OFFSIDE LABS

# Jupiter Lend Vaults

## Smart Contract Security Assessment

**August 2025**

**Prepared for:**

**Jupiter**

**Prepared by:**

**Offside Labs**

*Yao Li*

*Siji Feng*

# 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 *Fluid Solana* smart contracts, starting on July 23th, 2025, and concluding on August 4th, 2025.

### Project Overview

Fluid's Vault protocol enhances traditional mechanisms for locking collateral and borrowing debt by allowing users to borrow up to 95% of their assets' value, significantly improving capital efficiency. It features an innovative liquidation process inspired by Uniswap v3, reducing market impact and penalties to as low as 0.1%. Additionally, a robust oracle system provided by Pyth ensures accurate pricing data, enhancing security by providing multiple checks against price manipulation and bad debt liquidation.

### Audit Scope

The assessment scope contains mainly the smart contracts of the *vaults*, *oracle*, *lending_reward_rate_model* program for the *Fluid Solana* project.

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

- *Fluid Solana:*
  - Codebase: `https://github.com/Instadapp/fluid-contracts-solana`
  - Branch: audit-2-vault
  - Commit Hash: 59dc55276167f019fc1699995651103edade1c9f

We listed the files we have audited below:

- *Fluid Solana:*
  - programs/oracle/src/*.rs
  - programs/vaults/src/*.rs
  - programs/lendingRewardRateModel/src/*.rs

### Findings

The security audit revealed:

- 0 critical issue
- 1 high issue
- 2 medium issues
- 1 low issue
- 5 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 | Improper Use of Zero as Sentinel Value in connected_minima_tick | High | Fixed |
| 02 | Lack of Oracle Freshness Validation | Medium | Fixed |
| 03 | Limited Range of Liquidation Slippage | Medium | Fixed |
| 04 | Call update_rate Before Starting New Reward Cycle | Low | Fixed |
| 05 | Unused Account | Informational | Fixed |
| 06 | TWO_POWER_64 Inaccuracy | Informational | Fixed |
| 07 | Inaccurate Supply Only Position State Update | Informational | Partially Fixed |
| 08 | Suggestion for Optimizing Absorb Ticks Search | Informational | Acknowledged |
| 09 | Absorb May Be Blocked in Edge Case | Informational | Acknowledged |

# 4 Key Findings and Recommendations

## 4.1 Improper Use of Zero as Sentinel Value in connected_minima_tick

| Severity: High | Status: Fixed |
| --- | --- |
| Target: Vaults Program | Category: Logic Error |

### Description

The vaults program use tick 0 as sentinel value of the `connected_minima_tick` of the base branch at the following code positions:

1. programs/vaults/src/state/structs.rs#L230
2. programs/vaults/src/module/user.rs#L891
3. programs/vaults/src/module/user.rs#L927
4. programs/vaults/src/state/structs.rs#L312
5. programs/vaults/src/state/branch.rs#L57

The issue is that, the normal liquidation at the `liquidation_tick` of -1 can also set the `minima_tick` of a valid branch to 0.

When loading the `minima_tick` of 0 into memory, it will be rewritten to `i32:MIN` .

### Impact

If a new branch is created and connected to the affected branch with `minima_tick` 0, the `is_ref_tick_liquidated` status can't be reached anymore because the following `branch.minima_tick` is rewritten to `i32:MIN` .

```
655     (current_data.ref_tick, current_data.ref_tick_status) =
656         get_next_ref_tick(branch.minima_tick, next_tick,
    ↪ liquidation_tick)?;
```

programs/vaults/src/module/user.rs#L655-L656

This reserved value violation halts liquidation roll-through for the branches merged into the branch with `minima_tick` 0, creating toxic debt accumulation vectors.

### Recommendation

Using `i32:MIN` as the sentinel value of the base branch.

Note the process of resetting branches to connect them with the base branch, especially in the absorb process, which calling function `reset_branch_data` .

