OFFSIDE LABS

Bebop RFQ

Smart Contract Security Assessment

December 2025

Prepared for:

Bebop

Prepared by:

Offside Labs

Gongyu Shi

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 *Bebop RFQ* smart contracts, starting on December 7th, 2025, and concluding on December 8th, 2025.

Project Overview

Bebop RFQ is a market maker aggregator that splits basket trades across multiple makers to deliver the best composite price for the taker, and it supports multi-hop compositions where a shared PDA executes intermediary legs.

Audit Scope

The assessment scope contains mainly the smart contracts of the bebop_rfq program for the *Bebop RFQ* project.

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

- Bebop RFQ
  - Codebase: `https://github.com/bebop-dex/bebop_solana`
  - Commit Hash: 7719057f21f046f0d5dad7699ed70d9d3fabf606

We listed the files we have audited below:

- Bebop RFQ
  - programs/bebop_rfq/src/lib.rs
  - programs/bebop_rfq/src/instructions/swap.rs
  - programs/bebop_rfq/src/instructions/utils.rs

Findings

The security audit revealed:

- 0 critical issue
- 0 high issue
- 1 medium issue
- 1 low issue
- 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	Missing Signer Seeds for Shared-PDA Taker During Transfers	Medium	Acknowledged
02	Integer Overflow in Filled Maker Amount Calculation	Low	Fixed
03	Missing Memo Transfer Check for Token-2022 Account	Informational	Acknowledged
04	Zero Output Amount May Cause Misleading OrderExpired Error	Informational	Acknowledged
05	Token Accounts Are Not Enforced as ATAs	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Missing Signer Seeds for Shared-PDA Taker During Transfers

Severity: Medium	Status: Acknowledged
Target: Smart Contract	Category: Access Control

Description

The taker can be a shared PDA, but in several cases the signer seeds of this PDA are missing when transferring tokens from the taker to the maker:

- If `taker_input_mint_token_account` is `None`, a lamport transfer is executed, but the signer seeds are not provided.

```
67      system_program::transfer(
68          CpiContext::new(
69              ctx.accounts.system_program.to_account_info(),
70              system_program::Transfer {
71                  from: ctx.accounts.taker.to_account_info(),
72                  to: ctx.accounts.maker.to_account_info(),
73              },
74          ),
75          filled_taker_amount,
76      )?;
```

programs/bebop_rfq/src/instructions/swap.rs#L67-L76

- If `taker_input_mint_token_account` is not `None` and `maker_input_mint_token_account` is `None`, `unwrap_sol` is called and the wrapped SOL is transferred from the taker's token account to the temporary account, again without signer seeds.

```
119     token::transfer(
120         CpiContext::new(
121             token_program.clone(),
122             token::Transfer {
123                 from: sender_token_account.clone(),
124                 to: temporary_wsol_token_account.clone(),
125                 authority: sender.clone(),
126             },
127         ),
128         amount,
129     )?;
```

programs/bebop_rfq/src/instructions/utils.rs#L119-L129

Impact

Without the signer seeds, the transaction will fail when a shared PDA taker is involved in these cases, causing a DoS.

Recommendation

If swaps of wrapped SOL or SOL are allowed for a shared PDA taker, include the appropriate signer seeds in the corresponding CPI contexts. Otherwise, add upfront checks to detect such cases and return a clear error.

Mitigation Review Log

Bebop Team: We don't plan to use SOL as taker's input token for shared PDA cases. Moreover, we can guarantee that whenever a shared PDA is set as the taker and WSOL is transferred, the `maker_input_mint_token_account` is never `None`. Therefore, the DoS scenario will not occur in practice.

4.2 Integer Overflow in Filled Maker Amount Calculation

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Math Error

Description

The computation of `filled_maker_amount` performs a direct multiplication of two `u64` values (`output_amount * filled_taker_amount`) without widening or checked math. Large quotes will cause the intermediate product to exceed `u64::MAX` .

```
124 let filled_maker_amount: u64 = if filled_taker_amount < input_amount \
125     {output_amount * filled_taker_amount / input_amount} else
        ↪ {output_amount};
```

programs/bebop_rfq/src/instructions/swap.rs#L124-L124

Impact

An overflow in the intermediate multiplication will cause a panic and revert the transaction, which leads to a DoS condition.

Recommendation

Perform the intermediate calculation in `u128` to prevent overflow, then convert the final result back to `u64` .

Mitigation Review Log

Fixed in commit c5a9f5fe9dd672c4bdf21dccb7bcd7a6d495d8f0.

