Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.05.17, the SlowMist security team received the DeltaFi team's security audit application for Deltafi-dex-v2, 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.

Level	Description
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

3 Project Overview

3.1 Project Introduction

AMM 2.0 for efficient automated market making achieves minimized price slippage, sustainable liquidity profitability and optimized capital efficiency.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Best practice of accounts verification	Others	Suggestion	Fixed
N2	Get swap rewards by flashloan	Design Logic Audit	Medium	Fixed
N3	Value overflow risk	Integer Overflow and Underflow Vulnerability	Low	Fixed
N4	Oracle risk	Unsafe External Call Audit	Suggestion	Confirmed
N5	Risk of excessive authority	Authority Control Vulnerability	Low	Confirmed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

Audit version:

https://github.com/delta-fi/deltafi-dex-v2

commit: db4a2c9893b83000328d43aee8ec14dc55dbf4bf

4.2 Visibility Description

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

lib.rs				
Function Name	Lamport check	Account check coverage	OnlyManage	CPI
create_market_config	o	7/10	N	0
create_swap	o	12/14	Y	0
update_swap_config	-	3/3	Y	0
create_liquidity_provider	o	5/6	N	0
init_normal_swap	-	9/12	Y	2
init_stable_swap	-	9/12	Y	2
init_serum_swap	-	12/13	Y	2
deposit_to_normal_swap	-	7/10	N	2
deposit_to_stable_swap	-	7/10	N	2
withdraw_from_normal_swap	-	9/12	N	4
withdraw_from_stable_swap	-	9/12	N	4
normal_swap	-	7/12	N	3

lib.rs				
normal_swap_with_referrer	-	7/13	N	3
stable_swap	-	6/12	N	3
stable_swap_with_referrer	-	6/13	N	3
create_farm	o	6/7	Y	0
update_farm_config	-	3/3	Y	0
deposit_to_farm	-	4/5	N	0
withdraw_from_farm	-	4/5	N	0
claim_farm_rewards	-	7/8	N	1
create_deltafi_user	o	4/5	N	0
create_deltafi_user_with_referrer	o	4/6	N	0
claim_swap_rewards	-	5/6	N	1

4.3 Vulnerability Summary

[N1] [Suggestion] Best practice of accounts verification

Category: Others

Content

contracts/programs/deltafi-dex-v2/src/lib.rs

```
fn validate_liquidity_provider_info<'info>(
    liquidity_provider_info: &impl LiquidityProviderInfo<'info>,
) -> Result<()> {
    require!(
```

```
        liquidity_provider_info.user_token_base().mint ==
liquidity_provider_info.token_base().mint,
        error::ErrorCode::IncorrectMint
    );
    require!(
        liquidity_provider_info.user_token_quote().mint
            == liquidity_provider_info.token_quote().mint,
        error::ErrorCode::IncorrectMint
    );
    Ok(())
}
//...
fn validate_reward_token_accounts<'info>(
    market_config: &Account<'info, MarketConfig>,
    deltafi_token: &Account<'info, TokenAccount>,
    user_deltafi_token: &Account<'info, TokenAccount>,
) -> Result<()> {
    require!(
        market_config.deltafi_token == deltafi_token.key(),
        error::ErrorCode::InvalidAccount
    );
    require!(
        market_config.deltafi_mint == user_deltafi_token.mint,
        error::ErrorCode::IncorrectMint
    );
    Ok(())
}
//...
fn validate_admin_fee_token(
    swap_info: &SwapInfo,
    swap_direction: SwapDirection,
    admin_destination_token: &Account<TokenAccount>,
) -> Result<()> {
    let admin_fee_token_key = match swap_direction {
        SwapDirection::SellQuote => swap_info.admin_fee_token_base,
        SwapDirection::SellBase => swap_info.admin_fee_token_quote,
    };
    require!(
        admin_fee_token_key == admin_destination_token.key(),
        error::ErrorCode::InvalidAdmin
    );
    Ok(())
}
//...
```

```
pub fn create_swap(
//...
    let mint_base_key = ctx.accounts.mint_base.key();
    let mint_quote_key = ctx.accounts.mint_quote.key();
    require!(
        mint_base_key != mint_quote_key,
        error::ErrorCode::RepeatedMint
    );
    require!(
        ctx.accounts.admin_fee_token_base.mint == mint_base_key,
        error::ErrorCode::InvalidOwner
    );
    require!(
        ctx.accounts.admin_fee_token_quote.mint == mint_quote_key,
        error::ErrorCode::InvalidOwner
    );
//...
```

Solution

Recommend to wirte in account macros `#[account()]`.

