

Security Assessment Report

Orca Whirlpools

PRs 1311, 1317 and 1326

August 11, 2026

Summary

The Sec3 team was engaged to conduct an incremental review of PRs 1311, 1317 and 1326 of the Orca Whirlpools.

The artifact of the audit was the source code of the following programs, excluding tests, in https://github.com/orca-so/whirlpools.

The initial audit focused on the following versions and revealed 1 issues or questions.

#	Task	Type	Commit
P1	PR1311 - Remove migrate_repurpose_reward_authority_space ix	Solana	ad183d8 → 272c70a
P2	PR1317 - Prepare/Commit (aka on-chain quote interface)	Solana	8cea584 → 1fe7341
P3	PR1326 - Fix: u256 div edge case	Solana	14d4834 → 950d7a8

This report provides a detailed description of the findings and their respective resolutions.

Table of Contents

Result Overview 3
Findings in Detail 4
[P2-I-01] `TransferFee` reinitialization breaks `prepare/commit` assumptions 4
Appendix: Methodology and Scope of Work 7

Result Overview

Issue	Impact	Status
PR1311 - REMOVE MIGRATE_REPURPOSE_REWARD_AUTHORITY_SPACE IX		
No issues found		
PR1317 - PREPARE/COMMIT (AKA ON-CHAIN QUOTE INTERFACE)		
[P2-I-01] `TransferFee` reinitialization breaks `prepare/commit` assumptions	Info	Resolved
PR1326 - FIX: U256 DIV EDGE CASE		
No issues found		

Findings in Detail

PR1317 - PREPARE/COMMIT (AKA ON-CHAIN QUOTE INTERFACE)

[P2-I-01] `TransferFee` reinitialization breaks `prepare/commit` assumptions

Identified in commit `1fe7341`.

In the on-chain quote feature, the program enforces that `prepare_swap` and `commit_swap` execute within the same `slot`, in order to ensure atomic execution between the prepare and commit phases.

```
/* programs/whirlpool/src/state/prepared_swap.rs */
260 | if self.precondition
261 |     != (PreparedSwapPrecondition {
262 |         slot,
263 |         authority,
264 |         whirlpool,
265 |         whirlpool_state_sequence,
266 |         swap_tick_sequence_len,
267 |         amount,
268 |         sqrt_price_limit,
269 |         amount_specified_is_input,
270 |         a_to_b,
271 |     })
272 | {
273 |     return Err(ErrorCode::PreparedSwapPreconditionMismatch.into());
274 | }
```

The sequence check requires the Whirlpool state to remain unchanged between `prepare_swap` and `commit_swap`, but it does not detect changes to external accounts. The most relevant external accounts are the token mint accounts, which contain configuration such as transfer-fee parameters.

The same-slot requirement prevents the transfer-fee updates made through `set_transfer_fee`, as such changes do not take effect until a later epoch.

```
/* program/src/extension/transfer_fee/processor.rs */
101 | let epoch = Clock::get()?.epoch;
102 | if u64::from(extension.newer_transfer_fee.epoch) <= epoch {
103 |     extension.older_transfer_fee = extension.newer_transfer_fee;
104 | }
105 | // set two epochs ahead to avoid rug pulls at the end of an epoch
106 | let newer_fee_start_epoch = epoch.saturating_add(2);
107 | let transfer_fee = TransferFee {
108 |     epoch: newer_fee_start_epoch.into(),
109 |     transfer_fee_basis_points: transfer_fee_basis_points.into(),
110 |     maximum_fee: maximum_fee.into(),
111 | };
112 | extension.newer_transfer_fee = transfer_fee;
```

However, for the `initialize_transfer_fee_config` instruction of the `TransferFee` extension, a new transfer-fee configuration can take effect immediately in the current epoch.

