Cyberscope

Audit Report

HODL

December 2024

Network : BSC

Address : 0x536bE24339B8a560e47179d191523A10f68Be4D6

Audited by © cyberscope

Analysis

🔴 Critical 🟡 Medium ⚪ Minor / Informative 🔵 Pass

Severity	Code	Description	Status
⚪	ST	Stops Transactions	Acknowledged
🔵	OTUT	Transfers User's Tokens	Passed
🔵	ELFM	Exceeds Fees Limit	Passed
🔵	MT	Mints Tokens	Passed
🔵	BT	Burns Tokens	Passed
🔵	BC	Blacklists Addresses	Passed

Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

Severity	Code	Description	Status
●	DRA	Dynamic Reward Allocation	Acknowledged
●	IMRA	Inconsistent Maximum Reward Amount	Acknowledged
●	MMRR	Missing Maximum Reward Restriction	Acknowledged
●	L13	Divide before Multiply Operation	Acknowledged

Table of Contents

Risk Classification

The criticality of findings in Cyberscope’s smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
● Critical	Highly Likely / High Impact
● Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
● Minor / Informative	Unlikely / Low to no Impact

Review

Contract Name	HODL
Compiler Version	v0.8.26+commit.8a97fa7a
Optimization	16777215 runs
Explorer	https://bscscan.com/address/0x536be24339b8a560e47179d191523a10f68be4d6
Address	0x536be24339b8a560e47179d191523a10f68be4d6
Network	BSC
Decimals	18

Audit Updates

Initial Audit	29 Nov 2024 https://github.com/cyberscope-io/audits/blob/main/hodl/v1/audit.pdf
Corrected Phase 2	05 Dec 2024 https://github.com/cyberscope-io/audits/blob/main/hodl/v2/audit.pdf
Corrected Phase 3	13 Dec 2024 https://github.com/cyberscope-io/audits/blob/main/hodl/v3/audit.pdf
Corrected Phase 4	23 Dec 2024

Source Files

Filename	SHA256
HODL_v1.06.sol	e89d5578d959523c0c6923677fbf9bedeae0d601232a4470e293766d4f23edae
HODLTypes.sol	eef155255804841da50c9ab9f30e0335b987a8ae58cd99e02c60e3b0893f59e7
HODLOwnableUpgradeable.sol	17ae387a751222bd84f4b7df4d2d8366bbdd5d9a0a6e07305da630b68ad0bd39

Overview

The HODL contract implements a token mechanism with staking and reward distribution functionalities. Users can claim rewards collected by the contract in the form of fees. Stakers can also claim from the reward pool by compounding their share. Additionally, users may choose to reinvest their rewards by swapping them for HODL tokens. Main functionalities include:

startStacking Function:

This function enables staking for a user and stakes all of their token balance except for 1 token. The contract checks that the user does not have an active staking session and that the balance exceeds a minimum threshold. If that is true, the user's balance is transferred to the `STACKING_ADDRESS` and the staking information is stored in a structure including the staked amount, the timestamp at the start of the staking and the claim period.

redeemRewards Function:

The contract allows a holder of the token to claim a percentage of the reward pool based on their token balance . Specifically, if the current balance of the contract exceeds a threshold, the user's rewards are calculated as a percentage of that threshold, proportional to their balance as a percentage of the circulating supply. Otherwise, if the contract's balance is less than the threshold, the former is used. It is important to notice that the user's balance is used for the calculation of rewards in the `redeemRewards`, rather than the deposited balance in the `STACKING_ADDRESS` which is used within the `stopStackingAndClaim` function. The `redeemRewards` function can therefore be called by external addresses that do not necessarily stake the token.

stopStackingAndClaim Function:

This function allows users to terminate their staking and claim their rewards. Rewards are calculated through a call to the `getStacked` function. Here the contract considers several cases.

If the contract holds more native tokens than the cap set for the claimable rewards, the latter is used in the calculations. In this case, rewards are estimated as a portion of the cap proportionally to the staked balance of the user as a percentage of the `rewardPoolShare`, multiplied by the integer of periods the user has been staking.

$$rewards = rewardCap * \frac{userStaked}{circulatingSupply} * stakedPeriods$$

In this calculation, it is possible that the rewards exceed the balance of the contract.

