January 11, 2023

Minswap - Stableswap

TxPipe Shop

Contents

1 - Summary

This report provides a comprehensive audit of the Stableswap platform, a decentralized, smart contract-based exchange system on the Cardano network. The audit, conducted without warranties or guarantees of the code's security or quality, focused on several potential vulnerabilities. The vulnerabilities include scenarios where a malicious actor might exploit the validator to disrupt the platform or its users. It's important to note that this report only covers identified issues, and we do not claim to have detected all potential vulnerabilities.

1.1 - Overview

Stableswap operates on the Cardano blockchain, enabling low-slippage trades between stablecoins. Its main distinguishing feature is the use of the Curve algorithm which improves trades between assets of similar prices. The platform leverages validators to facilitate trades, where each validator action is a distinct function in the system. Key components include:

- **Curve's Stableswap Algorithm**: Stableswap leverages the Curve's Stableswap algorithm for facilitating stablecoin trades with minimal slippage and transaction fees. The algorithm's crucial metric is the Amplification Coefficient.

- **Batching Architecture**: To address concurrency on Cardano, Stableswap employs a batching architecture. Each user action creates an datum in the Order validator, and a Batcher applies them into a Pool. Only wallets containing a License Token, acting as a valid Batcher, can trigger a batching transaction.

- **Stableswap Validators**: The system consists of three validators:

  - **Order Validator**: It accumulates User actions, that then await application into a Pool.

  - **Pool Validator**: Also known as the Liquidity Pool, it holds all the assets for trading. This integral part of the system ensures that Orders are processed correctly and that Liquidity Providers' funds are secure.

  - **Liquidity Validator**: This Minting Policy's tokens represent the share of Liquidity Providers in the Liquidity Pool.

- **Liquidity Tokens**: These tokens denote the shares of the Liquidity Providers in the Pool, with each Pool having distinct LP tokens.

- **User Action**: The platform supports various User actions like exchanging assets, depositing assets in the Pool, withdrawing assets from the Pool, withdrawing a specific asset from the Pool, etc.

- **Admin Control**: The Minswap team or the Admin has permissions to update a Pool's Amplification Coefficient, withdraw the Admin fee, and change the Pool's stake address.

1.2 - Process

Our audit process involved a thorough examination of the Stableswap validators. Areas vulnerable to potential security threats were closely scrutinized, including those where attackers could exploit the validator's functions to disrupt the platform and its users. This included evaluating potential risks such as unauthorized asset addition, hidden market creation, and disruptions to interoperability with other Plutus scripts. This also included the common vulnerabilities such as double satisfaction and minting policy vulnerabilities.

The audit took place over a period of several weeks, and it involved the evaluation of the platform's mathematical model to verify that the implemented equations matched those of the curve algorithm. However, it should be noted that the audit did not include a verification of the curve algorithm itself.

Findings and feedback from the audit were communicated regularly to the Minswap team through Discord. Diagrams illustrating the necessary transaction structure for proper interaction with Stableswap are attached as part of this report. The Minswap team addressed these issues in an efficient and timely manner, enhancing the overall security of the platform.

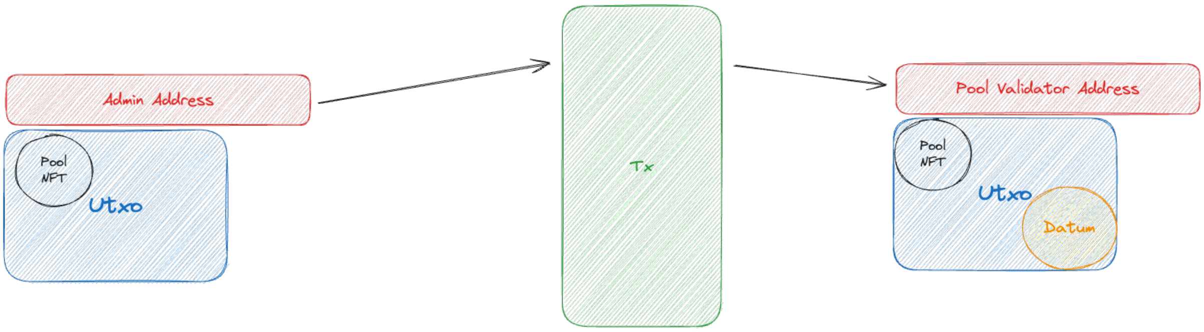
1.3 - Transactions

Below are a list of valid and invalid transactions as diagrams that can be build to interact with Stableswap.

