White Whale

Core Pool Contracts

Audit Report

Prepared for White Whale, 10th March 2023

Table of Contents

Introduction

SCV was engaged by White Whale to assist in identifying security threats and vulnerabilities that have the potential to affect their security posture. Additionally, SCV will assist the team in understanding the risks and identifying potential mitigations.

Scope

SCV performed the security assessment on the following codebase:

- Stableswap Pool
    - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/121
    - Code Hash: `8a4147b13237d9959298373ed4fea8f29f074218`
- Stableswap 3pool
    - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
    - Code Hash: `afc8fd1bf997637132dfbf693d1a57b8cb3ea1dd`
- Liquidity pool token token-factory minting
    - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/135
    - Code Hash: `9006e900680ecfaa472ad1904a64b5e9dcc73495`
    - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/141
    - Code Hash: `cbdcf7c90a8910a25bdea8b0b2c4393c7bd668b0`

Remediations were applied by the White Whale team and revised by SCV's up to the following commit hash.

- Stableswap Pool
  - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/121
  - Code Hash: `8a4147b13237d9959298373ed4fea8f29f074218`
- Stableswap 3pool
  - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
  - Code Hash: `5763c91ada1b4854b01e776a04befeffd2f72d1a`
- Liquidity pool token token-factory minting
  - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/135
  - Code Hash: `76870730e933a4d35fea0d914b72af3a4668d573`
  - https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/141
  - Code Hash: `46bb3f000dfcc00193b39d4e4e6961a348fb90ef`

SCV notes that the audit was focused on the code changes implemented by the pull requests mentioned above, rather than the entire codebase. This was done to efficiently allocate resources and prioritize critical areas for review while maintaining security and quality assurance. This approach allowed for a targeted examination of potential issues and timely resolution without delaying the release of new features or updates.

Methodologies

SCV performs a combination of automated and manual security testing based on the scope of testing. The testing performed is based on the extensive experience and knowledge of the auditor to provide the greatest coverage and value to White Whale. Testing includes, but is not limited to, the following:

- Understanding the application and its code base purpose.
- Deploying SCV in-house tooling to automate dependency analysis and static code review.
- Analyze each line of the code base and inspect application security perimeter.
- Review underlying infrastructure technologies and supply chain security posture.

Code Criteria and Test Coverage

This section below represents how *SUFFICIENT* or *NOT SUFFICIENT* each code criteria was during the assessment

Criteria	Status	Notes
Provided Documentation	**SUFFICIENT**	N/A
Code Coverage Test	**SUFFICIENT**	N/A
Code Readability	**SUFFICIENT**	The codebase had good readability and utilized many Rust and CosmWasm best practices.
Code Complexity	**SUFFICIENT**	N/A

Threat Modeling

The goal of threat modeling is to identify and evaluate potential threats to a system or application and to develop strategies to mitigate or manage those threats. Threat modeling is an important part of the software development life cycle, as it helps developers and security professionals proactively identify and address security risks before they can be exploited by attackers.

The main objectives of threat modeling include (not limited to) the following:

- **Identify threats**: The first objective of threat modeling is to identify potential threats that could affect the security posture of the underlying smart contracts or applications. This can include threats from external attackers, internal actors, or even accidental events that could happen.
- **Evaluate risks:** Once potential threats have been identified, the next objective is to evaluate the risks associated with each threat. This involves assessing the likelihood of each threat occurring and the potential impact it could have overall.
- **Mitigation strategies:** After identifying potential threats and evaluating the associated risks, the next objective is to develop strategies to mitigate or reduce the impact of threats. This can include implementing technical controls, such as access controls or further security measures around developing policies and procedures to reduce the likelihood or impact of a threat.
- **Communicate findings:** The final objective of threat modeling is to communicate the findings and recommendations to relevant stakeholders, such as developers, security teams, and management. This helps ensure that everyone involved in the development and maintenance understands the potential risks and the best strategies for addressing them.