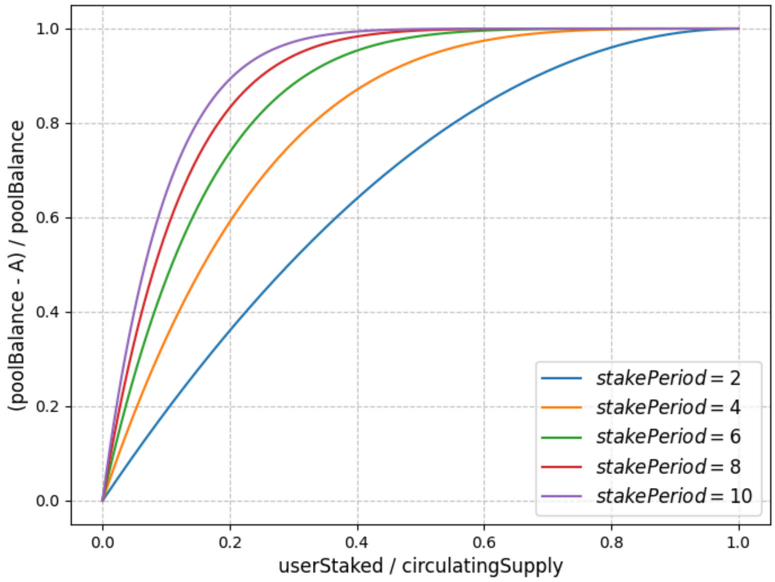
In this case, the rewards are re-evaluated by an interesting but rather complicated distribution mechanism which is inspired by the compound interest methodology. Specifically, the contract utilizes a series approximation to calculate the equation:

$$A = poolBalance * (1 - userStaked/circulatingSupply)^{stakedPeriods}$$

This calculation returns the portion of the pool balance that is inaccessible to the user. To estimate the portion that the user may claim, A is subtracted from the pool balance:

$$reward = poolBalance - A$$

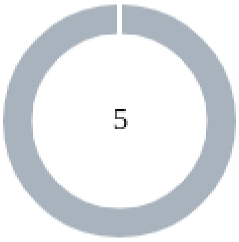
The latter estimates the claimable rewards as a function of the staked amount, the staking period and the pool size. In the following graph, the available rewards as a portion of the pool size are plotted for different magnitudes of staked amounts and staked durations:



In this figure, a larger portion of the pool becomes available, on the y-axis, as the staked amount or the staking period increases. Specifically, for a constant staked amount, more rewards are unlocked as time passes, while for the same staking period, larger staked amounts yield larger rewards.

Furthermore, these rewards are adjusted by a percentage of the pool amount that remains inaccessible proportionally to the user’s staked balance and the elapsed time from the current period. At the end of these calculations a hard threshold is applied on the estimated rewards ensuring that they cannot exceed a predefined limit, set at the start of the staking.

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	0	5	0	0

ST - Stops Transactions

Criticality	Minor / Informative
Location	HODL_v1.06.sol#L233,503
Status	Acknowledged

Description

The contract owner can set a maximum daily maximum sell limit of 0.25% of the total supply. Consequently, users will be restricted from selling more than this amount within a single day.

```
function changeMaxSellAmount(
uint256 newValue
) external onlyOwner onlyPermitted {
if (
newValue < (super.totalSupply() * 25) / 10_000 ||
newValue > (super.totalSupply() * 500) / 10_000
) revert ValueOutOfRange();
uint256 oldValue = maxSellAmount;
maxSellAmount = newValue;
emit ChangeValue(oldValue, newValue, "maxSellAmount");
}
```

```
// Ensures daily sell limit is enforced for each user
function ensureMaxSellAmount(address from, uint256 amount) private
{
WalletAllowance storage wallet = userWalletAllowance[from];

// Reset daily sell allowance if 24 hours have passed since last
transaction
if (block.timestamp > wallet.lastTransactionTimestamp + 1 days) {
wallet.lastTransactionTimestamp = 0;
wallet.dailySellVolume = 0;
}

uint256 totalAmount = wallet.dailySellVolume + amount;
if (totalAmount > maxSellAmount) revert ExceededDailySellLimit();

// Update daily allowance tracking
if (wallet.lastTransactionTimestamp == 0) {
wallet.lastTransactionTimestamp = block.timestamp;}
wallet.dailySellVolume = totalAmount;}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

Team Update

The team has acknowledged that this is not a security issue and states:

To ensure sustainable rewards we limit daily sales per wallet to this amount which is 0.25% of the total supply. Private keys are kept offline in cyphers. The contract uses multi sig, requiring 2 of 3 signers.

DRA - Dynamic Reward Allocation

Criticality	Minor / Informative
Location	HODL_v1.06.sol#L371
Status	Acknowledged

Description

The contract distributes rewards from a pool of accumulated fees. These rewards are finite and can be claimed at any time by any eligible address. At the time of the claim, rewards are calculated based on the available pool balance or a cap value. In the first case, race conditions and optimization points are induced where users are incentivized to claim rewards before the pool is depleted. This may disincentivize users from compounding their rewards through longer deposits to the `STACKING_ADDRESS` . While it could also result in unexpected redeemable rewards as the pool balance changes over time.