1.3.1 - Create Pool

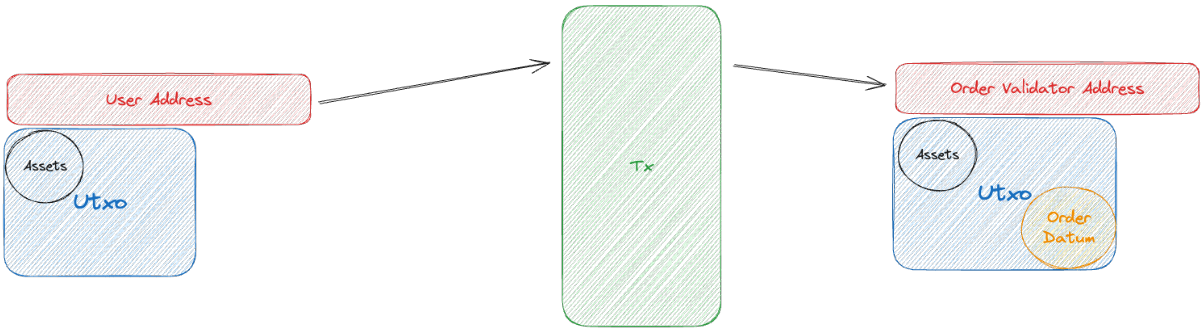
A Pool is a parameterized Pool validator that is created by the Admin. Pools that users can interact with on Minswap's interface are created and whitelisted by the Minswap team. This transaction only creates a Pool UTxO which contains the Pool NFT and some initial datum. The initial datum contains:

- **balances**: an array having N elements representing asset balances, each balance is initially 0
- **total_liquidity**: 0
- **amp**: initial Amplication Coefficient



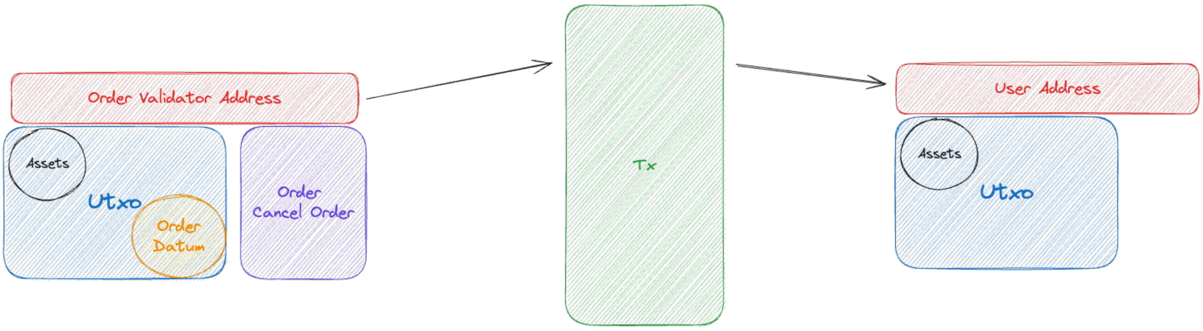
1.3.2 - Create Order

A Create Order transaction transfers user funds into a UTxO at an Order validator which was parameterized by a Pool validator. This transaction does not require the execution of a validator and simply stores a datum that contains information about the type of Order the user made. The assets sent are different depending on the type of Order but they all contain output ADA and the batcher fee. Exchange Orders will contain the amount of an asset that the user wants to exchange. Deposit Orders will contain the amount of an asset that the user wants to deposit into the Pool in exchange for LP tokens. Withdraw Orders will contain the amount of LP tokens that the user wants to burn in exchange for the underlying asset in the Pool. WithdrawImbalance and WithdrawOneCoin Orders contain the same kind of asset as Withdraw.



1.3.3 - Cancel Order

A Cancel Order transaction transfers user funds from an Order validator back to the user's wallet. This transaction requires the execution of a validator because it needs to spend from an Order validator. It is the reverse of a Create Order transaction.

