Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.05.10, the SlowMist security team received the BitKeep team's security audit application for BitKeep Swap Solana, 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.

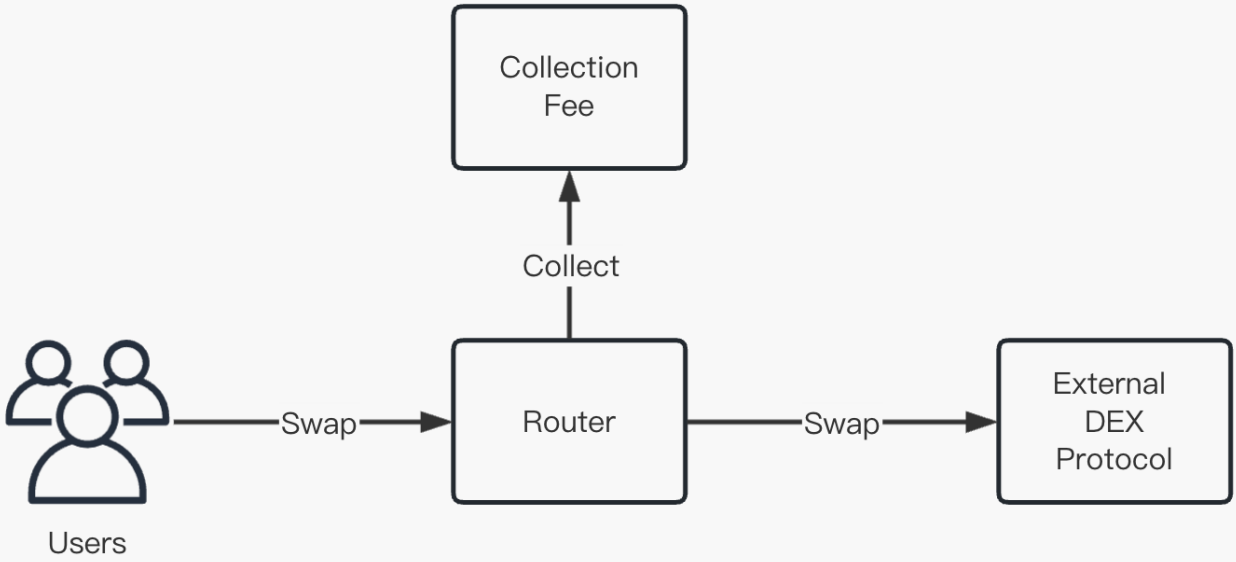
Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- 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

BitKeep Swap Solana is a DEX aggregation protocol deployed on Solana. The agreement consists of two parts: collection fee and router. The collection fee program is used to charge users who use BitKeep Swap, and the maximum fee ratio will not exceed 25%. The router program is used to connect with external DEX to help users perform token swap. The following is a simple architecture diagram:



3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risk of over-privilege	Authority Control Vulnerability Audit	Medium	Fixed
N2	Risk of the external protocol	Unsafe External Call Audit	Suggestion	Fixed
N3	Did not check whether the owner of the account is valid	Design Logic Audit	Low	Acknowledged
N4	Not checked if bkswap_program is valid	Design Logic Audit	Low	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/bitkeepwallet/bitkeep-swap-solana

commit: 15b997d2a83a2616894679eb11540f5dd5d13d52

Fixed Version:

https://github.com/bitkeepwallet/bitkeep-swap-solana

commit: 56273a81eb32475f02ba0713a85440d9694cfcfd

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:

Initialize			
Function Name	Account check coverage	OnlyManage	CPI
initialize	2/3	Y	0

CollectFee			
Function Name	Account check coverage	OnlyManage	CPI
collect_fee	4/5	N	1

SetAuthority			
Function Name	Account check coverage	OnlyManage	CPI
set_authority	2/2	Y	0

SetFeeRate			
Function Name	Account check coverage	OnlyManage	CPI
set_fee_rate	2/2	Y	0

SetFeeReceiver			
Function Name	Account check coverage	OnlyManage	CPI
set_fee_receiver	2/2	Y	0

ProxySwapBaseIn			
Function Name	Account check coverage	OnlyManage	CPI
from	3/22	N	0
proxy_swap_base_in	3/22	N	2

4.3 Vulnerability Summary

[N1] [Medium] Risk of over-privilege

Category: Authority Control Vulnerability Audit

Content

In the agreement, users need to pay a certain amount of fees through collect_fee when performing token swaps. The fee paid depends on the fee_rate parameter of the protocol, and the Bkswap authority can set any value of fee_rate through the set_fee_rate function. If the fee_rate is set wrongly or even exceeds 10000, it will lead to a loss of users' funds.

