OFFSIDE LABS

Jito (Re)staking Vault

Smart Contract Security Assessment

November 2024

Prepared for:

Jito Foundation

Prepared by:

Offside Labs

Sirius Xie

Yao Li

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 *Jito (Re)staking Vault* smart contracts, starting on Oct 28, 2024, and concluding on Nov 6, 2024.

Project Overview

Jito (Re)staking is a hybrid staking and restaking protocol that enhances capital efficiency for stakers, offering liquidity on staked assets while earning boosted rewards from protocols.

The vault program is a key part of Jito (Re)staking, responsible for storing deposits and managing the minting and burning of tokenized stakes. It supports various configurations and roles, including admins, operators, and NCNs, and offers features like token management and fee handling. These capabilities provide a comprehensive liquid staking solution.

Audit Scope

The assessment scope contains mainly the smart contracts of the vault program for the *Jito (Re)staking* project.

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

- restaking vault
  - Branch: master
  - Commit Hash: 60b388421be855466c7bed191edbea8f449c8a88
  - Codebase Link

We listed the files we have audited below:

- restaking vault
  - vault_program/src/*.rs
  - vault_core/src/*.rs

Findings

The security audit revealed:

- 0 critical issue
- 0 high issue
- 1 medium issues
- 1 low issues
- 4 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	Incorrect PDA Validation in CooldownVaultNcnTicket Instruction	Medium	Fixed
02	Vault with Token-2022 Mint is Unusable	Low	Acknowledged
03	Incorrect is_writable Validation in WarmupVaultNcnSlasherTicket Instruction	Informational	Acknowledged
04	Comment Mismatch in check_reward_fee_effective_rate	Informational	Acknowledged
05	Inconsistent Use of MAX_FEE_BPS and MAX_BPS	Informational	Acknowledged
06	Unclear Rent Receiver in close_vault_update_state_tracker	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Incorrect PDA Validation in CooldownVaultNcnTicket Instruction

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

Description

In the `CooldownVaultNcnTicket` instruction, the validation of the `vault_ncn_ticket` is performed using the following code:

```
33 VaultNcnTicket::load(program_id, vault_ncn_ticket, ncn, vault_info,
   ↪ true)?;
```

vault_program/src/cooldown_vault_ncn_ticket.rs#L33-L33

However, the `VaultNcnTicket::load` function is defined as follows:

```
95  pub fn load(
96      program_id: &Pubkey,
97      vault_ncn_ticket: &AccountInfo,
98      vault: &AccountInfo,
99      ncn: &AccountInfo,
100     expect_writable: bool,
101 ) -> Result<(), ProgramError> {
```

vault_core/src/vault_ncn_ticket.rs#L95-L101

In this implementation, the `vault` and `ncn` parameters are incorrectly positioned, leading to failure in the PDA validation.

Impact

Due to this misplacement, the PDA validation in the `CooldownVaultNcnTicket` instruction will always fail, which can hinder functionality and affect the contract's operations.

Recommendation

Correct the parameter order in the `VaultNcnTicket::load` call.

Mitigation Review Log

Fixed in the commit **aa109cb4599705a21f2c12fe7e988c25ac8cb75e.**

4.2 Vault with Token-2022 Mint is Unusable

Severity: Low	Status: Acknowledged
Target: Smart Contract	Category: Logic Error

Description

In `initialize_vault`, as long as `mint.owner` is either `spl_token::id()` or `spl_token_2022::id()`, the mint can be set as `vault.supported_mint`.

```
208 pub fn load_token_mint(info: &AccountInfo) -> Result<(), ProgramError> {
209     if !(info.owner.eq(&spl_token::id()) ||
        ↪ info.owner.eq(&spl_token_2022::id())) {
210         msg!("Account is not owned by the token program");
211         return Err(ProgramError::InvalidAccountOwner);
212     }
```

core/src/loader.rs#L208-L212

However, in other IXs, checks on the ATA owner linked to `vault.supported_mint` only permit `spl_token::id()`.

```
116 pub fn load_associated_token_account(
117     token_account: &AccountInfo,
118     owner: &Pubkey,
119     mint: &Pubkey,
120 ) -> Result<(), ProgramError> {
121     if token_account.owner.ne(&spl_token::id()) {
122         msg!("Account is not owned by the token program");
123         return Err(ProgramError::InvalidAccountOwner);
124     }
```

core/src/loader.rs#L116-L124

Therefore, the vault currently does not support operations such as transfers on a Token-2022 Mint.

Impact

If a vault is set up with a Token-2022 Mint, the above check prevents users from interacting with it through certain IXs, like `mint_to`, effectively rendering the vault unusable.

Recommendation

Adjust various token-related utility functions in `core`, such as `load_associated_token_account`, to add support for Token-2022.

Alternatively, disable support for Token-2022 Mint & Token Accounts until full support can be implemented.

