Cyberscope

Audit Report

BlaroThings

April 2025

SHA256	06d55a61da5d5142c66c3061928e2f5ad03c70813de7f216f5d99ff2933594e2
Audited by	© cyberscope

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

Audit Updates

Initial Audit	13 Jan 2025
Corrected Phase 2	23 Jan 2025
Corrected Phase 3	08 Apr 2025
Corrected Phase 4	22 Apr 2025

Source Files

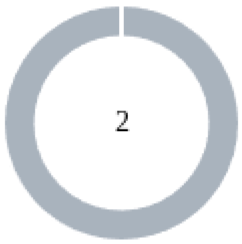
Filename	SHA256
staking.sol	06d55a61da5d5142c66c3061928e2f5ad03c70813de7f216f5d99ff2933594e2

Overview

The BLRStaking contract is a staking platform designed for the BLR token. It enables users to stake their BLR tokens into predefined pools with varying lock-up durations and associated annual percentage yields (APYs). The pools range from 30 days to 1095 days, each offering different rewards and multipliers. Users can opt for either locked or unlocked staking, where locked staking provides higher rewards through multipliers but restricts withdrawals until the lock-up period ends. The contract employs mechanisms to calculate pending rewards based on the staked amount, pool-specific APY, and time elapsed since staking. Additionally, early withdrawals for locked stakes are subject to a penalty.

The contract uses OpenZeppelin's Ownable and ReentrancyGuard contracts for secure and efficient management. The onlyOwner modifier restricts administrative functions such as toggling staking and withdrawing extra tokens to the contract owner. It incorporates non-reentrant modifiers to prevent vulnerabilities like reentrancy attacks. Furthermore, the contract allows users to claim rewards, withdraw their stake, and view their staking history. The withdrawExtraTokens function ensures that surplus tokens beyond the current staked amount can be securely retrieved by the owner, maintaining a clean contract balance.

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	2

	Severity	Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	2	0	0	0

Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

Severity	Code	Description	Status
●	OAA	Out-of-Bounds Array Access	Unresolved
●	PSU	Potential Subtraction Underflow	Unresolved

OAA - Out-of-Bounds Array Access

Criticality	Minor / Informative
Location	staking.sol#L106,124
Status	Unresolved

Description

The contract populates the fixed-length `pooldata` array at sequential indices 1–5, but later attempts to retrieve a pool's parameters using the user-supplied `lockupDuration` value as the array index. Since lockupDuration is a duration in seconds (e.g. 30 days = 2,592,000) rather than 1–5, indexing `pooldata[lockupDuration]` falls outside the initialized bounds and causes a revert.

```
pooldata[1] = PoolInfo(uint32(POOL_1_DURATION), 35, 15); // 3.5% APY, 1.5x multiplier
pooldata[2] = PoolInfo(uint32(POOL_2_DURATION), 80, 20); // 8% APY, 2x multiplier (6 months pool with penalty)
pooldata[3] = PoolInfo(uint32(POOL_3_DURATION), 125, 30); // 12.5% APY, 3x multiplier
pooldata[4] = PoolInfo(uint32(POOL_4_DURATION), 170, 40); // 17% APY, 4x multiplier
pooldata[5] = PoolInfo(uint32(POOL_5_DURATION), 222, 50); // 22.22% APY, 5x multiplier
...
PoolInfo storage pool = pooldata[lockupDuration];
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the user's input value is mapped correctly to the relevant staking plan. Additionally, the team could validate that the provided value equals to one of the predefined lockup durations.

PSU - Potential Subtraction Underflow

Criticality	Minor / Informative
Location	staking.sol#L234
Status	Unresolved

Description

The contract subtracts two values, the second value may be greater than the first value if the contract owner misuses the configuration. As a result, the subtraction may underflow and cause the execution to revert.

Specifically, as part of the reward calculation process, the contract subtracts the `PRECISION` from the `orderInfo.multiplier`. However, for locked orders, the multiplier's maximum value is 50. To calculate the total rewards for locked orders correctly, the contract could use the following expression: `baseReward + ((baseReward * orderInfo.multiplier) / PRECISION)`

