Orca Whirlpools

Security Assessment

August 21st, 2024 — Prepared by OtterSec

Michał Bochnak embe221ed@osec.io

Robert Chen r@osec.io

Table of Contents

01 — Executive Summary

Overview

Orca engaged OtterSec to assess the `whirlpools` program. The audit and the two follow-up reviews were performed during the period shown below:

- March 13th and May 27th, 2023 (PR#87 and PR#89).
- April 16th and May 10th, 2024 (PR#134).
- July 16th and July 26th, 2024 (PR#159 and PR#161).

For more information on our auditing methodology, refer to Appendix B.

Key Findings

We produced 2 findings throughout this audit engagement.

In particular, we made recommendations regarding the need for adherence to coding best practices and improving the overall efficiency and readability of the codebase (OS-OCA-SUG-00). We also identified a potential suggestion where, when a token is burned and its associated position is closed, metadata accounts persist (OS-OCA-SUG-01).

Scope

The source code was delivered to us in a Git repository at https://github.com/orca-so/whirlpools.

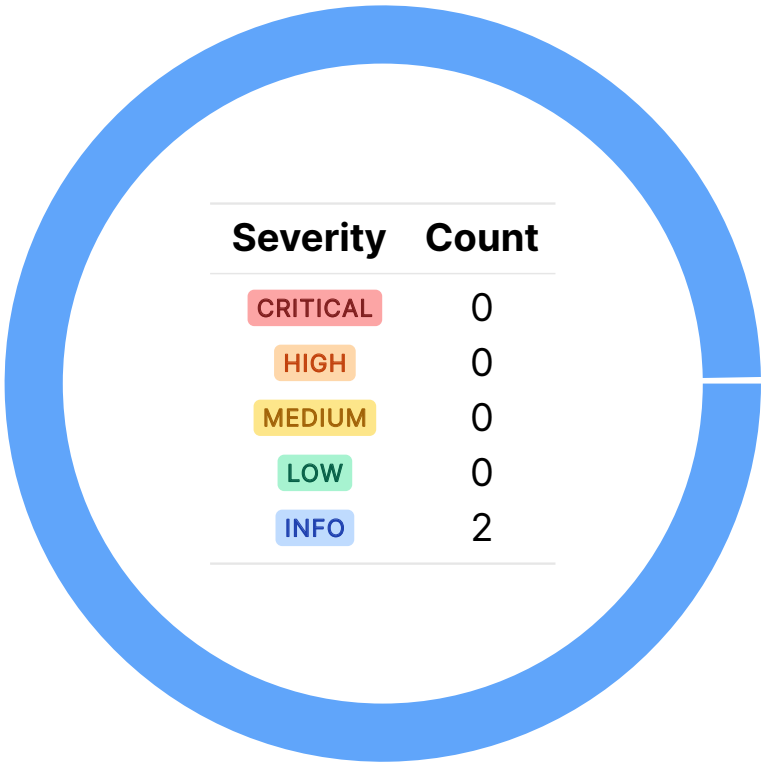
A brief description of the programs is as follows:

Name	Description
whirlpools	An open-source concentrated liquidity AMM contract on the Solana blockchain.

02 — Findings

Overall, we reported 2 findings.

We split the findings into **vulnerabilities** and **general findings**. Vulnerabilities have an immediate impact and should be remediated as soon as possible. General findings do not have an immediate impact but will aid in mitigating future vulnerabilities.



Severity	Count
CRITICAL	0
HIGH	0
MEDIUM	0
LOW	0
INFO	2

03 — General Findings

Here, we present a discussion of general findings during our audit. While these findings do not present an immediate security impact, they represent anti-patterns and may result in security issues in the future.

ID	Description
OS-OCA-SUG-00	Suggestions to ensure adherence to coding best practices.
OS-OCA-SUG-01	When a token is burned and its associated position is closed metadata accounts persist, resulting in potential conflicts if the same mint is reused.

Code Maturity OS-OCA-SUG-00

Description

1. The `is_mutable` flag is set to true in the call to `create_metadata_accounts_v3`, allowing the metadata to be updated. However, there is no instance in the code base where the metadata is updated. If there is no requirement to update metadata, it would be appropriate to make it immutable to reduce the overall risk factor.