```
if (initialBalance >= tmpStack.rewardPoolCapAtStart) {
    reward =(((uint256(tmpStack.rewardPoolCapAtStart) *
    tmpStack.stackedAmount) / currentRewardPoolShare)
*stackedTotal) / 1E6;
if (
    reward >= initialBalance ||
    initialBalance - reward < tmpStack.rewardPoolCapAtStart
) {
    reward = _calculateStackedReward(
    initialBalance,
    tmpStack,
    stacked,
    rest,
    currentRewardPoolShare);
    }
}
else {
    reward = _calculateStackedReward(
    initialBalance,
    tmpStack,
    stacked,
    rest,
    currentRewardPoolShare
);
}
```

Recommendation

Implementing an economic design that ensures proportional distribution according to the staked balance and period, while guaranteeing that the expected amount accumulates independently of changes in the reward balance, will enhance consistency and user trust in the system. The team is advised to implement a mechanism that updates the user's rewards at the time the balance is updated.

Team Update

The team has acknowledged that this is not a security issue and states:

Users have the option to stack their rewards or claim weekly. There are advantages for those stacking and it is limited to 0.1 BNB as it is for smaller investors.

IMRA - Inconsistent Maximum Reward Amount

Criticality	Minor / Informative
Location	HODL_v1.06.sol#L172
Status	Acknowledged

Description

The contract implements the `redeemRewards` function, which allows the caller to redeem rewards from the contract's reward reserves. In this calculation, the user's amount is estimated based on the caller's current balance, not the actual staked amount. Rewards are calculated proportionally to the balance as a function of the circulating supply. This approach disincentivizes users from staking in the contract, as rewards can be claimed simply by holding tokens. Additionally, it allows users to transfer funds between their own accounts to exploit favorable staking claim dates without triggering the calculation of a `newCycleBlock` in the new wallet.

```
function redeemRewards(uint8 perc) external nonReentrant {
    if (perc > 100) revert ValueOutOfRange();
    uint256 userBalance = super.balanceOf(msg.sender);
    if (nextClaimDate[msg.sender] > block.timestamp)
    revert ClaimPeriodNotReached();
    if (userBalance == 0) revert NoHODLInWallet();
    uint256 currentBNBPool = address(this).balance;
    uint256 reward = currentBNBPool > bnbRewardPoolCap
    ? (bnbRewardPoolCap * userBalance) / rewardPoolShare
    : (currentBNBPool * userBalance) / rewardPoolShare;
    executeRedeemRewards(perc, reward);
}
```

Recommendation

It is advisable to ensure that rewards are claimable only by staked accounts to maintain consistency and fairness in the reward distribution mechanism. Additionally, ensuring the proper application of claiming restrictions across all addresses will prevent potential manipulation of the reward pool.

MMRR - Missing Maximum Reward Restriction

Criticality	Minor / Informative
Location	HODL_v1.06.sol#L172
Status	Acknowledged

Description

The contract implements two methods for withdrawing rewards from the pool's reserves: by calling the `redeemRewards` function or by using the `stopStackingAndClaim` function to withdraw staked amounts and claim available rewards. In the latter case, a maximum reward amount is always applied, equal to a `rewardLimit` initialized at the start of staking. However, in the first case, this limit is not imposed, allowing users to claim rewards exceeding the limit.

```
function redeemRewards(uint8 perc) external nonReentrant {
    if (perc > 100) revert ValueOutOfRange();
    uint256 userBalance = super.balanceOf(msg.sender);
    if (nextClaimDate[msg.sender] > block.timestamp)
    revert ClaimPeriodNotReached();
    if (userBalance == 0) revert NoHODLInWallet();
    uint256 currentBNBPool = address(this).balance;
    uint256 reward = currentBNBPool > bnbRewardPoolCap
    ? (bnbRewardPoolCap * userBalance) / rewardPoolShare
    : (currentBNBPool * userBalance) / rewardPoolShare;
    executeRedeemRewards(perc, reward);
}
```

Recommendation

The team is advised to ensure consistency between the different claiming mechanisms. Calculating rewards using the same formula and restrictions will ensure the smooth operation of the rewards pool.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	HODL_v1.06.sol#L380,381,388,435,671,675,698,718,723,746,751
Status	Acknowledged

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 stacked = stackedTotal / 1E6
uint256 rest = stackedTotal - (stacked * 1E6)
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

Functions Analysis

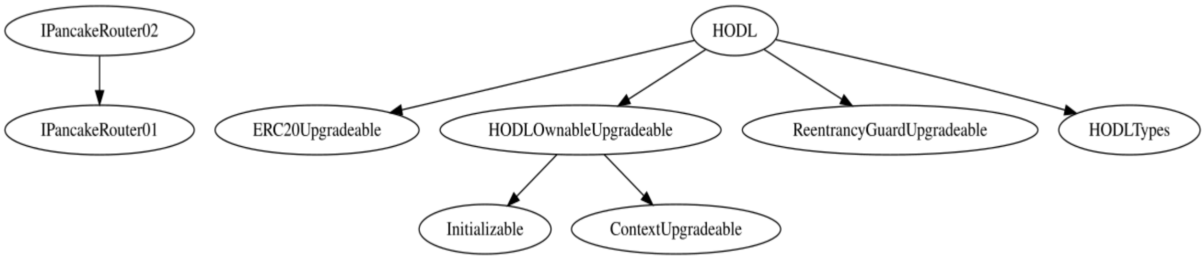
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
HODL	Implementation	ERC20Upgradeable, HODLOwnableUpgradeable, ReentrancyGuardUpgradeable, HODLTypes		
		External	Payable	-
	stopStackingAndClaim	External	✓	nonReentrant
	startStacking	External	✓	-
	redeemRewards	External	✓	nonReentrant
	updateIsTaxFree	External	✓	onlyOwner onlyPermitted
	excludeFromRewardPoolShare	External	✓	onlyOwner onlyPermitted
	changeBuyTaxes	External	✓	onlyOwner onlyPermitted
	changeSellTaxes	External	✓	onlyOwner onlyPermitted
	changeMaxSellAmount	External	✓	onlyOwner onlyPermitted
	changeMinTokensTriggerRewardSwap	External	✓	onlyOwner onlyPermitted
	changeSwapForRewardThreshold	External	✓	onlyOwner onlyPermitted
	changeBnbRewardPoolCap	External	✓	onlyOwner onlyPermitted

	changeRewardClaimPeriod	External	✓	onlyOwner onlyPermitted
	changeUpdateClaimDateRate	External	✓	onlyOwner onlyPermitted
