Cyberscope

Audit Report

Pocketcoin Staking

March 2025

SHA256

ad55d84a72eea0ee40c8d33606b6d8cbeeaf54c9114136799a47d0299de6c134

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

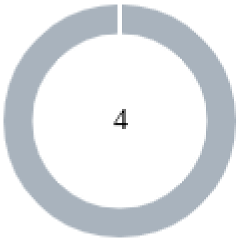
Audit Updates

Initial Audit	27 Feb 2025 https://github.com/cyberscope-io/audits/blob/main/pkoin/v1/audit.pdf
Corrected Phase 2	12 Mar 2025 https://github.com/cyberscope-io/audits/blob/main/pkoin/v2/audit.pdf
Corrected Phase 3	14 Mar 2025
Test Deploy	https://sepolia.etherscan.io/address/0x28612c2542D6682362B440c75958f58Ffe9b025c

Source Files

Filename	SHA256
StakingContract.sol	ad55d84a72eea0ee40c8d33606b6d8cbeeaf54c9114136799a47d0299de6c134

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	0	4	0	0

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CCR	Contract Centralization Risk	Acknowledged
●	CRP	Coupled Rewards Pool	Acknowledged
●	PTAI	Potential Transfer Amount Inconsistency	Acknowledged
●	L19	Stable Compiler Version	Acknowledged

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	StakingContract_TO_AUDIT (2).sol#L324,344,366,383
Status	Acknowledged

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
function withdrawRemainingRewards() external nonReentrant onlyOwner
{...}
function proposeMultiplierUpdate(uint256 lockDuration, uint256
newMultiplier) external onlyOwner {...}
function executeMultiplierUpdate(uint256 lockDuration) external
onlyOwner {...}
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

CRP - Coupled Rewards Pool

Criticality	Minor / Informative
Location	StakingContract_TO_AUDIT (2).sol#L228
Status	Acknowledged

Description

The contract maintains a balance of staked tokens and tokens aimed for rewards, where the staked tokens and reward tokens are the same token. The contract accepts a new stake by comparing its current balance to the staked amounts and the incoming deposit and rewards. The deposited balance of previous stakes is used to determine the eligibility of future stakes. Coupling user deposits with the expected rewards and the contract's balance is prone to errors and could yield inconsistencies.

```
uint256 contractBalance = token.balanceOf(address(this));
require(
contractBalance >= (totalStaked + totalPlannedRewards + amount +
plannedReward),
"Insufficient rewards in contract"
);
```

Recommendation

It is advisable to ensure consistency by maintaining a strict association between the available reward pool and the expected rewards. Decoupling the staked balance from the reward pool will ensure overall consistency.

PTAI - Potential Transfer Amount Inconsistency

Criticality	Minor / Informative
Location	StakingContract_TO_AUDIT (2).sol#L238
Status	Acknowledged

Description

The `transfer()` and `transferFrom()` functions are used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. According to the specification, the transferred amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

The following example depicts the diversion between the expected and actual amount.

Tax	Amount	Expected	Actual
No Tax	100	100	100
10% Tax	100	100	90

```
// Record the stake
stakes[msg.sender].push(
    Stake({
    amount: amount,
    startTime: block.timestamp,
    lockDuration: lockDuration,
    multiplier: multiplier,
    withdrawn: false,
    emergencyWithdrawn: false
    })
);

// Transfer tokens from user to contract
token.safeTransferFrom(msg.sender, address(this), amount);
```

Recommendation

The team is advised to take into consideration the actual amount that has been transferred instead of the expected.

It is important to note that an ERC20 transfer tax is not a standard feature of the ERC20 specification, and it is not universally implemented by all ERC20 contracts. Therefore, the contract could produce the actual amount by calculating the difference between the transfer call.