>_ *whirlpool/src/util/token.rs* rust

```rust
pub fn mint_position_token_with_metadata_and_remove_authority<'info>(
    [...]
) -> Result<()> {
    [...]
    invoke_signed(
        &create_metadata_accounts_v3(
            metadata_program.key(),
            position_metadata_account.key(),
            position_mint.key(),
            metadata_mint_auth_account.key(),
            funder.key(),
            metadata_update_auth.key(),
            WP_METADATA_NAME.to_string(),
            WP_METADATA_SYMBOL.to_string(),
            WP_METADATA_URI.to_string(),
            None,
            0,
            false,
            true,
            None,
            None,
            None,
        )
        [...]
    )?;
```

2. Currently, `position_bundle`, `position` and `bundled_position` utilize a common prefix position. The utilization of common prefixes for seeds in Solana programs may pose potential risks in terms of security and future-proofing. If multiple accounts utilize the same seed prefix, there is a theoretical of address collisions. Note that in the current implementation, because the remaining seed byte sequence lengths are different, there is no risk of collision.
3. `parse_remaining_accounts` allows for flexibility in how the slices of remaining accounts are provided. However, this flexibility may result in ambiguity or inefficiency in downstream processing, specifically, the possibility of passing a slice for the same account type multiple times. While this

may not directly result in errors, it complicates downstream logic and potentially introduces bugs or inefficiencies.

Remediation

1. Set `is_mutable` flag to true in the call to `create_metadata_accounts_v3`.
2. Utilize different prefixes for different seeds.
3. Refactor `parse_remaining_accounts` to enforce stricter rules regarding the input slices.

Patch

Orca decided against updating the `is_mutable` parameter to maintain the flexibility to modify NFT metadata as needed.

Failure To Close Metadata Account OS-OCA-SUG-01

Description

`burn_checked` does not close the metadata account. When a user creates a token with metadata, the metadata is stored in a Program Derived Address (PDA) associated with that `mint`. If the user decides to close their position by burning the token using `burn_checked`, the metadata account remains open and retains its data. This may result in an issue if the user tries to re-open a position using the same `mint` because the metadata PDA already exists and may contain outdated or incorrect information.

Remediation

The program should ensure that the metadata account is also closed when the token is burned and the associated position is closed.

Patch

Orca decided to retain the Metadata, considering that the mint of the Position is an NFT and that the remaining Metadata would not have a negative impact in the future. Additionally, the drawbacks of creating the necessary MasterEdition account to close the Metadata were also significant.

A — Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings may be found in the General Findings.

CRITICAL

Vulnerabilities that immediately result in a loss of user funds with minimal preconditions.

Examples:

- Misconfigured authority or access control validation.
- Improperly designed economic incentives leading to loss of funds.

HIGH

Vulnerabilities that may result in a loss of user funds but are potentially difficult to exploit.

Examples:

- Loss of funds requiring specific victim interactions.
- Exploitation involving high capital requirement with respect to payout.

MEDIUM

Vulnerabilities that may result in denial of service scenarios or degraded usability.

Examples:

- Computational limit exhaustion through malicious input.
- Forced exceptions in the normal user flow.

LOW

Low probability vulnerabilities, which are still exploitable but require extenuating circumstances or undue risk.

Examples:

- Oracle manipulation with large capital requirements and multiple transactions.

INFO

Best practices to mitigate future security risks. These are classified as general findings.

Examples:

- Explicit assertion of critical internal invariants.
- Improved input validation.

B — Procedure

As part of our standard auditing procedure, we split our analysis into two main sections: design and implementation.

When auditing the design of a program, we aim to ensure that the overall economic architecture is sound in the context of an on-chain program. In other words, there is no way to steal funds or deny service, ignoring any chain-specific quirks. This usually requires a deep understanding of the program's internal interactions, potential game theory implications, and general on-chain execution primitives.

One example of a design vulnerability would be an on-chain oracle that could be manipulated by flash loans or large deposits. Such a design would generally be unsound regardless of which chain the oracle is deployed on.

On the other hand, auditing the program's implementation requires a deep understanding of the chain's execution model. While this varies from chain to chain, some common implementation vulnerabilities include reentrancy, account ownership issues, arithmetic overflows, and rounding bugs.

As a general rule of thumb, implementation vulnerabilities tend to be more "checklist" style. In contrast, design vulnerabilities require a strong understanding of the underlying system and the various interactions: both with the user and cross-program.

As we approach any new target, we strive to comprehensively understand the program first. In our audits, we always approach targets with a team of auditors. This allows us to share thoughts and collaborate, picking up on details that the other missed.

While sometimes the line between design and implementation can be blurry, we hope this gives some insight into our auditing procedure and thought process.