Since Whirlpool also supports the `MintCloseAuthority` extension, for a mint that has not yet issued any tokens, the authority may be able to close the mint account, reinitialize it, and configure a new `TransferFee` extension.

```
/* program/src/extension/transfer_fee/processor.rs */
025 | fn process_initialize_transfer_fee_config(
026 |     accounts: &[AccountInfo],
027 |     transfer_fee_config_authority: COption<Address>,
028 |     withdraw_withheld_authority: COption<Address>,
029 |     transfer_fee_basis_points: u16,
030 |     maximum_fee: u64,
031 | ) -> ProgramResult {
032 |     let account_info_iter = &mut accounts.iter();
033 |     let mint_account_info = next_account_info(account_info_iter)?;
034 |     check_program_account(mint_account_info.owner)?;
035 |
036 |     let mut mint_data = mint_account_info.data.borrow_mut();
037 |     let mut mint = PodStateWithExtensionsMut::<PodMint>::unpack_uninitialized(&mut mint_data)?;
038 |     let extension = mint.init_extension::<TransferFeeConfig>(true)?;
039 |     extension.transfer_fee_config_authority = transfer_fee_config_authority
040 |         .try_into()
041 |         .map_err(|_| ProgramError::InvalidArgument)?;
042 |     extension.withdraw_withheld_authority = withdraw_withheld_authority
043 |         .try_into()
044 |         .map_err(|_| ProgramError::InvalidArgument)?;
045 |     extension.withheld_amount = 0u64.into();
046 |
047 |     if transfer_fee_basis_points > MAX_FEE_BASIS_POINTS {
048 |         return Err(TokenError::TransferFeeExceedsMaximum.into());
049 |     }
050 |     // To be safe, set newer and older transfer fees to the same thing on init,
051 |     // but only newer will actually be used
052 |     let epoch = Clock::get()?.epoch;
053 |     let transfer_fee = TransferFee {
054 |         epoch: epoch.into(),
055 |         transfer_fee_basis_points: transfer_fee_basis_points.into(),
056 |         maximum_fee: maximum_fee.into(),
057 |     };
058 |     extension.older_transfer_fee = transfer_fee;
059 |     extension.newer_transfer_fee = transfer_fee;
060 |
061 |     Ok(())
062 | }

/* programs/whirlpool/src/util/v2/token.rs */
271 | TokenExtensionType::MintCloseAuthority => {
272 |     if !is_token_badge_initialized {
273 |         return Ok(false);
274 |     }
275 | }
```

In certain edge cases, this means the transfer-fee configuration could change during the time window

between `prepare_swap` and `commit_swap`, potentially introducing risk to the protocol.

For example, suppose a token badge was issued for token A and token A has not yet been minted. A one-sided pool consisting of token A and token B is created with liquidity only on the token B side:

- A user first executes `prepare_swap` to quote a token A to token B swap.
- Before `commit_swap`, the `MintCloseAuthority` closes the `token A` mint account, reinitializes it, and uses `initialize_transfer_fee_config` to configure a new transfer fee for token A.
- When the user later executes `commit_swap`, the actual amount of token A received by the Whirlpool may differ from the amount recorded in state because the transfer-fee configuration in token A has changed.

It is recommended to add checks in `prepare_swap` / `commit_swap` to verify the actual token amount expected to be received by the pool.

Resolution

Resolved by commit 2483c19.

Appendix: Methodology and Scope of Work

Assisted by the Sec3 Scanner developed in-house, the manual audit particularly focused on the following work items:

- Check common security issues.
- Check program logic implementation against available design specifications.
- Check poor coding practices and unsafe behavior.
- The soundness of the economics design and algorithm is out of scope of this work

DISCLAIMER

The instance report ("Report") was prepared pursuant to an agreement between Coderrect Inc. d/b/a Sec3 (the "Company") and Orca Management Company, S.A (the "Client''). This Report solely includes the results of a technical assessment of a specific build and/or version of the Client's code specified in the Report ("Assessed Code") by the Company. The sole purpose of the Report is to provide the Client with the results of the technical assessment of the Assessed Code. The Report does not apply to any other version and/or build of the Assessed Code. Regardless of the contents of the Report, the Report does not (and should not be interpreted to) provide any warranty, representation or covenant that the Assessed Code: (i) is error and/or bug free, (ii) has no security vulnerabilities, and/or (iii) does not infringe any third-party rights. Moreover, the Report is not, and should not be considered, an endorsement by the Company of the Assessed Code and/or of the Client. Finally, the Report should not be considered investment advice or a recommendation to invest in the Assessed Code and/or the Client.

This Report is considered null and void if the Report (or any portion thereof) is altered in any manner.

ABOUT

The Sec3 audit team comprises a group of computer science professors, researchers, and industry veterans with extensive experience in smart contract security, program analysis, testing, and formal verification. We are also building automated security tools that incorporate static analysis, penetration testing, and formal verification.

At Sec3, we identify and eliminate security vulnerabilities through the most rigorous process and aided by the most advanced analysis tools.

For more information, check out our website and follow us on twitter.