### Mitigation Review Log

Fixed in the commits 8ecdb86a39bb1fe2f0b68f7def474418968bee48 and 0b5aa1cc2929cbd3e7a3149393af717477cd3222.

## 4.2 Lack of Oracle Freshness Validation

| Severity: Medium | Status: Fixed |
| --- | --- |
| Target: Oracle Program | Category: Data Validation |

### Description

The `read_pyth_source` function doesn't check if the `publish_time` is fresh enough.

### Impact

When Pyth pull-price feeds become unresponsive for extended periods, the vault persistently calculates threshold ticks using stale prices.

This behavior would cause inaccurate liquidation triggers and bad debt accumulation.

### Recommendation

Ensure the `publish_time` is not stale.

### Mitigation Review Log

Fixed in commit 08a8d061673edf0bcae26c398f157b9ca63d8164.

## 4.3 Limited Range of Liquidation Slippage

| Severity: Medium | Status: Fixed |
| --- | --- |
| Target: Vaults Program | Category: Math Error |

### Description

The `liquidate` instruction uses a `u64` type parameter `col_per_unit_debt` for slippage protection. This parameter represents the minimum collateral expected per unit of debt paid back, expressed with 18 decimal precision.

```rust
pub fn liquidate<'info>(
    ctx: Context<'_, '_, 'info, 'info, Liquidate<'info>>,
    debt_amt: u64,
    col_per_unit_debt: u64, // min collateral needed to receive per unit of debt paid back in 1e18
    absorb: bool,
    remaining_accounts_indices: Vec<u8>, // first index is sources, second is branches, third is ticks, fourth is tick has debt
) -> Result<(u128, u128)> {
```

programs/vaults/src/module/user.rs#L451-L457

However, since `col_per_unit_debt` is defined as a `u64`, its integer part is limited by the range of `u64::MAX / 10^18`.

### Impact

The slippage protection mechanism may fail due to the limited range of the `u64` type.

### Recommendation

Use the u128 type instead of u64.

### Mitigation Review Log

Fixed in commit 07ca3bb16691cd1cf3497795074944f25fe9b29f.

## 4.4 Call update_rate Before Starting New Reward Cycle

| Severity: Low | Status: Fixed |
| --- | --- |
| Target: LendingRewardRateModel Program | Category: Logic Error |

### Description

The function `LendingRewards.start` is used to start a new reward cycle if the previous one has completed at the current time. This operation will also override configs of the previous completed reward period.

The issue is that, the `Lending` instance linked to the completed reward period might haven't been updated when starting the new reward period.

### Impact

A part of rewards supplied by the pervious reward period will be lost.

#### Recommendation

Call `accounts.update_rate()?` before override the configuration of the pervious reward period in the `LendingRewards.start` function.

#### Mitigation Review Log

Fixed in commit 6e84bcffaed8f1e7c597d3761d82c5f0f12653f4.

## 4.5 Informational and Undetermined Issues

#### Unused Account

| Severity: Informational | Status: Fixed |
| --- | --- |
| Target: Vaults Program | Category: Optimization |

The `top_tick` account of liquidate instruction is never used.

programs/vaults/src/state/context.rs#L585-L585

#### TWO_POWER_64 Inaccuracy

| Severity: Informational | Status: Fixed |
| --- | --- |
| Target: Library | Category: Math |

The constant `TWO_POWER_64` should be defined as `1 << 64` rather than `u64::MAX`, since `u64::MAX` is actually 1 less than $2^{64}$.

#### Inaccurate Supply Only Position State Update

| Severity: Informational | Status: Partially Fixed |
| --- | --- |
| Target: Vaults Program | Category: Logic Error |

1. When a user's position transitions to a supply-only position after an operation, only the `tick` field in `memory_vars` is set to `i32::MIN`, while the `tick_id` field remains unchanged.

   programs/vaults/src/module/user.rs#L282-L282

2. Additionally, the `tick` field of a position will be set to `0` if it becomes a supply-only position.

   programs/vaults/src/state/position.rs#L216-L216

These implementations may lead to an inaccurate state for the `tick` and `tick_id` fields in supply-only positions. However, it does not have any functional impact.