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	Modifying amp with a large difference allows attackers to steal funds	**Critical**	**Resolved**
2	Creating a token factory denom will always fail due to insufficient denom creation fee	**Severe**	**Resolved**
3	Burn function might fail for token factory denoms	**Medium**	**Resolved**
4	Instantiating amp with zero value causes `Stableswap` to fail	**Medium**	**Resolved**
5	`CollectProtocolFees` does not handle the third asset for the trio contract	**Medium**	**Resolved**
6	Migrating contracts from version lower or equal to `1.0.4` cause executions to fail	**Medium**	**Acknowledged**
7	Slippage tolerance is not implemented for the trio contract's third asset	**Low**	**Resolved**
8	First depositor is unable to specify slippage tolerance when providing liquidity	**Low**	**Resolved**
9	Incorrect comment in the swap function	**Low**	**Resolved**
10	Trio contract's `ask_asset.amount` during the swap function is ignored	**Informational**	**Resolved**
11	Incorrect documentation for `ProtocolFeesResponse` struct	**Informational**	**Resolved**
12	Migrating trio contracts requires state migration	**Informational**	**Resolved**
13	Max spread should be validated below 100%	**Informational**	**Resolved**
14	`MigratePair` uses pair code id by default	**Informational**	**Resolved**
15	Contracts should use two-step ownership transfer	**Informational**	**Acknowledged**

16	Incorrect error name	**Informational**	**Resolved**

Detailed Vulnerabilities

1 – Modifying amp with a large difference allows attackers to steal funds

---

Risk Impact: Critical - **Status**: Resolved

Affected PR and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/commands.rs:181`
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/helpers.rs:21`
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/helpers.rs:74`

Description

When performing a swap in the trio pool, the `amp_factor` is used to provide tight spreads for stable pairs. Normally, the factor should gradually increase or decrease based on the configured timeline and `target_amp_factor`.

However, the `Stableswap` invariant calculator is instantiated with the `start_ramp_ts` and `stop_ramp_ts` as zero, causing the amp factor to be reflected instantly based on the configured value. By frontrunning the `UpdateConfig` message, an attacker can sell their tokens before the `amp_factor` is updated to a lower value, then repurchase them to gain profit. The bigger the amp factor value was decreased, the higher profitability for the attacker.

This issue is similar to the Curve Vulnerability Report by Peter Zeitz. Please see the test case in the following link to reproduce the issue.

Recommendations

Consider fully implementing the `Stableswap` invariant calculator with expected `start_ramp_ts` and `stop_ramp_ts` so the amp factor will change gradually instead of a big discrete step. As the amp factor changes in a small step, the fees and gas cost will decrease the profitability for the attacker to perform such an attack.

2 – Creating a token factory denom will always fail due to insufficient denom creation fee

Risk Impact: Severe - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/135
- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/141
- `contracts/liquidity_hub/pool-network/terraswap_pair/src/contract.rs:109-114`

Description

In most chains, a small amount of denom creation fee is required when creating a token factory denom to prevent spam. For example, the cost of creating a native token in the Juno chain is 1 JUNO.

Since the factory contract does not include native funds, as seen in `contracts/liquidity_hub/pool-network/terraswap_factory/src/commands.rs:147`, the pair contract instantiation will fail due to insufficient funds.

Recommendations

Consider including the denom creation fee if the `token_factory_lp` boolean is set to true. Optionally, consider querying the token creation fee to ensure the amount provided is sufficient.

3 – Burn function might fail for token factory denoms

Risk Impact: Medium - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `packages/terraswap/src/asset.rs:76-78`

Description

If the swap computation's burn fee amount is not zero, `BankMsg::Burn` would be executed to burn the native tokens. Suppose a scenario where a pair is created with token factory denoms. Since only the token admin is allowed to burn the funds, the burn function will fail because the pair contract is not the admin of the factory token.

Recommendations

Consider preventing token factory denoms from being created as pairs.

4 – Instantiating amp with zero value causes `Stableswap` to fail

Risk Impact: Medium - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/121
- `contracts/liquidity_hub/pool-network/terraswap_pair/src/contract.rs:48`
- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/contract.rs:75`
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/commands.rs:543`

Description

If the pair type is `StableSwap`, the `terraswap_pair` contract does not validate the amp to ensure the value is not zero. This would cause all swaps to fail due to computation errors, forcing users to withdraw liquidity.

