OFFSIDE LABS

Jupiter RFQ

Smart Contract Security Assessment

November 2024

Prepared for:

Jupiter

Prepared by:

Offside Labs

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 *RFQ* smart contracts, starting on *November 7th, 2024*, and concluding on *November 9th, 2024*.

Project Overview

Jupiter's Request For Quote (RFQ) module aims to improve liquidity on Solana by enabling users to fill large orders in a decentralized and trustless manner through a network of market makers.

The RFQ flow involves users requesting quotes via the frontend, with the Jupiter RFQ API forwarding these requests to market makers for competitive quotes. The API selects the best quote, creates a transaction, and works with users and market makers to finalize it. Integration requires webhook registration, strict timing, and expiry management to ensure efficiency, with future plans for custom transaction crafting and a unified API.

Audit Scope

The assessment scope contains mainly the smart contracts of the order-engine program for the *RFQ* project.

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

- RFQ
  - Codebase: https://github.com/jup-ag/rfq
  - Commit Hash: 295c39f49a40749ea1314555ad98fb4e518db4a7

We listed the files we have audited below:

- RFQ
  - programs/order-engine/*/*.rs
  - rfq-common/src/fill.rs

Findings

The security audit revealed:

- 0 critical issue
- 2 high issues
- 0 medium issue
- 1 low issues
- 1 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 Logical Operator in Account Comparison	High	Fixed
02	Incomplete Instruction Comparison Due to zip() Usage	High	Fixed
03	Unexpected SOL Balance Reduction from Temporary Account Rent	Low	Fixed
04	Ineffective Assertion in validate_fill_sanitizedx_message	Informational	Fixed

4 Key Findings and Recommendations

4.1 Incorrect Logical Operator in Account Comparison

Severity: High	Status: Fixed
Target: SDK	Category: Logic Error

Description

The code is meant to ensure instructions are identical by comparing `program_id`, `accounts`, and `data`. It mistakenly uses `&&` instead of `||` when comparing account attributes.

```
202         // Must be identical to original
203         if program_id != original_program_id
204             || accounts
205                 .iter()
206                 .zip(original_accounts)
207                 .any(|(accounts, original_accounts)| {
208                     accounts.pubkey != original_accounts.pubkey
209                         && accounts.is_signer !=
    ↪ original_accounts.is_signer
210                         && accounts.is_writable !=
    ↪ original_accounts.is_writable
211                 })
212             || data != original_data
213         {
214             bail!("Instruction did not match the original at index
                ↪ {index}, {original_program_id}");
215         }
```

rfq-common/src/fill.rs#L202-L215

The user can replace the original account with a different public key but the same attributes (`is_signer` or `is_writable`), allowing it to pass validation.

Recommendation

Use `||` to correctly identify any differences in account properties.

Mitigation Review Log

Fixed in the PR81, commit f48de20009b1b8f09eccffc037219aa028c71af9.

4.2 Incomplete Instruction Comparison Due to zip() Usage

Severity: High	Status: Fixed
Target: SDK	Category: Logic Error

Description

The code uses `zip()` to iterate over instructions from `sanitized_message` and `original_sanitized_message`. If the two have different numbers of instructions, `zip()` silently stops at the shortest, missing extra or absent instructions.

```
164     for (
165         index,
166         (
167             BorrowedInstruction {
168                 program_id,
169                 accounts,
170                 data,
171             },
172             BorrowedInstruction {
173                 program_id: original_program_id,
174                 accounts: original_accounts,
175                 data: original_data,
176             },
177         ),
178     ) in sanitized_message
179         .decompile_instructions()
180         .into_iter()
181         .zip(original_sanitized_message.decompile_instructions())
182         .enumerate()
```

rfq-common/src/fill.rs#L164-L182

This prevents detection of missing or extra instructions, leading to incomplete validation and potentially accepting incorrect messages.

Recommendation

Ensure that both messages have the same number of instructions before using `zip()`. Validate the instruction counts and handle any discrepancies accordingly.

Mitigation Review Log

Fixed in the PR82, commit 22b0344956ff60a1a9b5430a88b546ad83603a79.

4.3 Unexpected SOL Balance Reduction from Temporary Account Rent

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Corner Case

Description

If the maker provides SOL from a WSOL token account, we need to unwrap the SOL using a temporary account. We transfer the specified amount of SOL from the source account to the temporary account, then close the account and return the SOL balance to the maker. This process ensures the maker has enough unwrapped SOL for the final transfer to the taker.

Creating a temporary account requires a temporary rent charge from the maker. Since the rent is refunded when the WSOL token account is closed, we don't need to unwrap the full amount of WSOL specified. Instead, we reduce the amount sent from the source account accordingly.

```
247     let amount_to_send = if receiver.is_some() {
248
        ↪ amount.saturating_sub(temporary_wsol_token_account.get_lamports())
249     } else {
250         amount
251     };
```

programs/order-engine/src/instructions/fill.rs#L247-L251

Impact

This optimization can lead to unintended consequences:

1. The `output_amount` does not always match the reduction in the maker_output_mint_-token_account.
2. Each time the maker sends an `output_amount` from their WSOL token account, rent (about 0.002 SOL) is deducted, which may deplete their balance faster than expected.

Recommendation

Always send the exact `output_amount` of SOL from the specified token account to keep the maker's balance unchanged.

Mitigation Review Log

Fixed in the PR80, commit 8c17852ce2f1f40b0d641c683cc51c48efaf64a5.

4.4 Informational and Undetermined Issues

Ineffective Assertion in validate_fill_sanitized_message

Severity: Informational	Status: Fixed
Target: SDK	Category: Redundant Code

The assertion in `validate_fill_sanitized_message` is ineffective, as it checks if `taker` equals itself. It should instead verify `taker == order.taker`.

```
109             ensure!(taker == taker, "Invalid taker");
```

rfq-common/src/fill.rs#L109

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.

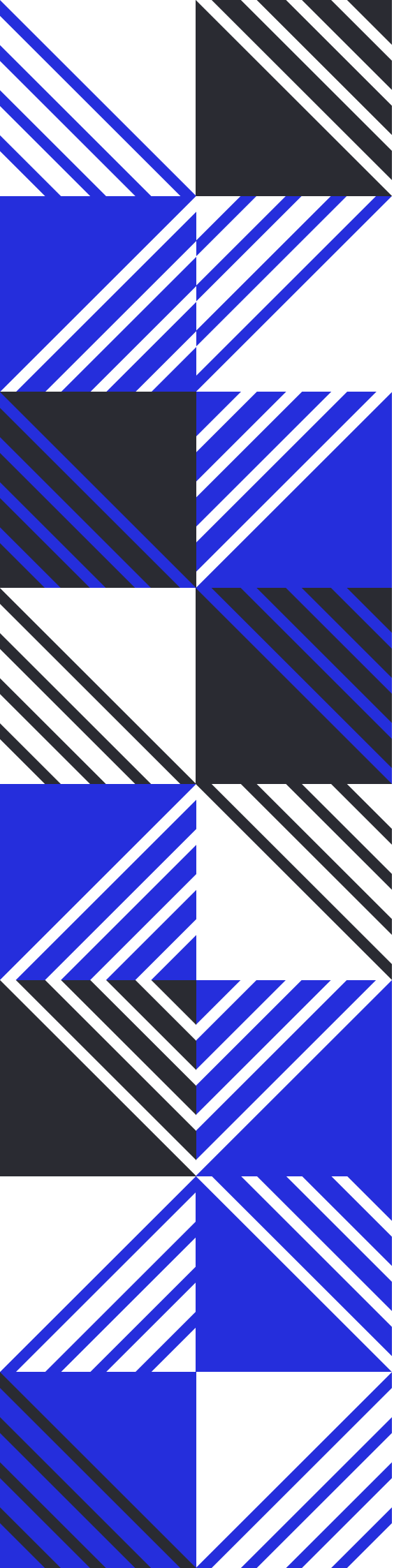
We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster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