OFFSIDE LABS

Stake Deposit Interceptor

Smart Contract Security Assessment

November 2024

Prepared for:

EXO Tech LLC
Jito Foundation

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 Deposit Interceptor* smart contracts, starting on Nov 22, 2024, and concluding on Nov 29, 2024.

Project Overview

Jito Stake Deposit Authority implements a decay mechanism for LST by introducing a time delay with a linear fee curve. This ensures that the LST received by DepositStake can only be fully accessed after the admin-defined time period; otherwise, a portion of it will be charged as a fee. This mechanism mitigates the risks introduced by socialized toxic flow.

Audit Scope

The assessment scope contains mainly the smart contracts of the stake-deposit-interceptor program for the *Stake Deposit Interceptor* project.

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

- Stake Deposit Interceptor
    - Codebase: https://github.com/exo-tech-xyz/stake-deposit-interceptor
    - Commit Hash: c68f476cc483658e12741f84fc70f9340ccc6575

We listed the files we have audited below:

- Stake Deposit Interceptor
    - program/src/**/*.rs

Findings

The security audit revealed:

- 1 critical issues
- 0 high issue
- 0 medium issue
- 2 low issues
- 3 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	Vault Tokens Vulnerable to Drain	Critical	Fixed
02	Precision Loss when Calculating Fee in claim_pool_token	Low	Fixed
03	Inconsistencies Between Spec and Code in Signer Checks	Low	Fixed
04	Missing initial_fee_bps Range Check in update_deposit_stake_authority	Informational	Fixed
05	Missing Account Validation in deposit_stake IXs	Informational	Fixed
06	Missing Program ID Checks in init_stake_pool_deposit_stake_authority	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Vault Tokens Vulnerable to Drain

Severity: Critical	**Status: Fixed**
Target: Smart Contract	**Category: Data Validation**

Description

In `deposit_stake` and `deposit_stake_with_slippage` , there is no check to verify whether the provided `stake_pool_program_info` matches either `deposit_stake_authority.stake_pool_program_id` or the SPL `stake_pool_program` .

```
    let stake_pool_program_info = next_account_info(account_info_iter)?;
    deposit_stake_cpi(
        stake_pool_program_info,
        ...
    )?;
```

program/src/processor.rs#L241-L317

If a user supplies a malicious `stake_pool_program_info` , it can be directly invoked within `deposit_stake_cpi` as the program.

```
fn deposit_stake_cpi<'a>(
    program_info: &AccountInfo<'a>,
    ...
) -> Result<(), ProgramError> {
    let ix = Instruction {
        program_id: *program_info.key,
        accounts,
        data,
    };
    let ret = invoke_signed(
        &ix,
        &account_infos,
        &[deposit_stake_authority_signer_seeds!(
            deposit_stake_authority
        )],
    );
```

program/src/processor.rs#L689-L769

Since this CPI uses `deposit_stake_authority` as a signer, accounts associated with `deposit_stake_authority` could be compromised by the malicious program.

Impact

An attacker could prepare a malicious `stake_pool_program` and use `deposit_stake` or `deposit_stake_with_slippage` to directly CPI into this malicious program. Because the CPI uses `deposit_stake_authority` as a signer, the attacker could then invoke `spl_token::set_authority` within the malicious program to change the authority of the `deposit_stake_authority.vault` Token Account to an attacker-controlled account. This would allow the attacker to steal all JitoSol from `deposit_stake_authority.vault` .

Proof of Concept

The steps for an attacker to steal all JitoSol from `deposit_stake_authority.vault` are as follows:

Step 1: Deploy a malicious `stake_pool_program` in advance. This malicious program can invoke `set_authority` , assigning the authority of a Token Account to any intended account.

Step 2: The attacker invokes the `deposit_stake` IX and fills some of the accounts in the `accounts` array as follows.

- accounts.stake_pool_program_info = `Malicious stake_pool_program`
- accounts.stake_pool_info = `deposit_stake_authority.vault`
- accounts.validator_stake_list_info = `Attacker's Account` After the `deposit_stake` IX is executed, the Token Account Authority of `deposit_stake_authority.vault` is changed to the attacker's account.

Step 3: The attacker can then use `spl_token::transfer` to directly transfer all JitoSol from `deposit_stake_authority.vault` .

Recommendation

It is recommended to add a check for `stake_pool_program_info` , ensuring it matches either `deposit_stake_authority.stake_pool_program_id` or the `SPL stake_pool_program` .

If `stake_pool_program_info` is required to match `deposit_stake_authority.stake_pool_program_id` , it is also recommended to add constraints on `stake_pool_program_info` in `process_init_stake_pool_deposit_stake_authority` .

