QuillAudits

Audit Report
May, 2024

For

HUNDRED

Table of Content

Executive Summary 02

Number of Security Issues per Severity 03

Checked Vulnerabilities 04

Techniques and Methods 05

Types of Severity 06

Types of Issues 06

High Severity Issues 07

1. Unrestricted create_state Function Leading to Denial of Service 07

2. Lack of Input Validation in Staking Program Functions 07

Medium Severity Issues 08

1. Use of unwrap in Staking Program Functions 08

2. Unused Variable amount_multipler 09

Informational Issues 09

1. Missing Event Emission in Functions 09

2. Missing NatSpec Documentation 10

3. Unauthorized Fund Withdrawal in withdraw Function 10

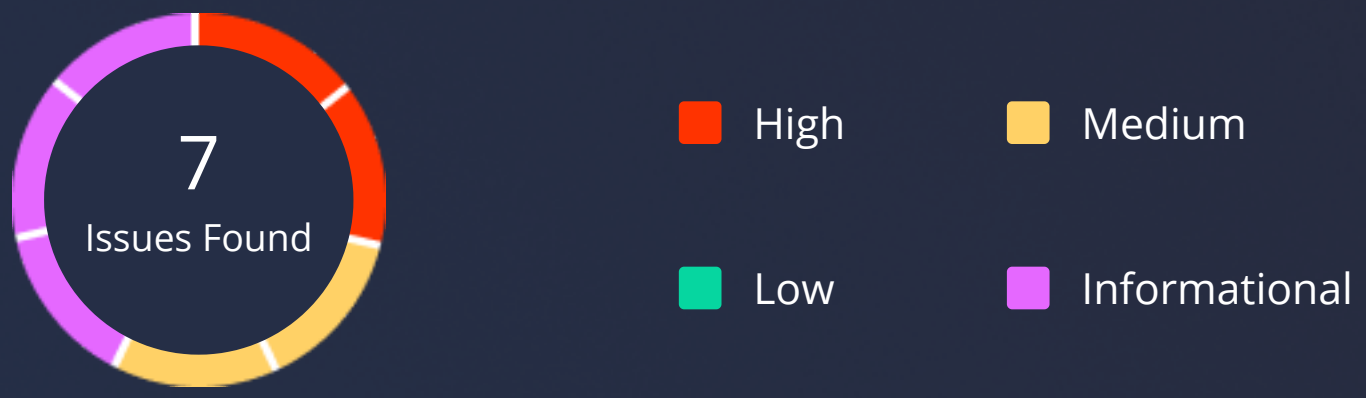
Closing Summary 11

Disclaimer 11

Executive Summary

Project Name	HUND
Project URL	*https://hundonsol.com/* *https://hundpad.com/*
Overview	HUND is a meme coin project launched on the Solana blockchain, drawing inspiration from the universally admired German Shepherd dog breed, renowned for its loyalty, intelligence, and versatility. The project aims to encapsulate these characteristics into a digital asset that fosters a community of enthusiasts and meme lovers, bridging the gap between cryptocurrency and digital art through a vibrant ecosystem centered around German Shepherd memes.
Timeline	9th May 2024 - 14th May 2024
Updated Code Received	21st May 2024
Second Review	22nd May 2024
Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Audit Scope	The scope of this audit was to analyse the HUND Contract for quality, security, and correctness.
Blockchain	Solana
Source Code	*https://drive.google.com/file/d/1j3_-8-lFX2xC0NzyxHRO80xObi3vbEQS/view?usp=share_link*
Contracts In-Scope	Staking.rs
Branch	NA
Fixed In	*https://drive.google.com/file/d/1I7qztlAgjjmgpqaah4J21_yuSZEevYuz/view?usp=share_link*

Number of Security Issues per Severity



	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	2	2	0	3

Checked Vulnerabilities

We have scanned the solana program for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Signer Authorization
- Sysvar Address Checking
- Type Cosplay
- Arbitrary CPI
- Bump Seed Canonicalization
- Incorrect Closing Accounts
- Arithmetic Overflows/Underflows
- Solana Account Confusions
- Insufficient SPL Token Account Verification
- Account Data Matching
- Owner Checks
- Initialization
- Duplicate Mutable Accounts
- PDA Sharing
- Missing Rent Exemption Checks
- Numerical Precision Errors
- Casting Truncation
- Signed Invocation of Unverified Programs

Techniques and Methods

Throughout the audit of HUND Staking Solana program, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of various token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the Solana programs.

Structural Analysis

In this step, we have analysed the design patterns and structure of Solana programs. A thorough check was done to ensure the Solana program is structured in a way that will not result in future problems.

