SLOWMIST

Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.01.21, the SlowMist security team received the Mask Social team's security audit application for Solana Redpacket, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Unsafe External Call Audit
- Design Logic Audit
- Scoping and Declarations Audit
- Account substitution attack Audit
- Malicious Event Log Audit

3 Project Overview

3.1 Project Introduction

This is the red packet module deployed by Mask Social on Solana.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Random number generation uses a predictable or controllable seed	Race Conditions Vulnerability	High	Confirmed
N2	Multiplication overflow risk	Integer Overflow and Underflow Vulnerability	High	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/fengshanshan/solana-redpacket

Initial audit commit: 7d090ae21d400ac4d43cf9fc733469e7918ebb78

Fix patch: https://github.com/DimensionDev/solana-redpacket/pull/9/files

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist security team analyzed the visibility of major contracts during the audit, the result as follows:

redpacket			
Function Name	Account check coverage	Auth Signer	Params Check
create_red_packet_with_spl_token	8/8	signer	8/8
create_red_packet_with_native_token	3/3	signer	8/8
claim_with_spl_token	10/10	payer	-

redpacket			
claim_with_native_token	6/6	payer	-
withdraw_with_spl_token	3/3	creator	-
withdraw_with_native_token	3/3	creator	-

4.3 Vulnerability Summary

[N1] [High] Random number generation uses a predictable or controllable seed

Category: Race Conditions Vulnerability

Content

In the `generate_random_number` function, a predictable or controllable seed is used for random number generation. This method may lead to the predictability of random number generation, allowing attackers to predict or control the generated random numbers, thereby affecting the fairness and security of red-packet receipt.

- programs/solana-redpacket/src/lib.rs

```
fn generate_random_number(redpacket_key: Pubkey, signer_key: Pubkey) -> u64 {
    let clock = Clock::get().unwrap();
    let current_timestamp = clock.unix_timestamp;


    let seed = format!("{}{}{}", redpacket_key, signer_key, current_timestamp);
    let hash_value = hash(seed.as_bytes());
    u64::from_le_bytes(hash_value.to_bytes()[0..8].try_into().unwrap())
}
```

Solution

Use a more secure random number generation method.

Status

Confirmed; Partially fixed, by adding `claimed_amount, current_slot` to increase the difficulty of random number prediction and control.

[N2] [High] Multiplication overflow risk

Category: Integer Overflow and Underflow Vulnerability

Content

In the `calculate_claim_amount` function, there is a risk of multiplication overflow when calculating the random claim amount. Specifically, `remaining_amount * 2` may cause overflow, especially when remaining_amount is close to the maximum value of `u64`. This may lead to incorrect calculation of the red-packet amount, or even cause the program to crash.

- programs/solana-redpacket/src/lib.rs

```
fn calculate_claim_amount(red_packet: &Account<RedPacket>, signer_key: Pubkey) -> u64
{
   //...
       let claim_value = random_value % ((remaining_amount * 2) /
(red_packet.total_number - red_packet.claimed_number) as u64);
   //...
}
```

Solution

By using the `checked_mul` method, you can effectively prevent multiplication overflow and ensure the accuracy and safety of the calculation of the red envelope amount.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002501220001	SlowMist Security Team	2025.01.21 - 2025.01.22	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 2 high risk vulnerabilities.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist