Information Technology
NETWORK

@ITNetwork

SMART CONTRACT SECURITY AUDIT OF

META OF CLASH

Smart Contract Audits | Solidty Analysis | Solidty Development | KYC | Project Evulation

SUMMARY

Auditing Firm	ITNetwork
Client Firm	MOC
Language	Solidity
Mandatory Audit Check	Static, Software, Auto Intelligent & Manual Analysis
Final Report	Date August 16, 2022

Audit Summary

ITNetwork team has performed a line-by-line manual analysis and automated review of the smart contracts. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

MetaOfClash's smart contract source code has LOW RISK SEVERITY
MetaOfClash has PASSED the smart contract audit
MetaOfClash uses mint to generate governance tokens. The function is Owner.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit. Please note, only a number of MetaOfClash contracts from their repository are audited, the contracts make external calls and import various code packages to work effectively, ITNetwork does not provide explicit guarantee on the safety and security of these calls/packages.

Verify the authenticity of this report on ITNetwork's GitHub:
https://github.com/itnetworkk

TABLE OF CONTENTS

PROJECT OVERVIEW

ITNetwork was consulted by MetaOfClash to conduct the smart contract security audit of their solidity source codes

About MetaOfClash

Meta of Clash is a free-to-play, play-and-earn MMO strategy game. Welcome to the arena! Defeat your battle deck in fast-paced real-time matches! A time and multiplayer game featuring your favorite Clash characters.
The MetaOfClash team comprises experienced MOC developers, and seasoned blockchain product and project leads.

PROJECT	META OF CLASH
Blockchain	Binance Smart Chain
Language	Solidity
Contracts	0xE35F4e1Bed03A6fbc921E23489E2E70Da02a6693
Website	https://www.metaofclash.world/
Telegram	https://t.me/metaofclash
Twitter	https://twitter.com/metaofclash
Telegram Turkey	https://t.me/metaofclashtr
Instagram	https://www.instagram.com/metaofclash/

PROJECT LOGO

Solidity Source Codes Under Scope

v MasterChef.sol
v MOCToken.sol
v MOCVault.sol
v PancakeFactory.sol
v PancakePair.sol
v PancakeRouter.sol
v PancakeLibrary.sol
v SmartChefFactory.sol
v SmartChefInitializable.sol

SHA-1 Hash

SHA-1 Hash Solidity source codes are audited at hash #f7c2bef01d6fcfd87103b0f0cdf4241bb8d0789f

Validated by https:/www.it-network.tech

AUDIT SCOPE & METHODOLOGY

The scope of this report is to audit the smart contract source codes of MOC. ITNetwork has scanned the contract codes and reviewed the project for common vulnerabilities, exploits, hacks, and backdoors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category	
Smart Contract Vulnerabilities	v Re-entrancy v Unhandled Exceptions v Transaction Order Dependency v Integer Overflow v Unrestricted Action v Incorrect Inheritance Order v Typographical Errors v Requirement Violation
Source Code Review	v Ownership Takeover v Gas Limit and Loops v Deployment Consistency v Repository Consistency v Data Consistency v Token Supply Manipulation
Functional Assessment	v Access Control and Authorization v Operations Trail and Event Generation v Assets Manipulation v Liquidity Access

Validated by https:/www.it-network.tech

ITNETWORK'S ECHELON AUDIT STANDARD

The aim of ITNetwork's "Echelon" standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:

v Review of the specifications, sources, and instructions provided to ITNetwork to make sure we understand the size, scope, and functionality of the smart contract.
v Manual review of code, which is the process of reading source code line-byline to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:

v Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
v Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

v Slither
v Consensys MythX, Mythril
v SWC Registry v Solidity Coverage
v Open Zeppelin Code Analyzer
v Solidity Code Complier

Validated by https:/www.it-network.tech

ITNETWORK'S RISK CLASSIFICATION

Smart contracts are generally designed to manipulate and hold funds. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level vulnerabilities could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity
! Medium	This level vulnerabilities are should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level vulnerabilities can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Validated by https:/www.it-network.tech

SMART CONTRACT — STATIC ANALYSIS

Symbol	Meaning
	Function can be modified
	Function is payable
	Function is locked
	Function can be accessed
	Important functionality

gratorChef;o* I Interf ace I 1 1 1
L I m igrate I External I INO !
1 1 1 1 1
"*"""""-8 t- erCh-ef;o* I Implementation I Ownable I 1 I"
L <Const ructor> I Public I 1 INO I 1
L updateMultiplier I Public I I onlyOwner I
L updateRewa rdPerBlock I Externa l I I onlyOwner L poollength I
External I INO ! 1
"L ,;id<j I Public I I onlyOwner I"
L ill I Public I I onlyOwner I
L updateStak ingPool I Internal Ii I I 1 L setMigrator I Public I I
onlyOwner L m igrate I Public I INO ! 1
L m igrateCont rol I Public ! 1 I onlyOwner
L getMultiplier I Public I INO ! 1
I pendingCake I External I I NO ! 1
L massUpdatePoo ls I Public I INO !
L updatePool I Public I 1 INO I 1
L deposit I Public ! 1 INO I
L withd raw I Public I 1 INO I 1
L enterStak ing I Public I 1 INO I
L leaveStak ing I Public I 1 INO I
L emergencyWithd raw I Public I 1 INO I L saf eCakeTransf er I Internal
lill I I 1 L setDevAdd ress I Public I 1 INO I 1
L setDevSha re I Public I INO I
L setSha reTo I Public I 1 INO I 1
1 1 1 I
=tak-e]okeQ I Implementation I BEP20 I 1 I
L I 'mi n rl I Public I I onlyOwner I
L I saf eCakeTransf er I Public I I onlyOwner

Validated by https:/www.it-network.tech

burn I Publ1c I 4f INO I
del tes I Exterl\ill I INO I
del te I Exterl\ill I I • INO f I del teBySig I Exterl\ill I I • INO I (letCurrentVotes I Exterl\ill I I (NO I (JetPr1orvot es I Exterl\ill I I (NO I I
_del te I Interl\ill il • I I
_moveoel tes I Interl\ill.a I il I
_wr1te<:M-ckpoint I Interl\ill Q I •
saf e32 I Interl\ill il I I
I (letCl\ainld I Interl\ill il I I
11111
"-cakc e"" Y' • • IP.. I Implementat 1on I Ooll\ilble, Pausable 111"
I <Constructor> I Public I • INO I
"depos 1t I Exterl\ill I 4f I ""'4'1enNotPaused notContract w1t.l'MirawAll I Exterl\ill I • I notContract I"
"I\arvest I Exterl\ill f I 8 I notContract ""'4'1enNotPaused setActnin I Exterl\ill I I • I onlyCM'ler I"
set Treasury I Exterl\ill I I • I onlyCM'ler I
set Perf ormanceFee I Exterl\ill I I 8 I onlyAdt11JI setcallFee I Exterl\ill I I • I onlyAd111J1 I setWI.t r.drawf ee I Exterl\ill f I 8 I onlyAdt11JI I
setWitl'MirawFeePeriod I Exterl'lil I I • I onlyAdt11JI nc-Ml.thdraw; I Exterl\ill I I 8 I onlyAdt11JI I
ineaseTokensGetSt uck I Exterl'lill I I • I onlyAdt11JI
"I Exterl\ill I I 8 I 011lyAd1'1JI ""'4'1enNotPaused I"
".!!'.IJ lliei I Exterl\ill ! I 4f I onlyAdt11JI ""'4'1enPaused I c.alculat eHarvestcakeRewards I Exterl\ill I (NO I"
c.alculat eTotalPel'MiikeRewards I External I INO (JetPricePerFullSl\are I Exterrlal I I INO I
w1t.l'Miraw I Publ1c I I • I l'lOtContract I ava1lable I Public f I INO f I
balanceOf I Public f I INO f I
_earn I Interl\ill il 8 I I
_ iscont ract I Interl\ill il I I
11111
-PaftcakeFactorr- I Implementat on I IPancakeFactory 111
I I <Constructor> I Public I I • INO I I
I allPairsleth I Exterl\ill f I INO f I I createPa1r I Exterl\ill I • INO ! I
I set.FeeTo I Exterl\ill ! I • INO I
I 1e:tll.Lhca.F I Exterl\ill ! I 4f INO !
I (JetExcl\a*Fee I Exterl\ill I INO I I setFeeSl\are I Exterl\ill I I • INO I I
I set.FeeActn1n I Exterl\ill I I • INO I I
11111
"-PaftcakePai,._ Implementat ion I 1Panc.akePa 1r , PancakeERC20 111"
I (letReserves Public I I (NO f I
_saf eTransf er I Private lii I 4il I I
<Constructor> I Public I I • INO I