Mitigation Review Log

Fixed in the commit **cf6b2373a048acb3e35e7678b3623eed6725f83f.**

4.2 Precision Loss when Calculating Fee in claim_pool_token

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Precision

Description

The calculation of the fee in `claim_pool_token` uses the following code:

```
        let fee_amount = u128::from(u32::from(self.initial_fee_bps))
            .checked_mul(cool_down_time_left as u128)
            .expect("overflow")
            .checked_mul(total_amount as u128)
            .expect("overflow")
            .checked_div(cool_down_seconds as u128)
            .expect("overflow")
            .div_ceil(Self::FEE_BPS_DENOMINATOR.into());
```

program/src/state.rs#L102-L109

Two consecutive integer divisions are performed. Although the second division applies ceiling rounding, there can still be precision loss for certain data.

Impact

If precision loss occurs, the calculated fee amount will be smaller, resulting in the fee token account receiving less fee.

Proof of Concept

Here is a group of example data:

```
cool_down_seconds: 488
cool_down_time_left: 473
initial_fee_bps: 8610
total_amount: 257252395374340
```

$$
\begin{aligned}
fee_{two\_div} &= \left\lceil \frac{\left\lfloor \frac{initial\_fee\_bps * cooldown\_time\_left * total\_amount}{cooldown\_seconds} \right\rfloor}{10000} \right\rceil \\
&= 214686085601201
\end{aligned}
$$

Disregarding the rounding issue, combining two consecutive integer divisions into one yields the following formula and result.

$$\begin{aligned} fee_{one\_div} &= \left\lceil \frac{initial\_fee\_bps * cooldown\_time\_left * total\_amount}{cooldown\_seconds * 10000} \right\rceil \\ &= 214686085601202 \end{aligned}$$

Recommendation

Adjusting the formula to combine the two integer divisions into one can improve the precision loss caused by the divisions.

Mitigation Review Log

Fixed in the commit **cf6b2373a048acb3e35e7678b3623eed6725f83f.**

4.3 Inconsistencies Between Spec and Code in Signer Checks

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Data Validation

Description

In the spec defined by `StakeDepositInterceptorInstruction` , the accounts passed to each IX are annotated with `w (writable)` and `s (signer)` .

However, when processing the IX, signer checks are not performed on the accounts marked as `s` in the spec, or signer checks are not required in the actual code but are still marked as `s` .

The inconsistencies between the spec and the code implementation are listed.

- InitStakePoolDepositStakeAuthority
  - authority_info: missing signer check.
- UpdateStakePoolDepositStakeAuthority
  - new_authority_info: missing signer check.
- DepositStake & DepositStakeWithSlippage
  - deposit_stake_authority_info: redundant signer flag.
  - base_info: missing signer check.

Imapct

This may cause program instructions to deviate from intended permission design.

Recommendation

Add corresponding accounts checks according to the instructions designs/comments.

Mitigation Review Log

Fixed in the commit **cf6b2373a048acb3e35e7678b3623eed6725f83f**.

4.4 Informational and Undetermined Issues

Missing initial_fee_bps Range Check in update_deposit_stake_authority

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Data Validation

In the `update_deposit_stake_authority` IX, the value of `deposit_stake_authority.initial_fee_bps` can be modified. However, unlike the `init_stake_pool_deposit_stake_authority` IX, there is no range check for the `initial_fee_bps` parameter in `update_deposit_stake` `_authority` IX. As a result, it is possible to mistakenly set `initial_fee_bps` to a value greater than 10000 during `update_deposit_stake_authority` , which could lead to transferring more fees than expected during a claim.

Missing Account Validation in deposit_stake IXs

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Data Validation

In `deposit_stake` and `deposit_stake_with_slippage` , after returning from the `stake_pool` CPI, a PDA account `DepositReceipt` is generated. The seed for this PDA account includes `stake_pool` and `base` . However, in `process_deposit_stake` , there is no validation to ensure that the provided `stake_pool` matches the `deposit_stake_authority.stake_pool` . This could allow a user to pass in a `stake_pool` that does not belong to the current `deposit_stake_authority` , thereby constructing an abnormal `DepositReceipt` .

Missing Program ID Checks in init_stake_pool_deposit_stake_authority

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Data Validation

In `init_stake_pool_deposit_stake_authority` , there is no validation of the ID for `stake_pool_program_info` . A malicious user can pass in any arbitrary `stake_pool_program` along with the corresponding `stake_pool` . They can also complete the initialization for a new `deposit_stake_authority` and subsequently use this `deposit_stake_authority` account in the `deposit_stake*` IXs to interact with the `SPL stake_pool_program` .

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