OFFSIDE LABS

Stake for Fee

Smart Contract Security Assessment

October 2024

Prepared for:

Meteora

Prepared by:

Offside Labs

Sirius Xie

Ronny Xing

Contents

1 About Offside Labs

Offside Labs is a leading security research team, composed of top talented hackers from both academia and industry.

We possess a wide range of expertise in modern software systems, including, but not limited to, *browsers*, *operating systems*, *IoT devices*, and *hypervisors*. We are also at the forefront of innovative areas like *cryptocurrencies* and *blockchain technologies*. Among our notable accomplishments are remote jailbreaks of devices such as the **iPhone** and **PlayStation 4**, and addressing critical vulnerabilities in the **Tron Network**.

Our team actively engages with and contributes to the security community. Having won and also co-organized *DEFCON CTF*, the most famous CTF competition in the Web2 era, we also triumphed in the **Paradigm CTF 2023** within the Web3 space. In addition, our efforts in responsibly disclosing numerous vulnerabilities to leading tech companies, such as *Apple*, *Google*, and *Microsoft*, have protected digital assets valued at over **$300 million**.

In the transition towards Web3, Offside Labs has achieved remarkable success. We have earned over **$9 million** in bug bounties, and **three** of our innovative techniques were recognized among the **top 10 blockchain hacking techniques of 2022** by the Web3 security community.

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs

2 Executive Summary

Introduction

Offside Labs completed a security audit of *Stake for Fee* smart contracts, starting on October 17, 2024, and concluding on October 23, 2024.

Project Overview

Stake for Fee is a staking program that distributes *Dynamic-AMM* fees to top-ranked stakers. Tokens creators can create pools on *Dynamic-AMM*, then lock LP tokens for the owner as a PDA vault of *Stake for Fee* program. Stakers can stake tokens into *Stake for Fee* program, where top stakers earn fees generated from the liquidity pool.

Audit Scope

The assessment scope contains mainly the smart contracts of the stake_for_fee program for the *Stake for Fee* project.

The audit is based on the following specific branches and commit hashes of the codebase repositories:

- *Stake for Fee*
  - Branch: main
  - Commit Hash:
    - eef6d5ba057fa5f8f0c06c90781a1b1d3d71563b
    - e4e6398d5e85309c77a280993c9d197356e91ec6
  - Codebase Link

We listed the files we have audited below:

- *Stake for Fee*
  - programs/stake-for-fee/src/**/*.rs

Findings

The security audit revealed:

- 0 critical issue
- 0 high issue
- 1 medium issues
- 0 low issue
- 2 informational issues

Further details, including the nature of these issues and recommendations for their remediation, are detailed in the subsequent sections of this report.

3 Summary of Findings

ID	Title	Severity	Status
01	Already Accumulated unclaimed_fee_pending is Not Considered	Medium	Fixed
02	Optimized Conditions to Prevent Precision Loss in Claim Fee	Informational	Fixed
03	Missing Fee Distribution Before Updating seconds_to_full_unlock	Informational	Fixed

4 Key Findings and Recommendations

4.1 Already Accumulated unclaimed_fee_pending is Not Considered

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Logic Error

Description

The `LockEscrowExt.get_pending_claim_fee` function only tracks the new fees that have not yet been accounted for, without considering the fees that have already been accumulated in `LockEscrow.unclaimed_fee_pending`.

In the Dynamic AMM, both the `lock` and `claim_fee` instructions can trigger fee accumulation into `LockEscrow.unclaimed_fee_pending`.

Impact

Due to the Dynamic AMM lock instruction is permissionless, an attacker can continuously call the lock instruction to lock 1 lamport token, causing the new fee obtained by `get_pending_claim_fee` to always be 0.

```
    /// Lock account
    #[account(
        mut,
        has_one=pool,
        // has_one=owner, // anyone can increase lock amount for an
         ↪  escrow
        has_one=escrow_vault,
    )]
    pub lock_escrow: Box<Account<'info, LockEscrow>>,

    /// Can be anyone
    #[account(mut)]
    pub owner: Signer<'info>,
```

programs/amm/src/instructions/lock.rs#L30-L41

This will cause the `should_lock_escrow_claim_fee` method to return false, thereby continuously preventing the vault from claiming and distributing fees.

Recommendation

Consider both `new_fee` and `LockEscrow.unclaimed_fee_pending` in the function `get_pending_claim_fee`.

Mitigation Review Log

Fixed in the commit **231e7f48fc7ce747cfdded73b75e1ee763e6f072**.

4.2 Informational and Undetermined Issues

Optimized Conditions to Prevent Precision Loss in Claim Fee

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Logic Error

To prevent changes in virtual price due to precision loss, `should_lock_escrow_claim_fee` checks whether both `pending_fee_a` and `pending_fee_b` are zero.

```
// Don't claim when there's no pending fee. To prevent virtual price
↪   of lock escrow increased, even there's 0 token to claim due to
↪   precision loss.
if pending_fee_a == 0 && pending_fee_b == 0 {
    return Ok(false);
}
```

programs/stake-for-fee/src/utils.rs#L180-L183

In a balanced pool, the values of token a and token b resulting from burning LP tokens are nearly equal. Therefore, we can use `||` instead of `&&` as the branch condition. This is because if either token is zero, it indicates that burning fee LP tokens at this time would face significant precision loss.

Missing Fee Distribution Before Updating seconds_to_full_unlock

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Logic Error

The parameter `seconds_to_full_unlock` is utilized in the `distribute_fees_to_stakers` to control the speed of fee release. If this parameter is updated without first claiming and distributing the fees, it may negatively impact the release of fees accrued since the last `distribute_fees_to_stakers` execution. Therefore, it is recommended to invoke `claim_lock_escrow_fees_and_distribute` at the start of the `update_seconds_to_full_unlock` IX. Alternatively, for a way without modifying the contract, since the `update_seconds_to_full_unlock` instruction is an admin IX, we could also place the `claim_fee_crank` instruction before it when sending the transaction from the admin client.

5 Disclaimer

This audit report is provided for informational purposes only and is not intended to be used as investment advice. While we strive to thoroughly review and analyze the smart contracts in question, we must clarify that our services do not encompass an exhaustive security examination. Our audit aims to identify potential security vulnerabilities to the best of our ability, but it does not serve as a guarantee that the smart contracts are completely free from security risks.

We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster the security of their smart contracts.

It is important to note that the scope of our audit is limited to the areas outlined within our engagement and does not include every possible risk or vulnerability. Continuous security practices, including regular audits and monitoring, are essential for maintaining the security of smart contracts over time.

Please note: we are not liable for any security issues stemming from developer errors or misconfigurations at the time of contract deployment; we do not assume responsibility for any centralized governance risks within the project; we are not accountable for any impact on the project's security or availability due to significant damage to the underlying blockchain infrastructure.

By using this report, the client acknowledges the inherent limitations of the audit process and agrees that our firm shall not be held liable for any incidents that may occur subsequent to our engagement.

This report is considered null and void if the report (or any portion thereof) is altered in any manner.

OFFSIDE LABS

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs