OFFSIDE LABS

Jupiter Aggregator v6

Smart Contract Security Assessment

October 2025

Prepared for:

Jupiter

Prepared by:

Offside Labs

Sirius Xie

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 *Jupiter Aggregator* smart contracts, starting on August 1st, 2025, and concluding on October 2nd, 2025.

Project Overview

Jupiter Aggregator is a decentralized exchange aggregator on Solana that finds the best rates for swapping SPL tokens. It routes trades across multiple liquidity sources to deliver optimal prices, low slippage, and efficient execution. Users benefit from a seamless interface, deep aggregated liquidity, and the ability to perform complex, multi-hop swaps within a single transaction.

In the latest release, the aggregator adds V2 route instructions along with a new `swapEvent` schema, introduces a positive slippage fee, and expands users' flexibility and choice when executing trades.

Audit Scope

The assessment scope contains mainly the smart contracts of the jupiter-aggregator-program program for the *Jupiter Aggregator* project.

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

- Jupiter Aggregator
  - Codebase: `https://github.com/jup-ag/jupiter-aggregator-program`
  - Commit Hash: 41d454b84058a9680441b779eb46dc2a00f7d4f1

We listed the files we have audited below:

- Jupiter Aggregator
  - programs/jupiter/src/**/*.rs

Findings

The security audit revealed:

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

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	Inconsistent and Discontinuous Positive Slippage Fee Calculation in calculate_exact_out_positive_slippage_fee	High	Fixed
02	Fee May Overcharge Beyond Positive Slippage in calculate_exact_in_positive_slippage_fee	High	Fixed
03	Possible Positive Slippage Fee Loss Due to Missing None Check	Medium	Fixed
04	Missing Memo Support for Token-2022 Transfers	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Inconsistent and Discontinuous Positive Slippage Fee Calculation in calculate_exact_out_positive_slippage_fee

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

Description

In the `exact_out_route` and `shared_accounts_exact_out_route` IX, a `positive_slippage_fee` is charged based on the result of the swap.

According to the implementation of `calculate_exact_out_positive_slippage_fee`, the fee can be defined as follows:

Let:

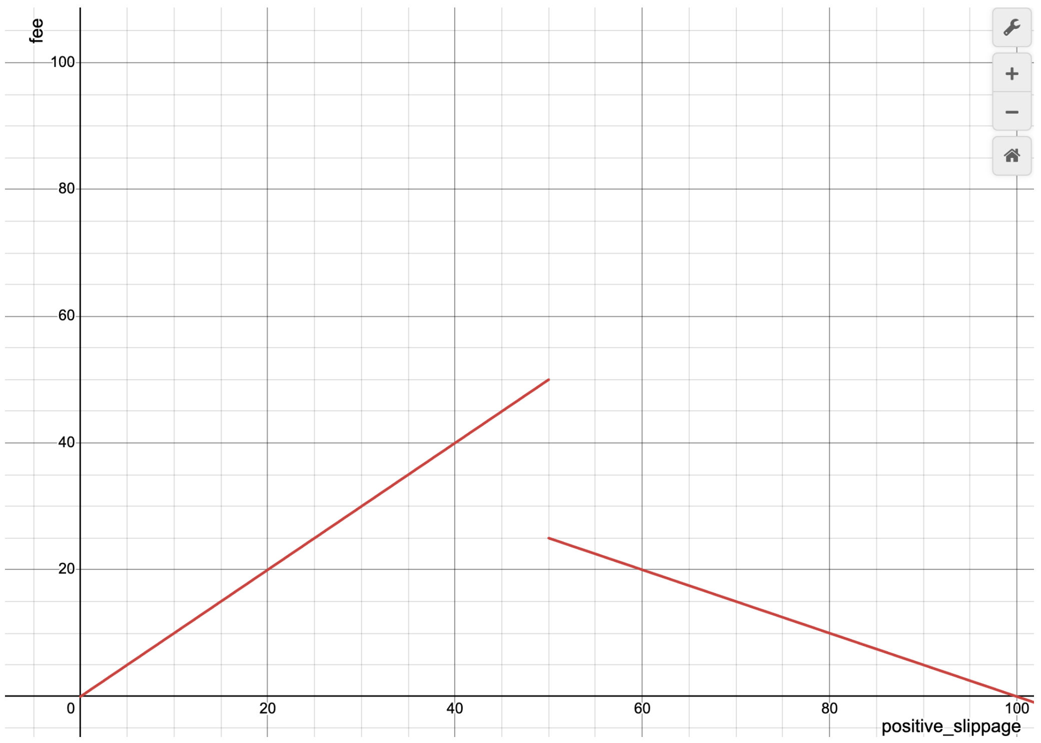
- x: `positive_slippage`
- y: `positive_slippage_fee`
- R: `positive_slippage_bps` (Assume it's an invariable here)
- Q: `quoted_in_amount` (Assume it's an invariable here)
- A: `in_amount`, which is $(Q - x)$

$$y = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } 0 < x \leq Q \cdot R \\ (Q - x) \cdot R & \text{if } x > Q \cdot R \end{cases}$$

From the definition, we can see that when $x > Q \cdot R$, as x increases, y gradually decreases. This implies that as the difference between `in_amount` and `quoted_in_amount` increases, the resulting fee actually becomes smaller.

Moreover, when x increases from $Q \cdot R$ to $Q \cdot R + 1$, the function y is not continuous.

Here is a demo curve for above formula, where $Q = 100$, $R = 0.5$



Impact

Generally, as the `positive_slippage` increases, the `positive_slippage_fee` is expected to increase accordingly. However, in this case, the `positive_slippage_fee` decreases as the `positive_slippage` grows, which may be inconsistent with the expected design of the fee.

In addition, the current tiered fee structure does not guarantee a monotonically increasing fee. There is even a discontinuous point between segments, meaning the segments are not connected continuously.

Recommendation

It is recommended to revise the fee calculation formula to ensure that the piecewise function is both increasing and continuous.

Mitigation Review Log

Fixed in the commit 20af53e33ef128b54f2c4cb48ecd04a033124cca.

4.2 Fee May Overcharge Beyond Positive Slippage in calculate_exact_in_positive_slippage_fee

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

Description

In the `route` and `shared_accounts_route` IX, a `positive_slippage_fee` is charged based on the result of the swap.

According to the implementation of `calculate_exact_in_positive_slippage_fee` , the fee can be defined as follows:

Let:

- x: `positive_slippage`
- y: `positive_slippage_fee`
- R: `positive_slippage_bps` (Assume it's an invariable here)
- Q: `quoted_out_amount` (Assume it's an invariable here)
- A: `out_amount` , which is (Q + x)

$$y = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } 0 < x \leq Q \cdot R \\ (Q + x) \cdot R & \text{if } x > Q \cdot R \end{cases}$$

From the definition, we can see that when x increases from $Q \cdot R$ to $Q \cdot R + 1$, the function y is not continuous.

Here is a demo curve for above formula, where $Q = 100$, $R = 0.5$

Further more,When $x > Q \cdot R$, y may be larger than x, which means `positive_slippage_fee` is larger than `positive_slippage` . This formula does not mathematically guarantee that the `positive_slippage` is always greater than or equal to the `positive_slippage_fee` .

Here is an example:

```
Q = 100
A = 121
R = 0.2
Q * R = 20

x = A - Q = 121 - 100 = 21
x > Q * R

y = (Q + x) * R = A * R = 24 > x

Finally, user could only receive 121-24 = 97, which is less then Q.
```

Impact

There is a discontinuous point between segments in the current tiered fee structure, meaning the segments are not connected continuously.

Since the `positive_slippage_fee` can exceed the actual `positive_slippage`, it could result in users receiving a lower swap output amount due to the positive slippage fee.

Recommendation

It is recommended to revise the fee calculation formula to ensure that the piecewise function is `continuous`, and to mathematically guarantee that:

$$positive\_slippage \geq positive\_slippage\_fee$$

One possible solution could be like this one, which ensure the `continuous` and not exceed the actual `positive_slippage`.

$$y = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } 0 < x \leq Q \cdot R \\ (Q + x) \cdot \frac{R}{1+R} & \text{if } x > Q \cdot R \end{cases}$$

Mitigation Review Log

Fixed in the commit 20af53e33ef128b54f2c4cb48ecd04a033124cca.

4.3 Possible Positive Slippage Fee Loss Due to Missing None Check

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

Description

In the `route` IX, if `positive_slippage_fee > 0`, it will be deducted from `out_amount_after_fees`.

```
        if positive_slippage_fee > 0 {
            if let Some(positive_slippage_fee_account) =
             ↪  positive_slippage_fee_account {
                token_transfer(
                    destination_token_program,
                    user_destination_token_account,
                    destination_mint,
                    positive_slippage_fee_account,
                    user_transfer_authority,
                    &[],
                    positive_slippage_fee,
                )?;
            }

            out_amount_after_fees = out_amount_after_fees
                .checked_sub(positive_slippage_fee)
                .ok_or(JupiterError::InvalidCalculation)?;
        }
```

programs/jupiter/src/lib.rs#L341-L357

The resulting `out_amount_after_fees` is then transferred to the user's `token_account`.

```
        if user_destination_token_account
            .key()
            .ne(destination_token_account.key())
        {
            token_transfer(
                destination_token_program,
                user_destination_token_account,
                destination_mint,
                destination_token_account,
                user_transfer_authority,
                signer_seeds,
                out_amount_after_fees,
            )?;
        }
```

programs/jupiter/src/lib.rs#L365-L378

However, the case where `positive_slippage_fee_account` is `None` is not handled here. If the IX does not include this account in `remaining_accounts`, then `positive_slippage_fee_account` can indeed be `None`.

```rust
pub fn extract_first_account_if_fee_not_zero<'a>(
    fee_bps: u16,
    remaining_accounts: &mut &'a [NoStdAccountInfo],
) -> Option<&'a NoStdAccountInfo> {
    if fee_bps == 0 || remaining_accounts.is_empty() {
        return None;
    }
        ...
```

programs/jupiter/src/account_infos.rs#L41-L47

Impact

when `positive_slippage_bps > 0` and `positive_slippage_fee_account` is `None`, the `positive_slippage_fee` will not be transferred to `positive_slippage_fee_account`.

In this case, the `positive_slippage_fee` may remain in the intermediate `user_destination_token_account` used in the route swap, and neither the `positive_slippage_fee_account` nor the user's `destination_token_account` will receive this fee.

The same issue also exists in the `shared_accounts_route` IX.

A similar issue also exists in `exact_out_route` and `shared_accounts_exact_out_route`, but the impact is different. In this case, it only results in the `positive_slippage_fee_account` not receiving the fee, while the user does not actually pay the fee.

Recommendation

It is recommended to check that `positive_slippage_fee_account` is not `None` in both the `route` and `shared_accounts_route` IXs before deducting the fee from `out_amount_after_fees`.

```
        if positive_slippage_fee > 0 {
            if let Some(positive_slippage_fee_account) =
             ↪  positive_slippage_fee_account {
                token_transfer(
                    destination_token_program,
                    user_destination_token_account,
                    destination_mint,
                    positive_slippage_fee_account,
                    user_transfer_authority,
                    &[],
                    positive_slippage_fee,
                )?;
                out_amount_after_fees = out_amount_after_fees
                    .checked_sub(positive_slippage_fee)
                    .ok_or(JupiterError::InvalidCalculation)?;
            }
        }
```

Alternatively, the IX can perform an earlier check to prevent such cases, i.e., where `positive_slippage_bps > 0` and `positive_slippage_fee_account` is `None`.

Mitigation Review Log

Fixed in the commit 87fc19c2949282eae8c5baa021cb81f8a2bc7086.

4.4 Informational and Undetermined Issues

Missing Memo Support for Token-2022 Transfers

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Token

In Token-2022, users can choose to enable the `Memo` extension on their token accounts, which requires a separate Memo instruction to be included before any transfer. And whether the user enables Memo is independent of the Mint. However, the `token_transfer` function currently does not handle logic related to the Memo extension. It is recommended to add support for Memo to prevent transfer failures caused by this requirement.

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