Validated by https:/www.it-network.tech

└	initialize	External		NO
└	_update	Private		
└	_mintFee	Private		
└	mint	External		lock
└	burn	External		lock
└	swap	External		lock
└	skim	External		lock
└	sync	External		lock						
PancakeRouter	Implementation	IPancakeRouter02		
└	<Constructor>	Public		NO
└	<Receive Ether>	External		NO
└	_addLiquidity	Internal		
└	addLiquidity	External		ensure
└	addLiquidityETH	External		ensure
└	removeLiquidity	Public		ensure
└	removeLiquidityETH	Public		ensure
└	removeLiquidityWithPermit	External		NO
└	removeLiquidityETHWithPermit	External		NO
└	removeLiquidityETHSupportingFeeOnTransferTokens	Public		ensure
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External		NO		└	_swap	Internal		
└	swapExactTokensForTokens	External		ensure		└	swapTokensForExactTokens	External		ensure		└	swapExactETHForTokens	External		ensure
└	swapTokensForExactETH	External		ensure		└	swapExactTokensForETH	External		ensure		└	swapETHForExactTokens	External		ensure
└	_swapSupportingFeeOnTransferTokens	Internal		
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External		ensure		└	swapExactETHForTokensSupportingFeeOnTransferTokens	External		ensure
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External		ensure
└	quote	Public		NO
└	getAmountOut	Public		NO
└	getAmountIn	Public		NO
└	getAmountsOut	Public		NO
└	getAmountsIn	Public		NO						
PancakeLibrary	Library			
└	sortTokens	Internal		
└	pairFor	Internal		
└	getReserves	Internal		
└	quote	Internal		
└	getAmountOut	Internal		
└	getAmountIn	Internal		
└	getAmountsOut	Internal		
└	getAmountsIn	Internal							
SmartChefFactory	Implementation	Ownable		
└	<Constructor>	Public		NO

└	allPoolsLength	External		NO
└	deployPool	External		onlyOwner
SmartChefInitializable	Implementation	Ownable, ReentrancyGuard		
└	<Constructor>	Public		NO
└	initialize	External		NO
└	deposit	External		nonReentrant		└	withdraw	External		nonReentrant
└	emergencyWithdraw	External		nonReentrant
└	emergencyRewardWithdraw	External		onlyOwner
└	recoverWrongTokens	External		onlyOwner
└	stopReward	External		onlyOwner
└	updatePoolLimitPerUser	External		onlyOwner
└	updateRewardPerBlock	External		onlyOwner
└	updateStartAndEndBlocks	External		onlyOwner
└	pendingReward	External		NO
└	_updatePool	Internal		
└	_getMultiplier	Internal		

SMART CONTRACT — SOFTWARE ANALYSIS

Function Signatures

26465826 => setCallFee(uint256)
32749461 => getReserves(address,address,address)
74496190 => setMigrator(IMigratorChef)
ef171420 => migrate(IBEP20)
5ffe6146 => updateMultiplier(uint256)
01f8a976 => updateRewardPerBlock(uint256)
081e3eda => poolLength()
0812e2c5 => add(uint256,IBEP20,bool)
64482f79 => set(uint256,uint256,bool)
9b9c4477 => updateStakingPool()
454b0608 => migrate(uint256)
95f86d80 => migrateControl(address)
8dbb1e3a => getMultiplier(uint256,uint256)
1175a1dd => pendingCake(uint256,address)
630b5ba1 => massUpdatePools()
51eb05a6 => updatePool(uint256)
e2bbb158 => deposit(uint256,uint256)
441a3e70 => withdraw(uint256,uint256)
41441d3b => enterStaking(uint256)
1058d281 => leaveStaking(uint256)
5312ea8e => emergencyWithdraw(uint256)
a2e6ddcc => safeCakeTransfer(address,uint256)
d0d41fe1 => setDevAddress(address)
03c0fa01 => setDevShare(uint256)
bf1bbdae => setShareTo(address)
40c10f19 => mint(address,uint256)
42966c68 => burn(uint256)
587cde1e => delegates(address)
5c19a95c => delegate(address)
c3cda5202=>delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32)
b4b5ea57 => getCurrentVotes(address)
782d6fe1 => getPriorVotes(address,uint256)
a28a42b3 => _delegate(address,address)
955f9fd8 => _moveDelegates(address,address,uint256)
ee59e77f => _writeCheckpoint(address,uint32,uint256,uint256)
869d1f83 => safe32(uint256,string)
3408e470 => getChainId()
b6b55f25 => deposit(uint256)
853828b6 => withdrawAll()
4641257d => harvest()
704b6c02 => setAdmin(address)
f0f44260 => setTreasury(address)
70897b23 => setPerformanceFee(uint256)
b6ac642a => setWithdrawFee(uint256)
1efac1b8 => setWithdrawFeePeriod(uint256)

