Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.05.09, the SlowMist security team received the BitKeep team's security audit application for Bwb Solana Staking Contract, 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

solana contract for bwb collection and pledges.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Preemptive Initialization	Race Conditions Vulnerability	Suggestion	Fixed
N2	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N3	Failure to verify that the mint account is the same as the distributor's mint	Others	Low	Fixed
N4	There are spillover risks	Integer Overflow and Underflow Vulnerability	Medium	Fixed
N5	Redundant code	Others	Suggestion	Fixed
N6	Potential single point of failure risk	Others	Information	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/bitkeepwallet/bwb-staking-solana

commit:

2af3ac603563cfd9da9b82a4ca32afb6a1253cae

review commit:

66c27285d9eafbf2ac56ec1db4f53bb703f87e43

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:

bwb_stake			
Function Name	Account check coverage	Auth Signer	Params Check
initialize	6/6	-	0/5
create_new_pool	5/5	pool_admin	3/5
update_pool	5/5	pool_admin	4/6
update_receiver	2/2	admin	0/1
update_cosigner	2/2	admin	0/1
update_operator	2/2	admin	0/1
update_pool_admin	2/2	admin	0/1
set_admin_is_paused	2/2	operator	0/1
set_pool_is_paused	3/3	operator	1/2
stake	11/11	cosigner,user	2/2
unstake	11/11	cosigner,user	3/3
claim_reward	11/11	cosigner,user	2/2
withdraw_bwb_token	5/5	operator	0/1
withdraw_other_token	7/7	operator	0/1

4.3 Vulnerability Summary

[N1] [Suggestion] Preemptive Initialization

Category: Race Conditions Vulnerability

Content

The initialize function does not restrict the caller and is subject to a preemptive initialization attack.

- programs/bwb-stake/src/lib.rs

Solution

Deployment contracts and settings can be executed in a deployment transaction.

Status

Fixed

[N2] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

The role of `admin` can be set `operator` role and `receiver`, if the private key of admin role is leaked, it will cause the loss of contract funds.

- programs/bwb-stake/src/lib.rs

```
admin can update_receiver
admin can update_cosigner
admin can update_operator
admin update_pool_admin
operator can set_admin_is_paused
operator can set_pool_is_paused
operator can withdraw_bwb_token
operator can withdraw_other_token
```

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. The authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged; The owner will be a multi-signature address to manage.operator can only withdraw funds to the specified receiver address.

[N3] [Low] Failure to verify that the mint account is the same as the distributor's mint

Category: Others

Content

If the contract has ownership of 2 tokens, then the tokens that can be claimed may be specified, not according to the distributor's settings.

- cosigner-distributor/programs/cosigner-distributor/src/lib.rs

```
pub struct Claim<'info> {
#[derive(Accounts)]
#[instruction(evm_claimer: [u8; 20])]
pub struct Claim<'info> {
   /// The [MerkleDistributor].
   #[account(
       mut,
       seeds = [b"MerkleDistributor".as_ref()],
       bump,
   )]
   pub distributor: Account<'info, MerkleDistributor>,

   ...

   /// The mint to distribute.
   pub mint: Account<'info, Mint>, //SlowMist// have to check if mint is equal to distributor.mint

   ...
}
```

- merkle-distributor/programs/merkle-distributor/src/lib.rs

```
pub struct Claim<'info> {
#[derive(Accounts)]
#[instruction(evm_claimer: [u8; 20])]
pub struct Claim<'info> {
   /// The [MerkleDistributor].
   #[account(
       mut,
       seeds = [b"MerkleDistributor".as_ref()],
       bump,
   )]
   pub distributor: Account<'info, MerkleDistributor>,
```

```
    ...

    /// The mint to distribute.
    pub mint: Account<'info, Mint>, //SlowMist// have to check if mint is equal to
distributor.mint

    ...
}
```

Solution

Qualify the mint to be consistent with distributor.mint

Status

Fixed

[N4] [Medium] There are spillover risks

Category: Integer Overflow and Underflow Vulnerability

Content

`order.reward_amount * passed_days` The multiplication of these two numbers may, in extreme cases, overflow and lead to inaccurate results.

In other implementations of the function it is also straightforward to use the + - method of computation, which carries the risk of overflow, and it is recommended to use the safe method of computation.

- programs/bwb-stake/src/lib.rs

```
pub fn claim_reward(
    ctx: Context<ClaimReward>,
    order_id: u64,
    reward_amount: u64
) -> Result<()> {
    msg!("Instruction: Claim Reward");

    ...

    let reward = order.reward_amount * passed_days / period_days -
order.claimed_reward; //SlowMist// Failure to use safe multiplication, possible risk
of overflow.
    //(passed_days - claimed_days)
    msg!("passed_days is {:?}", passed_days);
    msg!("period_days is {:?}", period_days);
    msg!("reward is {:?}", reward);
```

```
    ...
}
```

Solution

When performing arithmetic operations on numeric variables, use the checked_add, checked_mul, and checked_sub arithmetic functions provided by the Rust standard library. These functions return None on overflow.

Status

Fixed

[N5] [Suggestion] Redundant code

Category: Others

Content

The structure UpdateDistributor is not used.

- merkle-distributor/programs/merkle-distributor/src/lib.rs

```
/// Accounts for [merkle_distributor::update_distributor].
#[derive(Accounts)]
pub struct UpdateDistributor<'info> {
   /// Admin key of the distributor.
   pub admin_auth: Signer<'info>,

   #[account(mut, has_one = admin_auth @ ErrorCode::DistributorAdminMismatch)]
   pub distributor: Account<'info, MerkleDistributor>,
}
```

Solution

Delete the unused structure.

Status

Fixed

[N6] [Information] Potential single point of failure risk

Category: Others

Content

In the `bwb_stake` contract, both take and unstake need to be signed by an official cosigner before the function can

be executed, because the backend is also synchronized to calculate whether the reward is abnormal or not, which is a safety measure that users need to be aware of. This is a safety measure that users need to be aware of, but beware of single points of failure.

Solution

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002405150001	SlowMist Security Team	2024.05.09 - 2024.05.15	Low 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 medium risk, 1 low risk, 1 info, 3 suggestion 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