Static Analysis

Static analysis of Solana programs was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of Solana programs.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, and their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of Solana programs in production. Checks were done to know how much gas gets consumed and the possibilities of optimising code to reduce gas consumption.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

High Severity Issues

1. Unrestricted create_state Function Leading to Denial of Service

Path

programs/solana_token_staking/src/lib.rs

Description

The **create_state** function allows any user to create a new state with a specified **id** without any restrictions. Since the **create_pool** function requires the authority of an existing state, and only one pool can be created per **id** within the range of 0 to 12, an attacker can create 12 different states using **create_state** with ids from 0 to 12. This would prevent legitimate users from creating new states or pools, effectively causing a denial of service (DoS) attack on the staking program.

Impact

An attacker can exhaust the available id range by creating states with all possible ids (0 to 12), thereby preventing legitimate users from creating new pools. This results in a denial of service (DoS) for the staking program, rendering it unusable for normal users.

Remediation

Rewrite the function to allow users to create pools with the pool ID within the required range (0 to 12), but allow the state ID to be above 12.

Status

Resolved

2. Lack of Input Validation in Staking Program Functions

Path

programs/solana_token_staking/src/lib.rs

Description

The staking program functions, such as **create_state**, **create_pool**, **update_pool**, **amount**, **stake**, **unstake**, and **withdraw**, lack proper input validation on function parameters. This deficiency can lead to unintended behavior, such as setting total points to 0, which can cause a division by zero panic.

Additionally, unchecked input values can result in the creation of states or pools with invalid parameters, leading to potential exploits and disruption of normal operations.

Remediation

Implement thorough input validation checks for all function parameters. Ensure that values fall within expected ranges and handle edge cases appropriately. Define caps and constraints for input values to prevent unintended behavior and potential exploits.

Status

Resolved

Medium Severity Issues

1. Use of unwrap in Staking Program Functions

Path

programs/solana_token_staking/src/lib.rs

Description

The staking program functions contain multiple instances of the unwrap method, which can lead to panics if an error occurs. The unwrap method forces a program to terminate if the result is an Err, which can be exploited by attackers to cause unintentional crashes. This lack of proper error handling can lead to unintended crashes.

Remediation

Replace all instances of **unwrap** with proper error handling mechanisms. Use **?** to propagate errors or match expressions to handle different error cases gracefully. Ensure that all potential error conditions are accounted for and handled appropriately to prevent panics and maintain program stability.

Status

Resolved

2. Unused Variable amount_multipler

Path

programs/solana_token_staking/src/lib.rs

Description

The variable **amount_multipler** is defined and initialized in the **create_pool** function, but it is not used in any calculations or further logic within the staking program. This unused variable introduces unnecessary complexity.

Remediation

Incorporate the amount_multipler variable into the reward calculation logic to properly account for its intended effect on user rewards. Ensure that this variable is used consistently and correctly within the program to enhance reward calculations for users.

Status

Resolved

Informational Issues

1. Missing Event Emission in Functions

Description

Several functions within the solana_token_staking program lack necessary event emissions. Emitting events is crucial for providing transparency and traceability of important actions and state changes within the smart contract. The missing events include:
1. **create_state** 2. **fund_reward_token** 3. **withdraw**

Remediation

Add appropriate event emissions in the functions to ensure all significant actions and state changes are logged.

Status

Resolved

2. Missing NatSpec Documentation

Description

The program lacks NatSpec (Ethereum Natural Specification Format) documentation for its functions and data structures. NatSpec is a standardized format for documenting programs and is used to provide clear and detailed explanations of the purpose, inputs, outputs, and behavior of the contract's functions and data structures.

Remediation

Add comprehensive NatSpec documentation for all functions, data structures, and critical logic within the program.

Status

Resolved

3. Unauthorized Fund Withdrawal in withdraw Function

Path

programs/solana_token_staking/src/lib.rs

Description

The **withdraw** function allows anyone with the authority to call the function and drain the reward amount and pool amount from the pool and reward vault without proper authorization checks. There is no verification that the reward amount and pool amount are indeed present or have been inputted by the user. Additionally, there are no checks to ensure that the user is authorized to perform the withdrawal.

Impact

An attacker with the authority can exploit this vulnerability to drain all funds from the reward vault and pool vault, leading to a complete loss of funds for the legitimate users of the staking program.

Remediation

Implement proper authorization checks to ensure that the caller is authorized to withdraw funds. Verify that the reward amount and pool amount are indeed present and have been correctly inputted by the user before allowing the transfer.

Status

Resolved

Closing Summary

In this report, we have considered the security of the HUND Staking Contract. We performed our audit according to the procedure described above.

Some issues of High, Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. In the End, HUND team resolved all of the Issues.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in HUND smart contract. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of HUND smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the HUND to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
May, 2024

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com