Since the factory contract does not allow duplication of pool creations, the contract owner will be forced to recreate the pair or perform a state migration using the `MigratePair` message. This affects both stableswap and trio contracts.

Recommendations

Consider ensuring the amp factor amount is higher than `MIN_AMP` and lower than `MAX_AMP` as defined here.

5 – `CollectProtocolFees` does not handle the third asset for the trio contract

---

Risk Impact: Medium - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/commands.rs:562-574`

Description

The `collect_protocol_fees` function resets the amount of the first and second assets when the fees are sent to the fee collector address. As the trio contract holds three assets, the last asset is ignored.

This would cause the protocol fee for the last asset to be uncollected and cause the `ALL_TIME_COLLECTED_PROTOCOL_FEES` storage state to record an incorrect amount.

Recommendations

We recommend updating the `collect_protocol_fees` function to include all three protocol fee assets for the trio correctly.

6 – Migrating contracts from version lower or equal to `1.0.4` cause executions to fail

---

Risk Impact: Medium - **Status**: Acknowledged

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/terraswap_pair/src/contract.rs:247-251`

Description

The `migrate` function for the pair contract will perform migration if the contract version is lower or equal to `1.0.4` or `1.1.0` using an if else statement. If the contract version is `1.1.0`, the contract migration will succeed without any issues.

However, if the contract version is lower or equal to `1.0.4`, the `migrate_to_v110` function will be executed, and the contract version will be updated to the latest version. This is problematic because the `migrate_to_v120` function will not be executed to implement the required burn structs, causing the contract to be unusable.

Furthermore, the contract admin cannot perform another migration due to the version check in `line 240`. Ultimately, the contract admin must manually upload a new code identifier and perform migration to recover the situation.

Recommendations

Consider executing the `migrate_to_v120` function automatically when migrating from a version lower or equal to `1.0.4`.

Revision Notes

The client has acknowledged this finding and stated that there are no contracts live on a version lower or equal to `1.0.4` that can fall into that case.

7 – Slippage tolerance is not implemented for the trio contract's third asset

Risk Impact: Low - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/helpers.rs:214-221`

Description

When asserting slippage tolerance, the third asset's deposit and pool are not included to ensure max slippage is not exceeded. This would cause slippage tolerance to be unenforced for the third asset in the trio contract.

Recommendations

Consider resolving the TODO comment and implementing slippage tolerance for the third asset.

8 – First depositor is unable to specify slippage tolerance when providing liquidity

---

Risk Impact: Low - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/helpers.rs:216`

Description

When asserting slippage tolerance, the total asset amount in the pool is used as the denominator. Since there are no funds inside the pool yet, the amount would be zero.

This means that the first depositor cannot specify any slippage tolerance, or else a division by zero error would occur, preventing them from providing liquidity.

Recommendations

Consider ignoring slippage checks when the first depositor is providing liquidity.

9 – Incorrect comment in the swap function

Risk Impact: Informational - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/121
- `contracts/liquidity_hub/pool-network/terraswap_pair/src/commands.rs:299`

Description

The comment mentions, "*The user must IncreaseAllowance on the token if it is a cw20 token they want to swap*". This is incorrect, as users will call the `Cw20HookMsg::Swap` message to swap CW20 tokens instead of increasing allowances.

Recommendations

Consider modifying the comment according to the intended functionality.

10 – Trio contract's `ask_asset.amount` during the swap function is ignored

Risk Impact: Informational - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/commands.rs:314`

Description

The `ask_asset` variable type is `Asset` which consists of the asset information and amount. Supplying any amount will be pointless because the `ask_asset.amount` is not used within the contract.

Recommendations

Consider refactoring the variable type of `ask_asset` into `AssetInfo` instead so the amount does not need to be provided.

11 – Incorrect documentation for `ProtocolFeesResponse` struct

Risk Impact: Informational - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `packages/terraswap/src/trio.rs:168`

Description

The documentation for the `ProtocolFeesResponse` struct is the same with the `ReverseSimulationResponse` struct, which is incorrect.