Mitigation Review Log:

1. There should be no impact and we will prefer to keep things for this the same way as for EVM, contracts/protocols/vault/vaultT1/coreModule/main.sol#L395 and contracts/protocols/vault/vaultT1/coreModule/main.sol#L479C18-L479C27
2. Fixed in commit/96e6fa6459acc018f6173457a80ab2004e745a53

### Suggestion for Optimizing Absorb Ticks Search

| Severity: Informational | Status: Acknowledged |
| --- | --- |
| Target: Vaults Program | Category: Logic Error |

The `fetch_next_tick_absorb` function does not clear the current tick, making the `safe_add(1)` adjustment to `current_tick` parameter unnecessary.

```
873         .fetch_next_tick_absorb(
874             tick_accounts,
875             vault_state.topmost_tick.safe_add(1)?,
876             max_tick,
877         )?;
```

programs/vaults/src/module/user.rs#L873-L877

In the worst case, this optimization can reduce the traversal of a map.

### Absorb May Be Blocked in Edge Case

| Severity: Informational | Status: Acknowledged |
| --- | --- |
| Target: Vaults Program | Category: Logic Error |

When the condition `liquidation_tick + 1 == tick_info.tick && tick_info.partials == 1` is reached in the liquidate instruction, the instruction will panic: programs/vaults/src/module/user.rs#L608-L610

Although there is nothing to be liquidated, the liquidate instruction should still be used to absorb bad debts.

Therefore, it is recommended to skip liquidation here and proceed directly with the subsequent absorb process, rather than throwing a panic.

# 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.

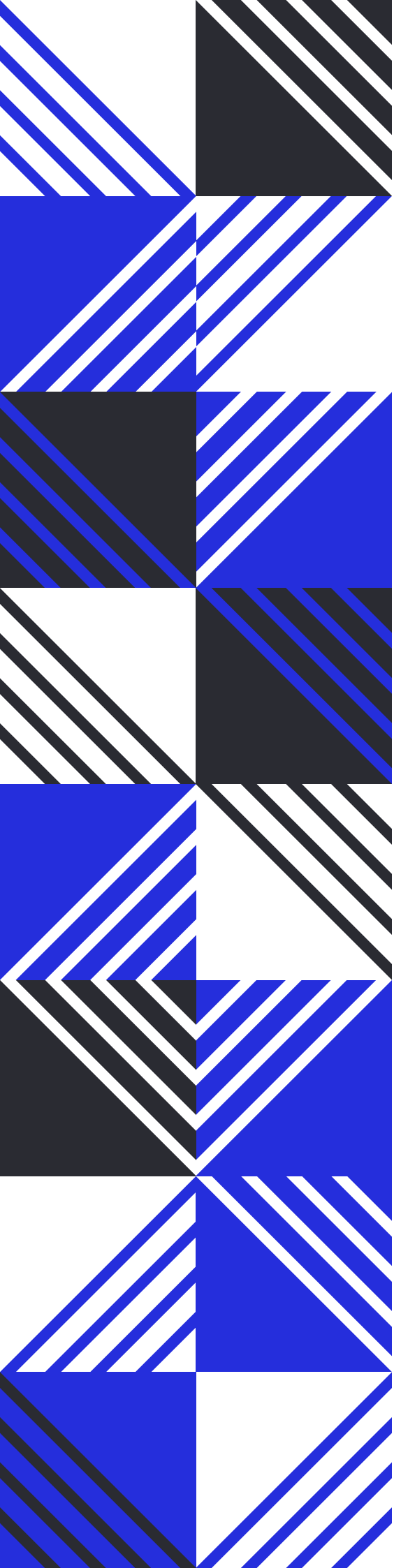
We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster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