4.3 Informational and Undetermined Issues

Incorrect is_writable Validation in WarmupVaultNcnSlasherTicket Instruction

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

In the `WarmupVaultNcnSlasherTicket` instruction, the `vault_ncn_slasher_ticket` account is not validated as writable, despite being modified later in the code. The current validation uses `is_writable = false`, as shown below:

```
30 VaultNcnSlasherTicket::load(
31     program_id,
32     vault_ncn_slasher_ticket,
33     vault_info,
34     ncn,
35     slasher,
36     false,
37 )?;
```

vault_program/src/warmup_vault_ncn_slasher_ticket.rs#L30-L37

Comment Mismatch in check_reward_fee_effective_rate

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

In the `check_reward_fee_effective_rate` function, the `st_vrt_ratio` is calculated using the following code:

```
850 let precision_factor = MAX_FEE_BPS as u128;
851
852 // Calculate st_vrt_ratio with higher precision (multiply by 1e6 for 6
    ↪ decimal places)
853 let st_vrt_ratio = new_st_balance_u128
854     .checked_mul(precision_factor)
855     .and_then(|v: u128| v.checked_div(vrt_supply_u128))
856     .ok_or(VaultError::VaultOverflow)?;
```

vault_core/src/vault.rs#L850-L856

The comment indicates the precision is 6 decimal places but `MAX_FEE_BPS` is 10000.

Inconsistent Use of MAX_FEE_BPS and MAX_BPS

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

In the vault, there are four `*_fee_bps` parameters that record the fee ratios under different scenarios. Before setting these fields, the program checks whether the new parameter values are within the specified range.

However, there is an issue in the program where the new values of `*_fee_bps` are compared against different constants. For example, in the case of `reward_fee_bps`, during the `initialize_vault`, it is compared against both `MAX_FEE_BPS` and `MAX_BPS`.

```
56      || reward_fee_bps > MAX_FEE_BPS
```

vault_program/src/initialize_vault.rs#L56-L56

```
180     if reward_fee_bps > MAX_BPS {
```

vault_core/src/vault.rs#L180-L180

Although in the current version, `MAX_FEE_BPS` and `MAX_BPS` have the same value of 10,000, as the code continues to be updated, if these two values diverge, this inconsistency may lead to abnormal *_fee_bps values.

The same issue exists in the variables `Vault.deposit_fee_bps`, `Vault.withdrawal_fee_bps`, and the functions and `Vault.check_fee_change_ok` `Vault.check_reward_fee_effective_rate`.

Unclear Rent Receiver in close_vault_update_state_tracker

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

The normal flow of vault updating should be:

1. initialize_vault_update_state_tracker
2. crank_vault_update_state_tracker (0 to N times)
3. close_vault_update_state_tracker

All of these instructions could be invoked by anyone at the beginning of each epoch.

The `vault_update_state_tracker_info` account will be closed at the end of `close_vault_update_state_tracker` IX, and the rent of this account will be transferred to the IX signer. The `permissionless` nature of this IX means that anyone who invoke it could receive the rent.

```
88  close_program_account(program_id, vault_update_state_tracker_info,
    ↪  payer)?;
```

vault_program/src/close_update_state_tracker.rs#L88-L88

Suppose there is a malicious user who could monitor and wait for the vault state until the condition `tracker.all_operators_updated == true` is met. Then the malicious user attempts to send `close_vault_update_state_tracker` IX before the cranker, allowing him to receive the rent released when closing the `vault_update_state_tracker_info` account.

It is recommended to record the rent payer in `initialize_vault_update_state_tracker` IX. For example, a field could be added to `VaultUpdateStateTracker` account, allowing the rent to be returned to the original payer when `close_vault_update_state_tracker` is invoked.

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.

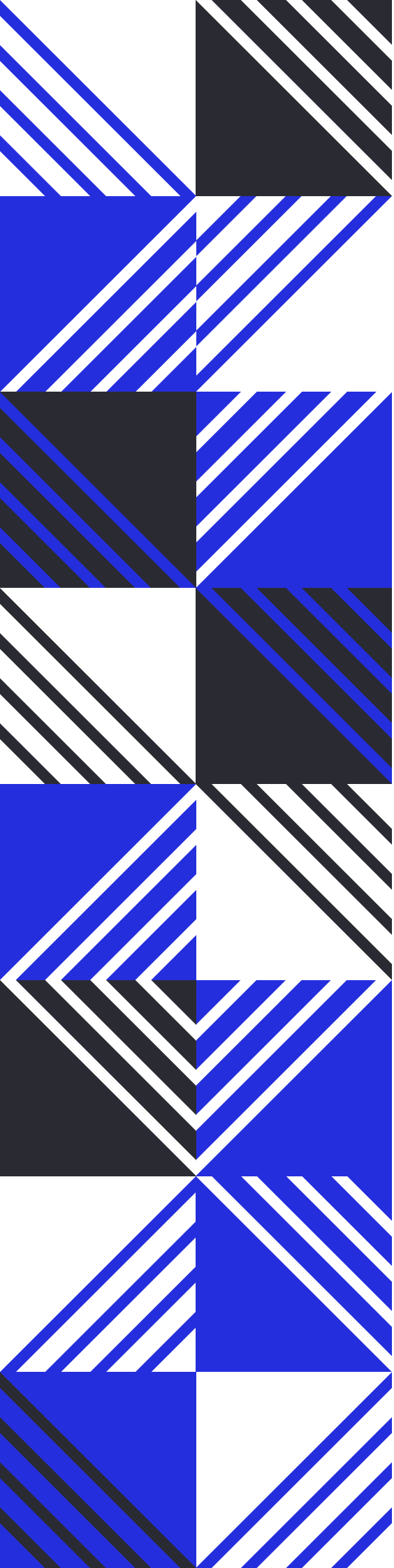
We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster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