db2e21bc => emergencyWithdraw()
def68a9c => inCaseTokensGetStuck(address)
8456cb59 => pause()
3f4ba83a => unpause()
9d72596b => calculateHarvestCakeRewards()
58ebceb6 => calculateTotalPendingCakeRewards()
77c7b8fc => getPricePerFullShare()
2e1a7d4d => withdraw(uint256)
48a0d754 => available()
722713f7 => balanceOf() 6f48813d => _earn()
7d48441f => _isContract(address)
574f2ba3 => allPairsLength()
c9c65396 => createPair(address,address)
f46901ed => setFeeTo(address)
692eb56f => setExchangeFee(address,uint256)
5f4153f7 => getExchangeFee(address)
b3cba4a2 => setFeeShare(uint256)
6eb2d031 => setFeeAdmin(address)
0902f1ac => getReserves()
26e6cdde => _safeTransfer(address,address,uint256)
485cc955 => initialize(address,address)
7dc0a1fa => _update(uint256,uint256,uint112,uint112)
f65d5f86 => _mintFee(uint112,uint112)
6a627842 => mint(address)
89afcb44 => burn(address)
022c0d9f => swap(uint256,uint256,address,bytes)
bc25cf77 => skim(address) fff6cae9 => sync()
6d7746bc => _addLiquidity(address,address,uint256,uint256,uint256,uint256)
e8e33700 => addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256) f305d719 => addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde => removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) 02751cec => removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c=>removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes3 2,bytes32)
ded9382a=>removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt es32)
af2979eb=removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u int256,bool,uint8,bytes32,bytes32)
f5901d4d => _swap(uint256[],address[],address)
38ed1739 => swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee => swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5 => swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a => swapTokensForExactETH(uint256,uint256,address[],address,uint256)

18cbafe5 => swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41 => swapETHForExactTokens(uint256,address[],address,uint256)
d1c474e3 => _swapSupportingFeeOnTransferTokens(address[],address)
5c11d795swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95=>swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947=>swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
ad615dec => quote(uint256,uint256,uint256) 054d50d4 => getAmountOut(uint256,uint256,uint256)
85f8c259 => getAmountIn(uint256,uint256,uint256)
d06ca61f => getAmountsOut(uint256,address[])
1f00ca74 => getAmountsIn(uint256,address[])
544caa56 => sortTokens(address,address)
6d91c0e2 => pairFor(address,address,address)
cef496b1 => getAmountOut(address,address,uint256,uint256,uint256)
f1dfeabe => getAmountIn(address,address,uint256,uint256,uint256)
bb7b9c76 => getAmountsOut(address,uint256,address[])
192128b2 => getAmountsIn(address,uint256,address[])
efde4e64 => allPoolsLength()
eb8bfb05 => deployPool(IBEP20,IBEP20,uint256,uint256,uint256,uint256,address)
48cf8750 => initialize(IBEP20,IBEP20,uint256,uint256,uint256,uint256,address)
3279beab => emergencyRewardWithdraw(uint256)
3f138d4b => recoverWrongTokens(address,uint256)
80dc0672 => stopReward()
a0b40905 => updatePoolLimitPerUser(bool,uint256)
9513997f => updateStartAndEndBlocks(uint256,uint256)
f40f0f52 => pendingReward(address)
2061e1e5 => _updatePool()
8e356d7a => _getMultiplier(uint256,uint256)

SMART CONTRACT — SWC ATTACKS

SWC ID	Description	Verdict
SWC - 101	Integer Overflow and Underflow	Passed
SWC - 102	Outdated Compiler Version	Passed
SWC - 103	Floating Pragma	Passed
SWC - 104	Unchecked Call Return Value	Passed
SWC - 105	Unprotected Ether Withdrawal	Passed
SWC - 106	Unprotected SELFDESTRUCT Instruction	Passed
SWC - 107	Re-entrancy	Passed
SWC - 108	State Variable Default Visibility	Passed
SWC - 109	Uninitialized Storage Pointer	Passed
SWC - 110	Assert Violation	Passed
SWC - 111	Use of Deprecated Solidity Functions	Passed
SWC - 112	Delegate Call to Untrusted Callee	Passed
SWC - 113	DoS with Failed Call	Passed
SWC - 114	Transaction Order Dependence	Passed
SWC - 115	Authorization through tx.origin	Passed
SWC - 116	Block values as a proxy for time	Passed
SWC - 117	Signature Malleability	Passed
SWC - 118	Incorrect Constructor Name	Passed

Validated by https:/www.it-network.tech

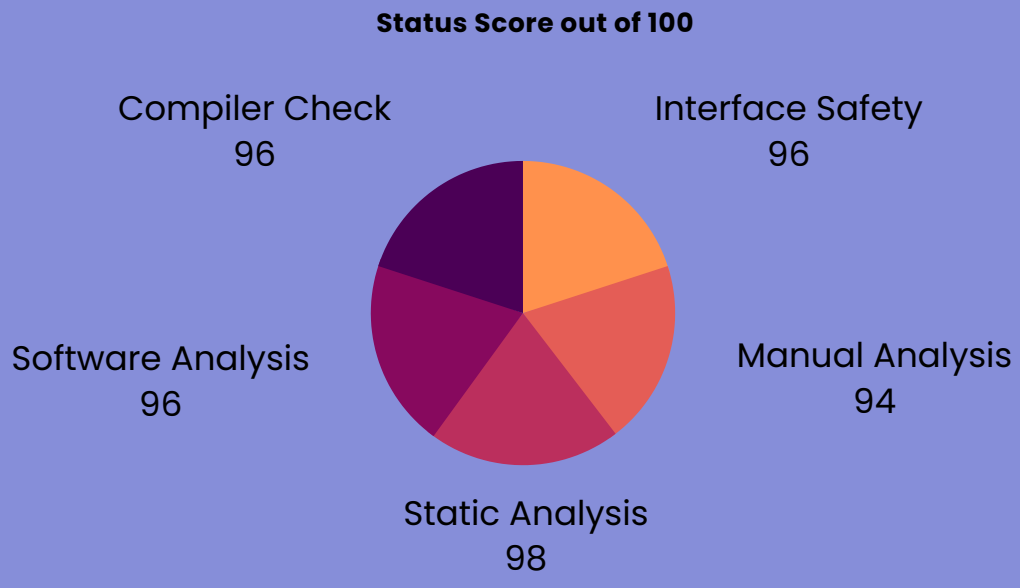
SMART CONTRACT — SWC ATTACKS

SWC ID	Description	Verdict
SWC - 119	Shadowing State Variables	Passed
SWC - 120	Weak Sources of Randomness from Chain Attributes	Passed
SWC - 121	Missing Protection against Signature Replay Attacks	Passed
SWC - 122	Lack of Proper Signature Verification	Passed
SWC - 123	Requirement Violation	Passed
SWC - 124	Write to Arbitrary Storage Location	Passed
SWC - 125	Incorrect Inheritance Order	Passed
SWC - 126	Insufficient Gas Griefing	Passed
SWC - 127	Arbitrary Jump with Function Type Variable	Passed
SWC - 128	DoS With Block Gas Limit	Passed
SWC - 129	Typographical Error	Passed
SWC - 130	Right-To-Left-Override control character (U+202E)	Passed
SWC - 131	Presence of unused variables	Passed
SWC - 132	Unexpected Ether balance	Passed
SWC - 133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC - 134	Message call with hardcoded gas amount	Passed
SWC - 135	Code With No Effects (Irrelevant/Dead Code)	Passed
SWC - 136	Unencrypted Private Data On-Chain	Passed

Validated by https:/www.it-network.tech

SMART CONTRACT – RISK STATUS & RADAR CHART

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	2 low severity issues identified
Verified	54 functions and instances verified and checked
Safety Score	98 out of 100



Validated by https:/www.it-network.tech

AUDITOR'S VERDICT

ITNetwork team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

MOC's smart contract source code has LOW RISK SEVERITY.

MOC has PASSED the smart contract audit.

Note for stakeholders

- **Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.**
- **Make sure that the project team's KYC/identity is verified by an independent firm, e.g., ITNetwork.**

Validated by https:/www.it-network.tech

IMPORTANT DISCLAIMER

ITNetwork provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without ITNetwork's prior written consent.

ITNetwork provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, ITNetwork does not guarantee the explicit security of the audited smart contract.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed

This report should not be considered as an endorsement or disapproval of any project or team. The information provided on this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

ABOUT ITNETWORK

ITNetwork provides intelligent blockchain solutions. ITNetwork is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. ITNetwork's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

ITNetwork is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. ITNetwork provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.

To learn more, visit https://it-network.tech
To view our audit portfolio, visit https://github.com/itnetworkk
To book an audit, message https://t.me/ITNetworkGlobal

@ITNetwork

Information Technology NETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA