve	External		●	NO
	transferFrom	External		●	NO
SafeMath	Library				
	add	Internal			
	sub	Internal			
	sub	Internal			
	mul	Internal			
	div	Internal			
	div	Internal			
	mod	Internal			
	mod	Internal			
	mod	Internal			
Address	Library				
	isContract	Internal			
	sendValue	Internal		●	
	functionCall	Internal		●	
	functionCall	Internal		●	
	functionCallWithValue	Internal		●	
	functionCallWithValue	Internal		●	
	_functionCallWithValue	Private		●	
	Implementation	Context			
Ownable	<Constructor>	Internal		●	
	owner	Public			NO
	renounceOwnership	Public		●	onlyOwner
	transferOwnership	Public		●	onlyOwner
	geUnlockTime	Public			NO
	lock	Public		●	onlyOwner
	unlock	Public		●	NO
IUniswap					
V2Factory	Interface				
	feeTo	External			NO
	feeToSetter	External			NO
	getPair	External			NO
	allPairs	External			NO
	allPairsLength	External			NO
	createPair	External		●	NO


**IUniswap
V2Pair**

setFeeTo	External		NO
setFeeToSetter	External		NO
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

**IUniswap
V2Router0
1**

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



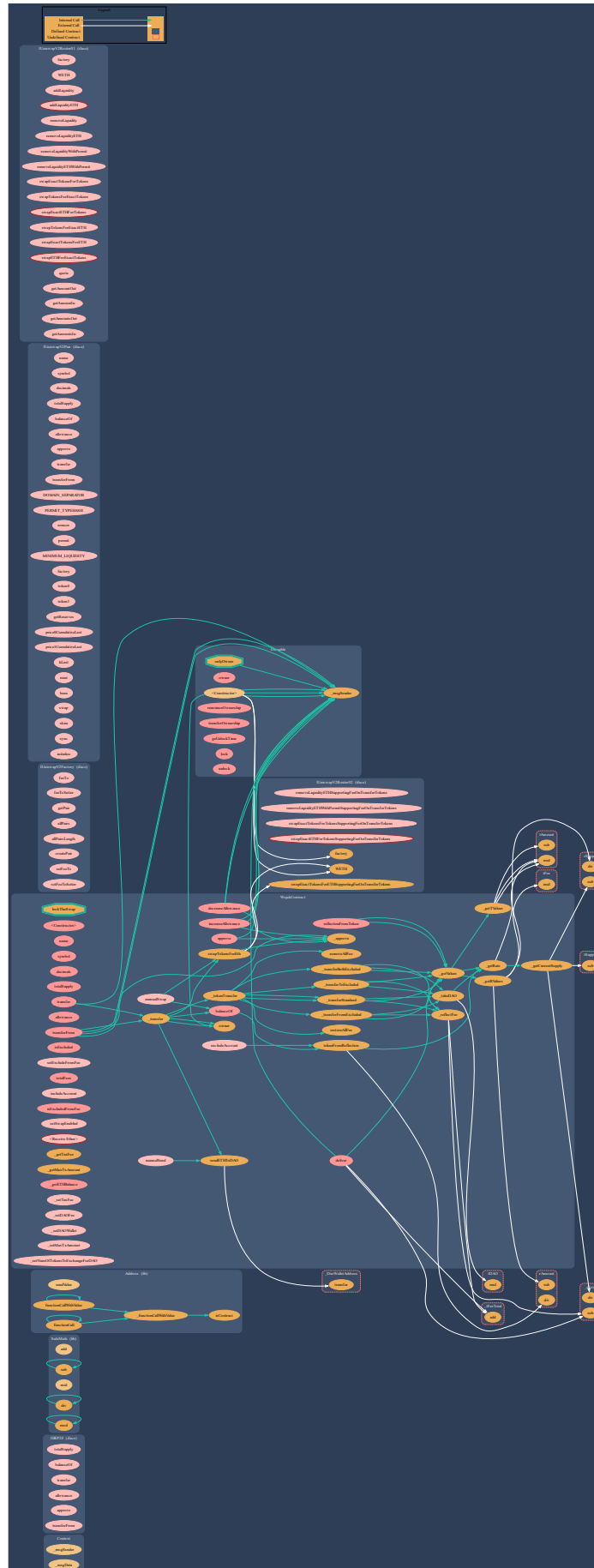
	getAmountIn	External		NO
	getAmountsOut	External		NO
	getAmountsIn	External		NO
IUniswap V2Router0 2	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	●	NO
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	●	NO
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	●	NO
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	●	NO
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	●	NO
WojakContract	Implementation	Context, IBEP20, Ownable		
	<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
	isExcluded	Public		NO
	setExcludeFromFee	External	●	onlyOwner
	totalFees	Public		NO
	deliver	Public	●	NO
	reflectionFromToken	Public		NO
	tokenFromReflection	Public		NO
	excludeAccount	External	●	onlyOwner
	includeAccount	External	●	onlyOwner
	removeAllFee	Private	●	
	restoreAllFee	Private	●	
	isExcludedFromFee	Public		NO
	_approve	Private	●	
	_transfer	Private	●	
	swapTokensForEth	Private	●	lockTheSwap
	sendETHToDAO	Private	●	
	manualSwap	External	●	onlyOwner
	manualSend	External	●	onlyOwner
	setSwapEnabled	External	●	onlyOwner
	_tokenTransfer	Private	●	

_transferStandard	Private		
_transferToExcluded	Private		
_transferFromExcluded	Private		
_transferBothExcluded	Private		
_takeDAO	Private		
_reflectFee	Private		
<Receive Ether>	External		NO
_getValues	Private		
_getTValues	Private		
_getRValues	Private		
_getRate	Private		
_getCurrentSupply	Private		
_getTaxFee	Private		
_getMaxTxAmount	Private		
_getETHBalance	Public		NO
_setTaxFee	External		onlyOwner
_setDAOFee	External		onlyOwner
_setDAOWallet	External		onlyOwner
_setMaxTxAmount	External		onlyOwner
_setNumOfTokensToExchangeForDAO	External		onlyOwner

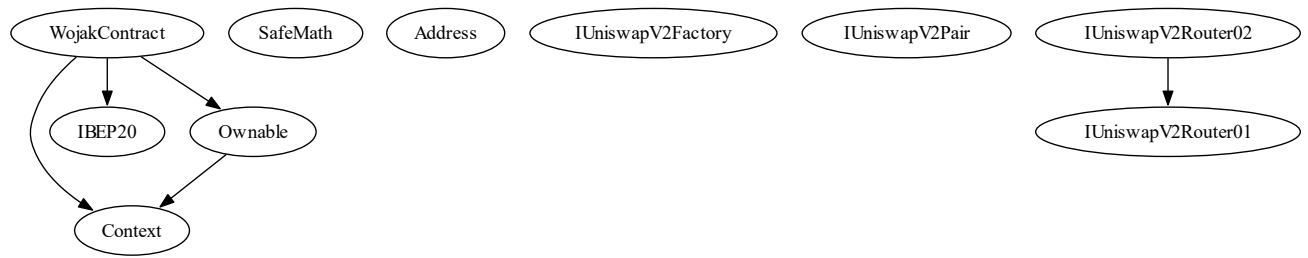
Where Symbol Meaning

- Function can modify state 
-  Function is payable

Functional Flow diagram



Inheritance graph



Liquidity lock

Liquidity locked period	Status
NO	NA

Token Ownership renounced

Token ownership Renounced	Status
NA	NA

Deployers actions

Can the deployer/owner mint a new token?	Status
NO	NA

Can the deployer/owner blacklist any wallet from selling?	Status
NO	NA

Can deployer/owner lock or burn user funds?	Status
Yes	NA

Can the deployer/owner pause the contract?	Status
NA	NA

Can the deployer/owner increase the fees?	Status
Yes	NA

SWC Attacks

Line	SWC	Severity	Description	Status
https://bscscan.com/address/0x62dd11e6a799eca6664827d2417c31dd7278b1c2#code	134	Low	Call with hardcoded gas amount. The highlighted function call forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions. If this was done to prevent reentrancy attacks, consider alternative methods such as the checks-effects-interactions pattern or reentrancy locks instead.	Open

Test Results

Slither results

NA

Mythx results

Report for WojakProject.sol https://dashboard.mythx.io/#/console/analyses/06ba69ff-6671-47b0-89e1-d0e3461f50bf			
Line	SWC Title	Severity	Short Description
7	(SWC-103) Floating Pragma	Low	A floating pragma is set.
32	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
41	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
48	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
49	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
57	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
67	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
147	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
355	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "***" discovered
355	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
356	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
356	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
368	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
370	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "***" discovered
370	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
371	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
371	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "***" discovered
477	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
478	(SWC-110) Assert Violation	Unknown	Out of bounds array access
479	(SWC-101) Integer Overflow and Underflow	Unknown	Compiler-rewritable "<uint> - 1" discovered
479	(SWC-110) Assert Violation	Unknown	Out of bounds array access
479	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
534	(SWC-110) Assert Violation	Unknown	Out of bounds array access
535	(SWC-110) Assert Violation	Unknown	Out of bounds array access
649	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
650	(SWC-110) Assert Violation	Unknown	Out of bounds array access
651	(SWC-110) Assert Violation	Unknown	Out of bounds array access
652	(SWC-110) Assert Violation	Unknown	Out of bounds array access

Mythril results

```
root@sv-VirtualBox:/home/sv# myth analyze WojakProject.sol
The analysis was completed successfully. No issues were detected.
```

Linter results

NA

Conclusion

In this audit, we thoroughly analysed WOJAK's Smart Contract. The current code base is well organized but there are promptly some Low type of issues found in the first phase of Smart Contract Audit.

Meanwhile, we need to emphasize that smart contracts as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedback or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

Disclaimer

Cysro has analysed this smart contract in accordance with the best practices at the date of this report. This report is based on extensive methodological examination and analysis of code, in relation to the cyber security vulnerabilities, blockchain security, and cryptocurrency. The report only represents advice and remediations for clients to improve the quality of code while intending to diminish the inherent risks of blockchains. Cysro recommends conducting a bug bounty program to confirm a high level of security of this smart contract. Cysro does not provide any assurance of a complete bug-free contract.

While Cysro has given its best in conducting the analysis and producing this report, it is important to note that you should not rely on this report to make any decision for investment or involvement in any particular project. This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. Please conduct your own due diligence before investing in any asset. Cysro shall not be liable for any losses incurred in these cases.

The analysis of the security by Cysro is solely based on the smart contract. No other applications or functionalities were reviewed.

About

Cysro is a privately held London and India based cyber security and blockchain technology company. It is built by a team of ethical hackers to aid businesses in battling off cyberattacks.

We specialize in providing services of penetration testing, smart contract auditing, and know your customer. Our mission is to offer the best services possible with the right people, right methodology, right scope, and right report.

Our detailed audit reports shall assist you in comprehending your risk exposure, addressing security issues, and improving data security for your business.



SMART CONTRACT SECURITY AUDIT