4.3 Informational and Undetermined Issues

Missing Memo Transfer Check for Token-2022 Account

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

In Token-2022, users can enable the `Memo` extension on their token accounts. This requires a separate `Memo` instruction to be included before any transfer. Note that whether a user enables this feature is independent of the token mint.

The current implementation of the program supports Token-2022 token accounts, but it does not handle the `Memo` extension.

It is recommended to add support for the `Memo` extension to prevent transfer failures, or explicitly check for this extension and throw an error indicating that it is not supported.

Zero Output Amount May Cause Misleading OrderExpired Error

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Code QA

When fetching the current `output_amount`, the only constraint is that `amount_with_expiry.amount <= output_amounts[i - 1].amount`, but it does not check that the amount is non-zero. As a result, an `OrderExpired` error may be thrown even when the order has not actually expired, which can be confusing.

```
27      let mut output_amount: u64 = 0;
28      for (i, amount_with_expiry) in output_amounts.iter().enumerate() {
29          require!(
30              i == 0 ||
31              (amount_with_expiry.amount <= output_amounts[i - 1].amount &&
    ↪ amount_with_expiry.expiry > output_amounts[i - 1].expiry),
32              BebopError::InvalidOutputAmount
33          );
34          if amount_with_expiry.expiry >= now {
35              output_amount = amount_with_expiry.amount;
36              break;
37          }
38      }
39      require!(output_amount > 0, BebopError::OrderExpired);
```

programs/bebop_rfq/src/instructions/swap.rs#L27-L39

It is suggested to add the check for `amount_with_expiry.amount` , and throw `InvalidOutputAmount` if it is zero.

Token Accounts Are Not Enforced as ATAs

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

The `swap` IX accepts four optional token accounts, and according to the test codes, these token accounts are expected to be the ATAs of the corresponding authorities. But the program does not enforce the passed-in accounts are ATAs, any token accounts with intended authority and mint are accepted.

```
211     #[account(
212         mut,
213         token::authority = taker,
214         token::mint = input_mint,
215         token::token_program = input_token_program
216     )]
217     pub taker_input_mint_token_account: Option<Box<InterfaceAccount<'info,
        ↪ TokenAccount>>>,
```

programs/bebop_rfq/src/instructions/swap.rs#L211-L217

It is recommended to use the `associated_token` constraints instead.

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.

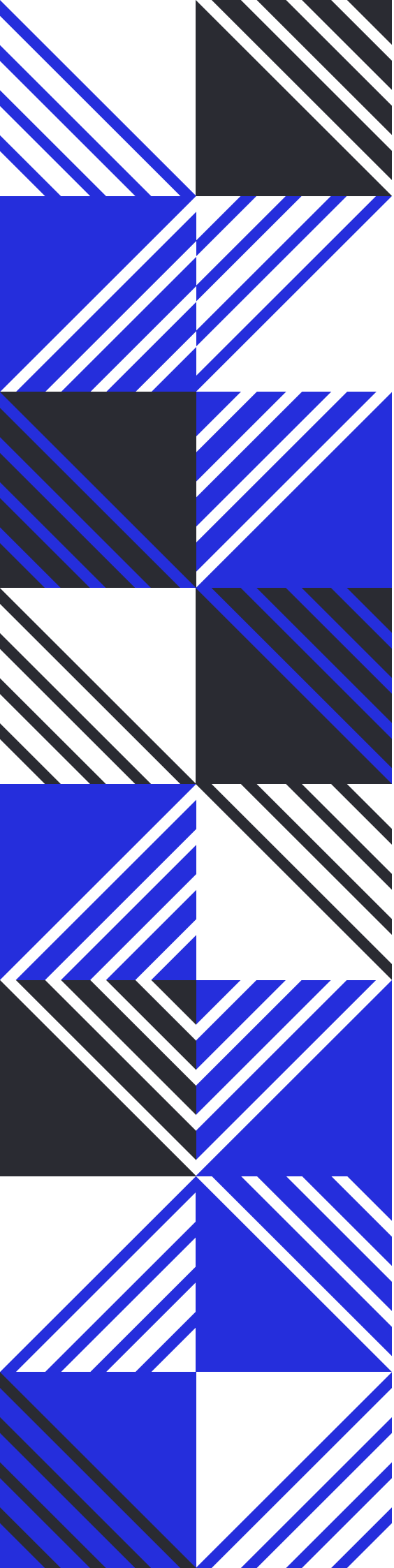
We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster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