Cyberscope

Audit Report

HODL

October 2025

Network : BSC

Address : 0x248C42D8a38f8C65a98ec401384c31f8bc891A6F

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	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
	IMRA	Inconsistent Maximum Reward Amount	Unresolved
	MMRR	Missing Maximum Reward Restriction	Unresolved
	L13	Divide before Multiply Operation	Unresolved

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/0x248c42d8a38f8c65a98ec401384c31f8bc891a6f
Address	0x248c42d8a38f8c65a98ec401384c31f8bc891a6f
Network	BSC
Decimals	18

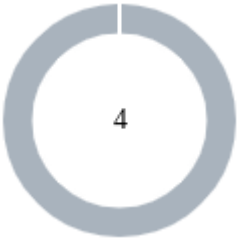
Audit Updates

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 https://github.com/cyberscope-io/audits/blob/main/hodl/v4/audit.pdf
Corrected Phase 5	07 Oct 2025

Source Files

Filename	SHA256
HODL_v1.14.sol	d6af5dec063d5630a7db99a3fd106f6f2bc242f43a56dd46d086e09b175c3f2f
HODLTypes.sol	1fd19ef87cdaec1dec087e08f34080deaf7e238ea53d9d044c769ff2b571a587
HODLOwnableUpgradeable.sol	17ae387a751222bd84f4b7df4d2d8366bbdd5d9a0a6e07305da630b68ad0bd39

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

ST - Stops Transactions

Criticality	Minor / Informative
Location	HODL_v1.14.sol#L263,420,464
Status	Unresolved

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.

```shell
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");

}
```

```shell
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.

IMRA - Inconsistent Maximum Reward Amount

Criticality	Minor / Informative
Location	HODL_v1.14.sol#L146
Status	Unresolved

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 allows users to transfer funds between their own accounts to exploit favorable claiming conditions without triggering the calculation of a `newCycleBlock` in the new wallet. If inconsistent fees, claiming periods or address exclusions are set, it may become economically viable to manipulate the reward pool.

```shell
function redeemRewards(uint8 perc, address token) 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;
...

}
```

Recommendation

It is advisable to ensure proper claiming restrictions and transaction fees across all addresses to disincentivize potential manipulation of the reward pool.

MMRR - Missing Maximum Reward Restriction

Criticality	Minor / Informative
Location	HODL_v1.14.sol#L146
Status	Unresolved

Description

The contract implements the `redeemRewards` function which allows users to withdraw rewards from the pool's reserves. However, this function does not enforce a maximum reward limit per user. As a result, users may be able to claim rewards that exceed the intended limits, potentially affecting the reward pool balance.

```shell
function redeemRewards(uint8 perc, address token) 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;
...

}
```

Recommendation

The team is advised to ensure consistency by introducing a maximum claim amount per user for every reward period. This approach helps maintain predictable reward distribution and prevents any single user from disproportionately claiming rewards.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	HODL_v1.14.sol#L156,173,588,592
Status	Unresolved

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.

```
Shell
rewardBNB = (reward * perc) / 100
uint256 reward = currentBNBPool > bnbRewardPoolCap
? (bnbRewardPoolCap * userBalance) / rewardPoolShare :
(currentBNBPool * userBalance) / rewardPoolShare
```

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.

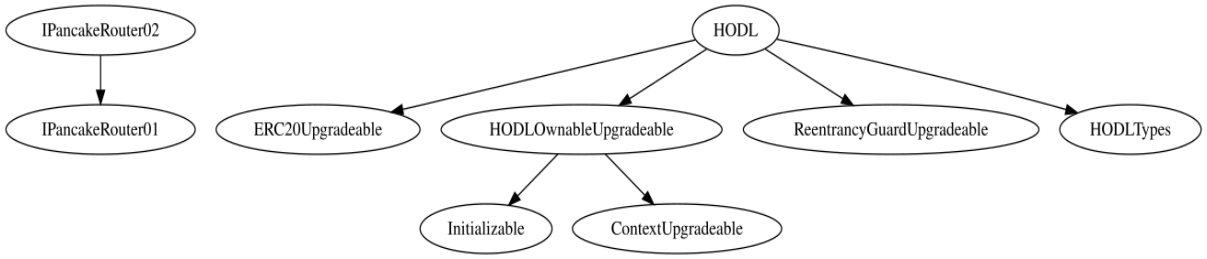
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
HODL	Implementation	ERC20Upgradeable, HODLOwnableUpgradeable, ReentrancyGuardUpgradeable, HODLTypes		
		External	Payable	-
	upgrade	External	✓	onlyOwner reinitializer
	redeemRewards	External	✓	nonReentrant
	updateIsTaxFree	External	✓	onlyOwner onlyPermitted
	updateIsRewardToken	External	✓	onlyOwner onlyPermitted
	excludeFromRewardPoolShare	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	✓	-
	triggerSwapForReward	External	✓	lockTheSwap
	getCurrentBNBReward	External		-
	getRewardPoolShare	Public		-
	getTokensValue	Public		-
	_update	Internal	✓	
	updateRewardPoolShare	Private	✓	
	ensureMaxSellAmount	Private	✓	
	updateClaimDateAfterTransfer	Private	✓	
	swapForReward	Private	✓	lockTheSwap
	getTokensToSell	Private	✓	
	swapTokensForEth	Private	✓	
	calculateUpdateClaim	Private		
HODLTypes	Implementation			
HODLOwnable Upgradeable	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



Summary

HODL contract implements a token and reward distribution 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. 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