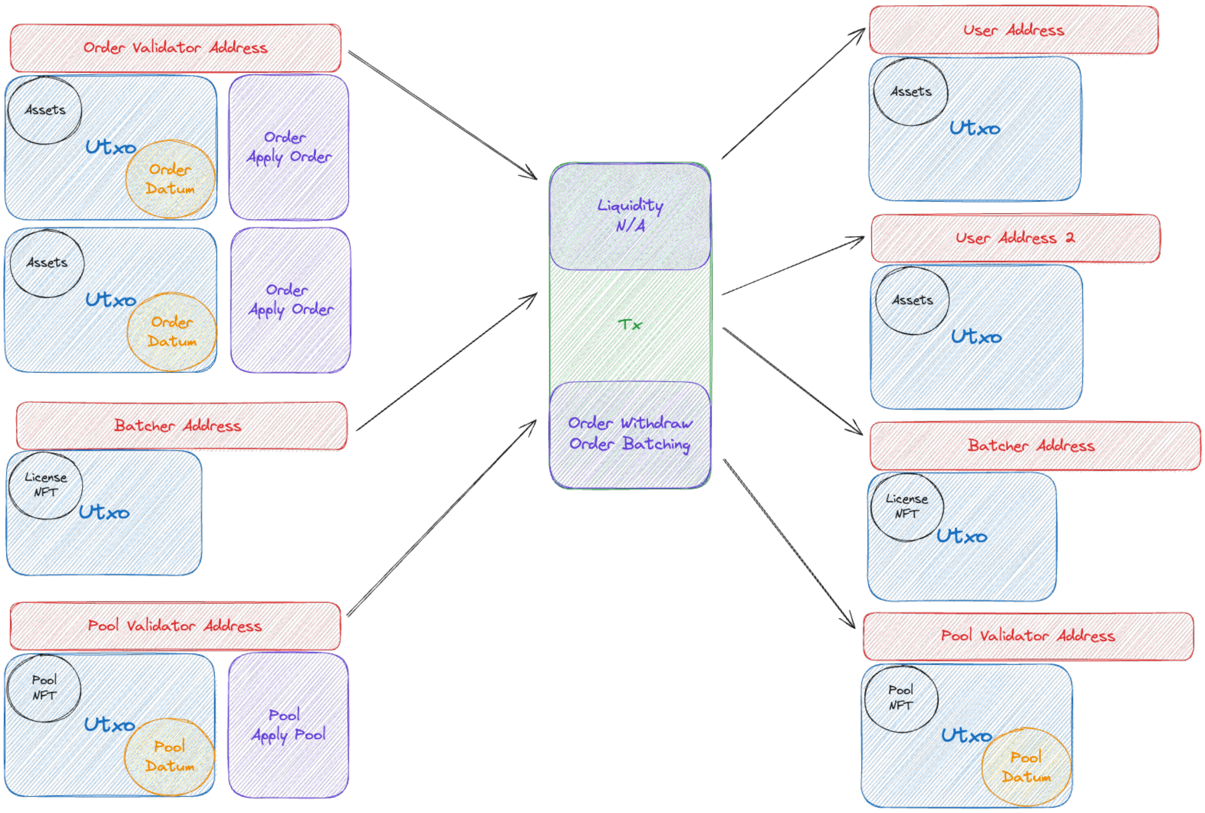


1.3.3.1 - Expected Failure Scenarios

- The transaction should fail if it is **not** signed by the owner that is specified in the Order datum.

1.3.4 - Batching

Batching transactions are a complex, yet crucial, part of the system, requiring a Pool UTxO, Batcher UTxOs (which must include a Batcher License Token), and Order UTxOs. The Batcher's role is to locate all pending Orders on the blockchain, identify the matching Pool UTxO, and systematically process these Orders. The transaction outcomes for exchanges, deposits, and various types of withdrawals are calculated with respect to the user's funds, the output ADA, and the assets involved. The Batcher also manages fees, retaining a portion to cover transaction costs, and keeping the remaining amount in the Batcher wallet. The third LP validator may get executed if any of the Orders are deposits or withdraws which requires the minting and burning of LP tokens. The Order Withdraw validator is used by the Order validator inputs to quickly and efficiently validate the existence of the pool input.