Code location: programs/bkswap/src/instructions/set_fee_rate.rs

```
pub fn set_fee_rate(ctx: Context<SetFeeRate>, fee_rate: u16) -> Result<()> {
    let bkswap_account = &mut ctx.accounts.bkswap_account;

    bkswap_account.fee_rate = fee_rate;
```

```
    Ok(())
}
```

Solution

It is recommended to limit the setting of fee_rate to a range to avoid this risk.

Status

Fixed; The project team added a range to the fee_rate setting.

[N2] [Suggestion] Risk of the external protocol

Category: Unsafe External Call Audit

Content

The token exchange of the protocol depends on the external Raydium protocol, which will cause unpredictable risks in the Raydium protocol, and the bkswap will also be affected.

Solution

It is recommended to add an emergency pause function to the protocol, or to ensure a fast reaction in the event of an emergency.

Status

Fixed; The project team has added a pause function to this protocol in case of emergencies.

[N3] [Low] Did not check whether the owner of the account is valid

Category: Design Logic Audit

Content

In the raydium_router program, the user will CPI the collect_fee of bkswap_program to pay the fee when performing the proxy_swap_base_in operation, but in ProxySwapBaseIn, it is not checked whether bkswap_account is derived from bkswap_program. Malicious users can pass in fake bkswap_account and bkswap_program to avoid paying fees.

Code location: programs/raydium_router/src/instructions/swap_base_in.rs

```
#[derive(Accounts, Clone)]
pub struct ProxySwapBaseIn<'info> {
    ...
    pub bkswap_account: AccountInfo<'info>,
```

```
    ...
    pub bkswap_program: Program<'info, Bkswap>,
    ...
}
```

Solution

It is recommended to check whether the owner of bkswap_account is bkswap_program.

Status

Acknowledged; After communicating with the project team, the project team stated that those who have the ability to bypass the fee collection can directly perform swap by themselves without going through BitKeep, so the project team accepts the risk of not charging these users.

[N4] [Low] Not checked if bkswap_program is valid

Category: Design Logic Audit

Content

In the raydium_router program, the user will CPI the collect_fee of bkswap_program to pay the fee when performing the proxy_swap_base_in operation, but in ProxySwapBaseIn, it is not checked whether bkswap_program is the real bkswap_program. Malicious users can pass in fake bkswap_program, so that users can perform swap_base_in operation without paying fees.

Code location: programs/raydium_router/src/instructions/swap_base_in.rs

```
pub struct ProxySwapBaseIn<'info> {
    ...
    pub bkswap_program: Program<'info, Bkswap>,
    ...
}


pub fn proxy_swap_base_in(
    ctx: Context<ProxySwapBaseIn>,
    amount_in: u64,
    minimum_amount_out: u64,
) -> Result<()> {
    ...
    let bkswap_program = ctx.accounts.bkswap_program.to_account_info();

    let cpi_ctx = CpiContext::new(bkswap_program, cpi_accounts);
    let amount_in = bkswap::cpi::collect_fee(cpi_ctx, amount_in)?.get();
```

```
    amm_anchor::swap_base_in(ctx.accounts.into(), amount_in, minimum_amount_out)
}
```

Solution

It is recommended to check whether the key of bkswap_program is the real pubkey of bkswap_program in ProxySwapBaseIn.

Status

Acknowledged; After communicating with the project team, the project team stated that those who have the ability to bypass the fee collection can directly perform swap by themselves without going through BitKeep, so the project team accepts the risk of not charging these users.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002305120002	SlowMist Security Team	2023.05.10 - 2023.05.12	Passed

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 2 low risks, and 1 suggestion. All the findings were fixed or acknowledged. The code was not deployed to the mainnet.

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