`Actual Transferred Amount = Balance After Transfer - Balance Before Transfer`

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	StakingContract_TO_AUDIT (2).sol#L4
Status	Acknowledged

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.26;
```

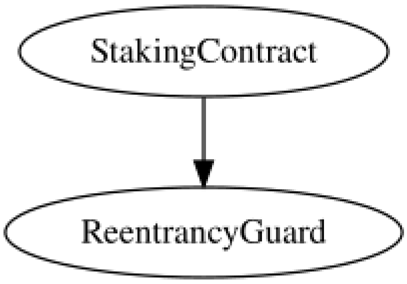
Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

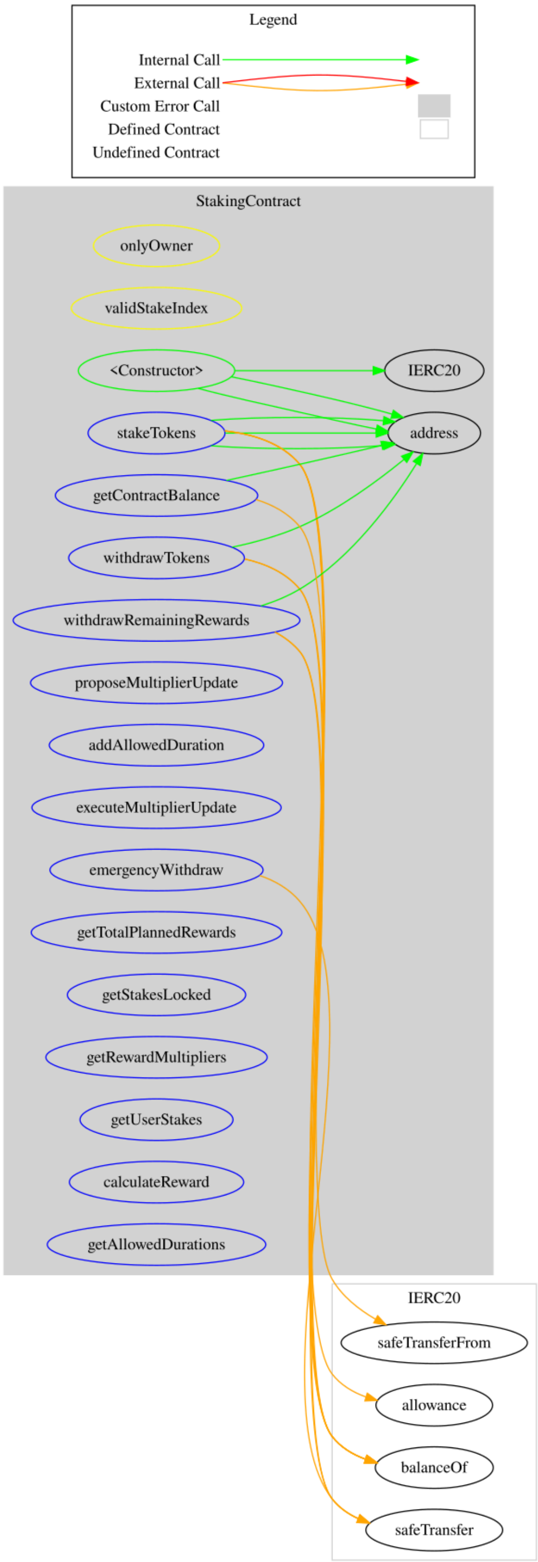
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
StakingContract	Implementation	ReentrancyGuard		
		Public	✓	-
	stakeTokens	External	✓	nonReentrant
	withdrawTokens	External	✓	nonReentrant validStakeIndex
	emergencyWithdraw	External	✓	nonReentrant validStakeIndex
	withdrawRemainingRewards	External	✓	nonReentrant onlyOwner
	proposeMultiplierUpdate	External	✓	onlyOwner
	addAllowedDuration	External	✓	onlyOwner
	executeMultiplierUpdate	External	✓	onlyOwner
	getContractBalance	External		-
	getTotalPlannedRewards	External		-
	getStakesLocked	External		-
	getRewardMultipliers	External		-
	getUserStakes	External		-
	calculateReward	External		validStakeIndex
	getAllowedDurations	External		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
StakingContract
onlyOwner
validStakeIndex
<Constructor>
IERC20
stakeTokens
address
getContractBalance
withdrawTokens
withdrawRemainingRewards
proposeMultiplierUpdate
addAllowedDuration
executeMultiplierUpdate
emergencyWithdraw
getTotalPlannedRewards
getStakesLocked
getRewardMultipliers
getUserStakes
calculateReward
getAllowedDurations
IERC20
safeTransferFrom
allowance
balanceOf
safeTransfer

Summary

Pocketcoin contract implements a staking mechanism. This audit investigates security issues, business logic concerns and potential improvements. The Smart Contract analysis reported no compiler error or critical issues.

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