Recommendations

Consider amending the documentation accordingly.

12 – Migrating trio contracts requires state migration

Risk Impact: Informational - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/contract.rs:268-272`

Description

State migrations are executed in lines 268 to 272 when performing a contract migration. This is unneeded as the trio contract is newly implemented, so there are no previous versions.

Recommendations

Consider removing unnecessary state migrations for the trio contract.

13 – Max spread should be validated below 100%

Risk Impact: Informational - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/helpers.rs:167`

Description

The max spread value provided by the sender is not validated to be below 100%. As the max spread percentage calculated is 100%, supplying a value higher than 100% is essentially the same as performing a swap without any spread limit.

Recommendations

Consider ensuring the provided max spread value is lower than 100%.

14 – `MigratePair` uses pair code id by default

Risk Impact: Informational - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/terraswap_factory/src/commands.rs:383`

Description

The `execute_migrate_pair` function will automatically set the `code_id` value to `pair_code_id` is the value provided is `None`. This might cause the contract owner to migrate a trio contract pair without providing a specific `code_id`. As a result, the trio contract would be migrated into the pair contract's code, which is incorrect.

Recommendations

Consider forcing the contract owner to provide a `code_id` instead of automatically unwrapping into the pair contract's code id.

15 – Contracts should use two-step ownership transfer

Risk Impact: Low - **Status**: Acknowledged

Affected PR and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/131
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/commands.rs:181`
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/helpers.rs:21`
- `contracts/liquidity_hub/pool-network/stableswap_3pool/src/helpers.rs:74`

Description

The current ownership transfer for each of the contracts is executed in one step, which imposes a risk that if the new owner is incorrect, then the admin privileges of the contract are effectively transferred and lost. A two-step ownership transfer is best practice because the new admin must accept ownership before the transfer and config changes occur.

Recommendations

We recommend implementing a two-step ownership transfer where the current owner proposes a new owner address, and then that new owner address must call the contract to accept ownership within a finite time frame.

16 – Incorrect error name

Risk Impact: Informational - **Status**: Resolved

Affected PRs and code lines

- https://github.com/White-Whale-Defi-Platform/white-whale-core/pull/141
- `contracts/liquidity_hub/pool-network/terraswap_factory/src/error.rs:35`

Description

The error in `contracts/liquidity_hub/pool-network/terraswap_factory/src/error.rs:35` called `UnExistingTrio` could be changed to "*NonExistantTrio*" for more accuracy.

Recommendations

We recommend updating the error to "*NonExistantTrio*".

Document control

Version	Date	Approved by	Changes
0.1	13/03/2023	Vinicius Marino	Document Pre-Release
0.2	08/04/2023	SCV Team	Remediation Revisions
1.0	10/04/2023	Vinicius Marino	Document Release

Appendices

A. Appendix – Risk assessment methodology

A qualitative risk assessment is performed on each vulnerability to determine the impact and likelihood of each.

Risk rate will be calculated on a scale. As per criteria Likelihood vs Impact table below:

	Rare	Unlikely	Possible	Likely
Critical	**Medium**	**Severe**	**Critical**	**Critical**
Severe	**Low**	**Medium**	**Severe**	**Severe**
Moderate	**Low**	**Medium**	**Medium**	**Severe**
Low	**Low**	**Low**	**Low**	**Medium**
Informational	**Informational**	**Informational**	**Informational**	**Informational**

LIKELIHOOD

- Likely: likely a security incident will occur;
- Possible: It is possible a security incident can occur;
- Unlikely: Low probability a security incident will occur;
- Rare: In rare situations, a security incident can occur;

IMPACT

- Critical: May cause a significant and critical impact;
- Severe: May cause a severe impact;
- Moderate: May cause a moderated impact;
- Low: May cause low or none impact;
- Informational: May cause very low impact or none.

B. Appendix – Report Disclaimer

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts SCV-Security to perform a security review. The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The content of this audit report is provided “as is”, without representations and warranties of any kind, and SCV-Security disclaims any liability for damage arising out of, or in connection with, this audit report.