	changeReinvestBonusCycle	External	✓	onlyOwner onlyPermitted
	changeBuySellCooldown	External	✓	onlyOwner onlyPermitted
	updatePairAddress	External	✓	onlyOwner onlyPermitted
	updateMMAddress	External	✓	onlyOwner onlyPermitted
	triggerSwapForReward	External	✓	lockTheSwap
	getCurrentBNBReward	External		-
	getRewardPoolShare	Public		-
	getTokensValue	Public		-
	getStacked	Public		-
	_calculateStackedReward	Internal		
	_update	Internal	✓	
	updateRewardPoolShare	Private	✓	
	ensureMaxSellAmount	Private	✓	
	executeRedeemRewards	Private	✓	
	updateClaimDateAfterTransfer	Private	✓	
	swapForReward	Private	✓	lockTheSwap
	getTokensToSell	Private	✓	
	swapTokensForEth	Private	✓	
	calculateUpdateClaim	Private		

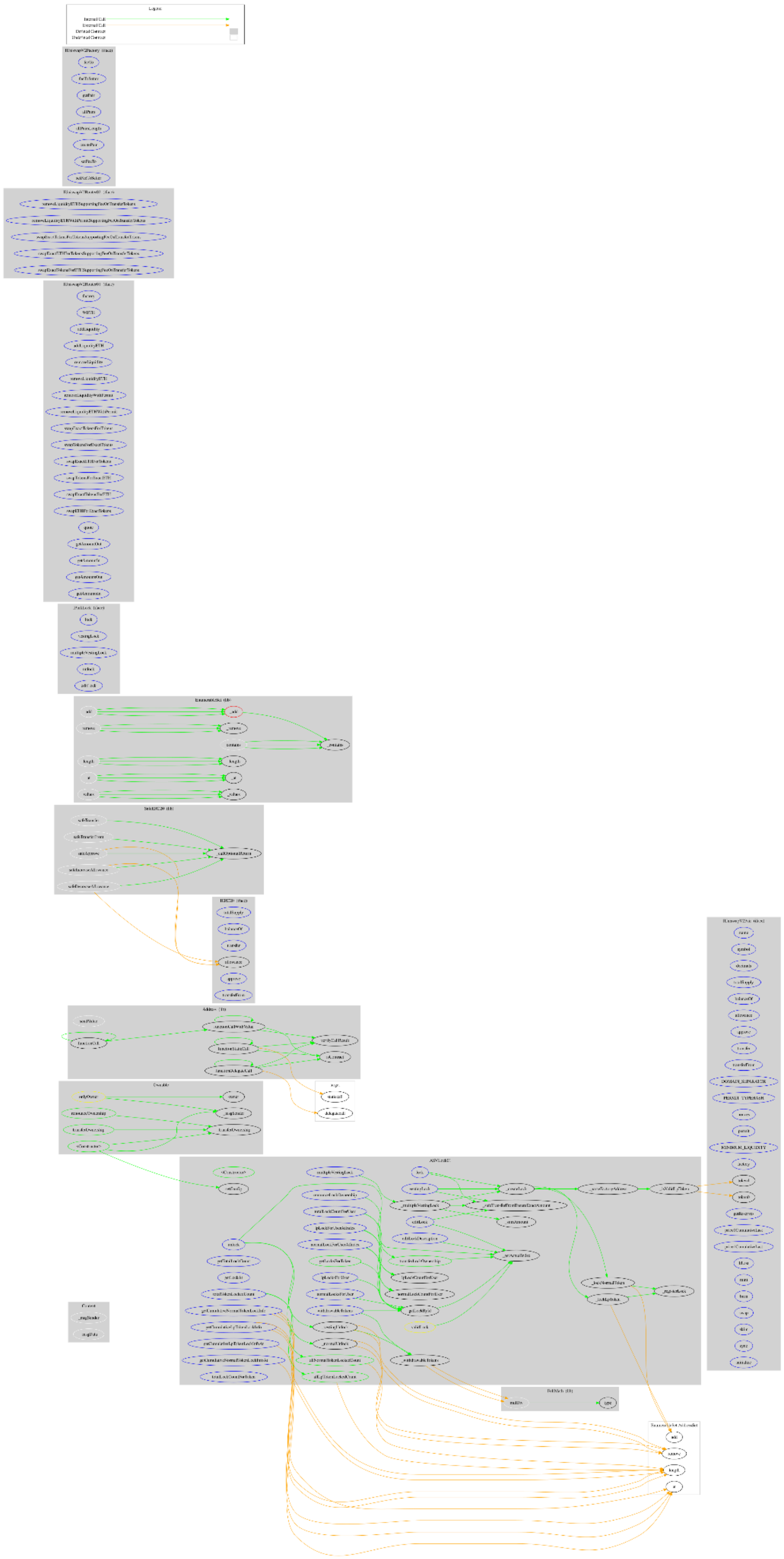
	calculateReward	Private		
HODLTypes	Implementation			
HODLOwnableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	_getOwnableStorage	Private		
	__Ownable_init	Internal	✓	onlyInitializing
	__Ownable_init_unchained	Internal	✓	onlyInitializing
	owner	Public		-
	owner2	Public		-
	owner3	Public		-
	permittedBy	Public		-
	permittedTo	Public		-
	permittedAt	Public		-
	_isOwner	Internal		
	_checkOwner	Internal		
	_checkPermission	Internal		
	_cancelPermission	Internal	✓	
	givePermission	External	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner onlyPermitted
	transferOwner2	Public	✓	onlyOwner onlyPermitted

	transferOwner3	Public	✓	onlyOwner onlyPermitted
	_transferOwnership	Internal	✓	
	_transferOwner2	Internal	✓	
	_transferOwner3	Internal	✓	

Inheritance Graph



Flow Graph


lock
multipleVestingLock
unlock
add
remove
contains
length
at
values
_add
_remove
_contains
_length
_at
_values
safeTransfer
safeTransferFrom
safeApprove
safeIncreaseAllowance
safeDecreaseAllowance
_callOptionalReturn
totalSupply

Summary

HODL contract implements a token and staking mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions. The team has acknowledged the findings. Renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope’s prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

cyberscope.io