Status

Fixed; Fixed in https://github.com/delta-fi/deltafi-dex-v2/pull/156

[N2] [Medium] Get swap rewards by flashloan

Category: Design Logic Audit

Content

Usesr can get `DELFI` token while they swap tokens, attacker can use flashloan to get lots of rewards in short time,

or create an useless token pair to trade for rewards.

Function:

fn normal_swap

fn normal_swap_with_referrer

fn stable_swap

fn stable_swap_with_referrer

contracts/programs/deltafi-dex-v2/src/lib.rs

```
let swap_result = do_swap(
    ctx.accounts,
    swap_direction,
    amount_in,
    min_amount_out,
    &mut pool,
)?;
ctx.accounts
    .deltafi_user
    .add_swap_rewards(swap_result.swap_reward_amount)?
```

Solution

Add swap reward limit.

Status

Fixed; Fixed in https://github.com/delta-fi/deltafi-dex-v2/pull/151

[N3] [Low] Value overflow risk

Category: Integer Overflow and Underflow Vulnerability

Content

contracts/programs/deltafi-dex-v2/src/state/swap.rs

```
pub fn claim_rewards(&mut self, farm_config: &FarmConfig, ts: UnixTimestamp) ->
Result<u64> {
    let rewards = self.base_position.claim_rewards(farm_config, ts)?
        + self.quote_position.claim_rewards(farm_config, ts)?;
    if rewards == 0 {
        Err(ErrorCode::InsufficientClaimAmount.into())
    } else {
        Ok(rewards)
    }
}

pub fn get_total_share(&self) -> (u64, u64) {
    (
```

```
            self.base_share + self.base_position.deposited_amount,
            self.quote_share + self.quote_position.deposited_amount,
        )
    }
```

deltafi-dex-v2-main/contracts/programs/deltafi-dex-v2/src/lib.rs

```
pool.collect_trade_fee(
    withdraw_fee_base - admin_fee_base,
    withdraw_fee_quote - admin_fee_quote,
)?;
//...
pub fn claim_swap_rewards(ctx: Context<ClaimSwapRewards>) -> Result<()> {
    let reward_amount = ctx.accounts.deltafi_user.claim_swap_rewards()?
        + ctx.accounts.deltafi_user.claim_referral_rewards()?;
    transfer_reward(ctx.accounts, reward_amount)?;
    Ok(())
}
```

Solution

Use `checked_add/checked_sub/checked_div/checked_mul` , instead of `+-*/`

Status

Fixed; Fixed in: https://github.com/delta-fi/deltafi-dex-v2/pull/155

[N4] [Suggestion] Oracle risk

Category: Unsafe External Call Audit

Content

The oracle price is provided by the Pyth Network, if the Pyth Network fails and provides the wrong price, the market will be destroyed and cannot be recovered.

Solution

Use multiple price discovery mechanisms, including multiple oracles, and establish risk control mechanisms

Status

Confirmed

[N5] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

contracts/programs/deltafi-dex-v2/src/instructions/lib.rs

```
pub fn create_market_config
pub fn update_swap_config
pub fn update_farm_config
```

`admin` can modify the configuration, it will affect asset security.

Solution

It is recommended to set `admin` address to timelock contract, governance contract, or multi-sign contract to reduce the risk of private key loss.

Status

Confirmed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002206020001	SlowMist Security Team	2022.05.17 - 2022.06.02	Passed

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 1 medium risk, 2 low risk, 2 suggestion vulnerabilities. And 1 low risk, 1 suggestion vulnerabilities were confirmed and being fixed; All other findings were fixed. 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