```
uint256 totalReward = orderInfo.locked
    ? baseReward + ((baseReward * (orderInfo.multiplier - PRECISION)) /
PRECISION)
    : baseReward;
```

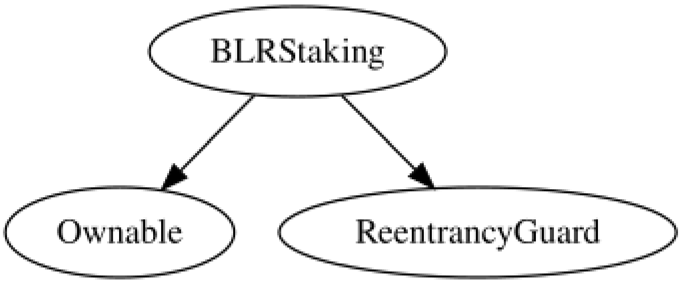
Recommendation

The team is advised to properly handle the code to avoid underflow subtractions and ensure the reliability and safety of the contract. The contract should ensure that the first value is always greater than the second value. It should add a sanity check in the setters of the variable or not allow executing the corresponding section if the condition is violated.

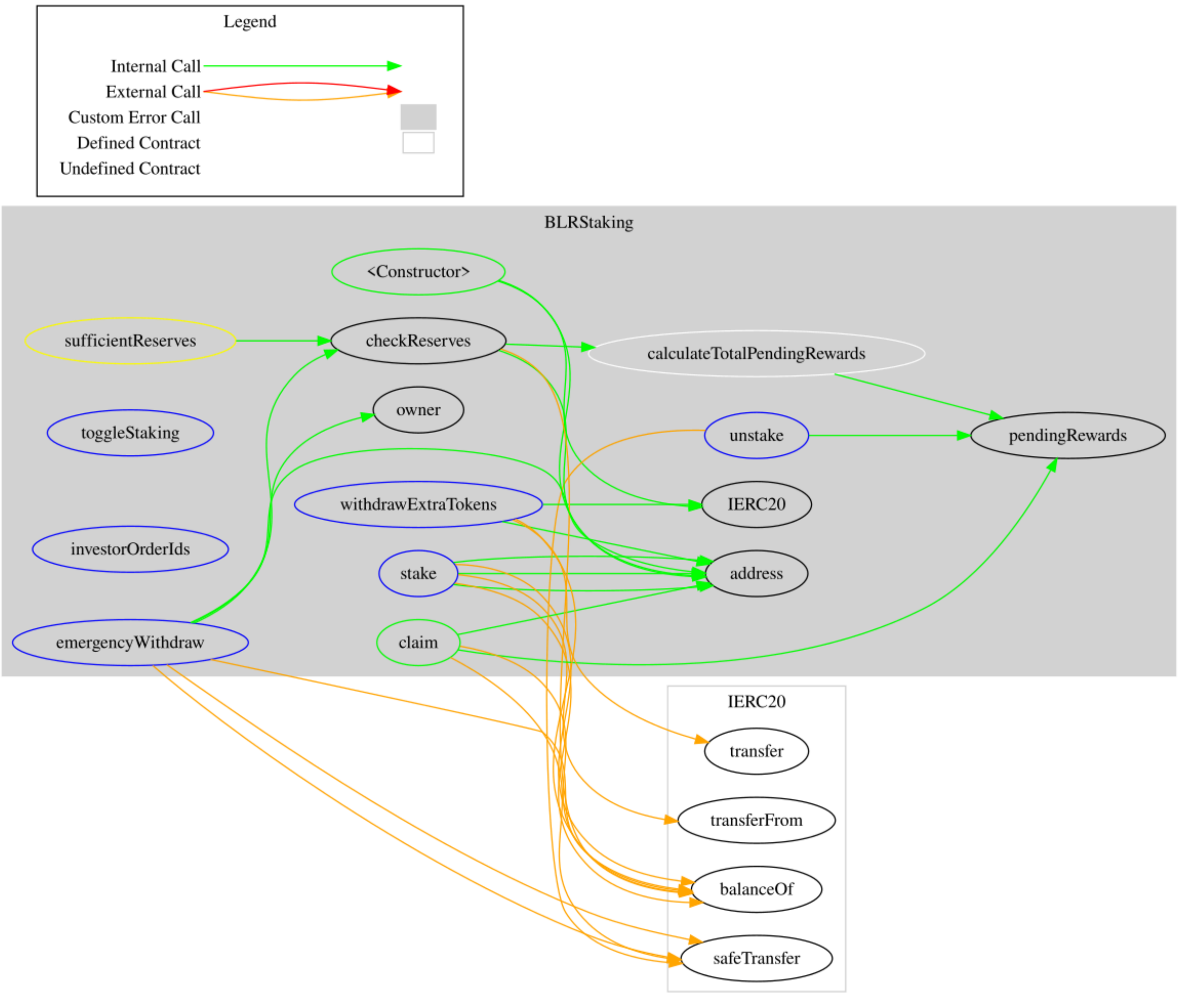
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
BLRStaking	Implementation	Ownable, ReentrancyGuard		
		Public	✓	Ownable
	stake	External	✓	nonReentrant sufficientReserves
	claim	Public	✓	nonReentrant
	unstake	External	✓	nonReentrant
	pendingRewards	Public		-
	toggleStaking	External	✓	onlyOwner
	investorOrderIds	External		-
	withdrawExtraTokens	External	✓	onlyOwner
	calculateTotalPendingRewards	Internal		
	checkReserves	Internal		
	emergencyWithdraw	External	✓	nonReentrant

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
BLRStaking
<Constructor>
sufficientReserves
checkReserves
calculateTotalPendingRewards
owner
toggleStaking
unstake
pendingRewards
withdrawExtraTokens
IERC20
investorOrderIds
stake
address
emergencyWithdraw
claim
IERC20
transfer
transferFrom
balanceOf
safeTransfer

Summary

BlaroThings contract implements a staking and rewards mechanism. This audit investigates security issues, business logic concerns and potential improvements.

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