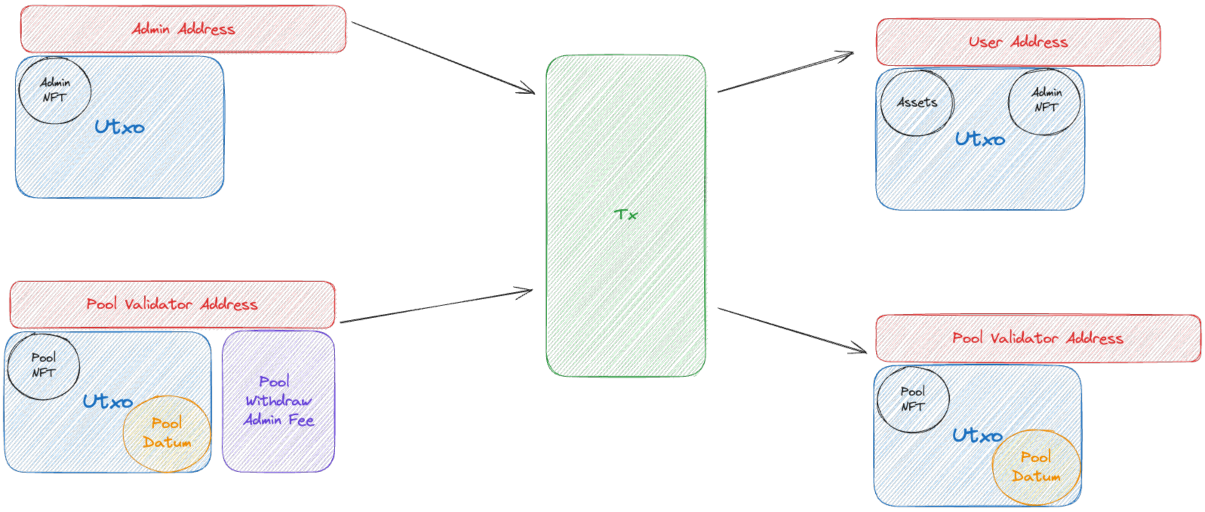
1.3.4.1 - Expected Failure Scenarios

- Batcher License Token is **not** provided in the batcher inputs
- Batcher License Token is expired
- Batcher License Token has a validity period greater than the maximum validity period
- Pool input does **not** contain a Pool NFT
- Pool input does **not** contain a Pool datum
- More than one Pool input
- Pool output does **not** contain a Pool NFT
- Pool output does **not** contain a Pool datum
- Pool output asset amounts do **not** match the onchain calculated amounts
- Pool output asset amounts is **not** greater than or equal to datum balance
- Pool output datum balance does **not** match the onchain calculated amounts
- Pool datum balance contains any assets other than the "Pool assets"
- More than one Pool output

- Pool input and output do **not** have the same address
- Pool output contains any assets other than the “Pool assets”, ADA, and Pool NFT
- Order inputs do **not** have an Order datum
- Order inputs are **not** processed in the batcher specified Order
- Any Order input is processed more than once or not at all
- Order inputs have a different Order validator hash
- Order inputs and outputs lengths don’t match
- Order outputs go to a different address than the datum owner address
- Order outputs contain any assets other than the “Pool assets”, ADA, and the Pool LP tokens
- Order outputs asset amounts do **not** surpass the minimum specified in the inputs (slippage surpassed)
- Order outputs asset amounts do **not** match the onchain calculated amounts
- Assets other than POOL LP tokens are minted
- Contains inputs **not** from the Order validator, Pool validator, or Batcher address
- Does **not** have at least one input from the Order validator, Pool validator, and Batcher address each
- Order inputs are spent without the Withdraw validator being present
- Redeemer lists contain only unique indices have a length of at least one
- The Withdraw validator is included without a pool being present.

1.3.5 - Withdraw Admin Fee

The withdrawal of the Admin Fee necessitates the Admin’s License Token. This transaction utilizes the Admin’s License Token, computes the accumulated Admin Fee amounts, and subsequently transfers these amounts to the Admin Wallet.



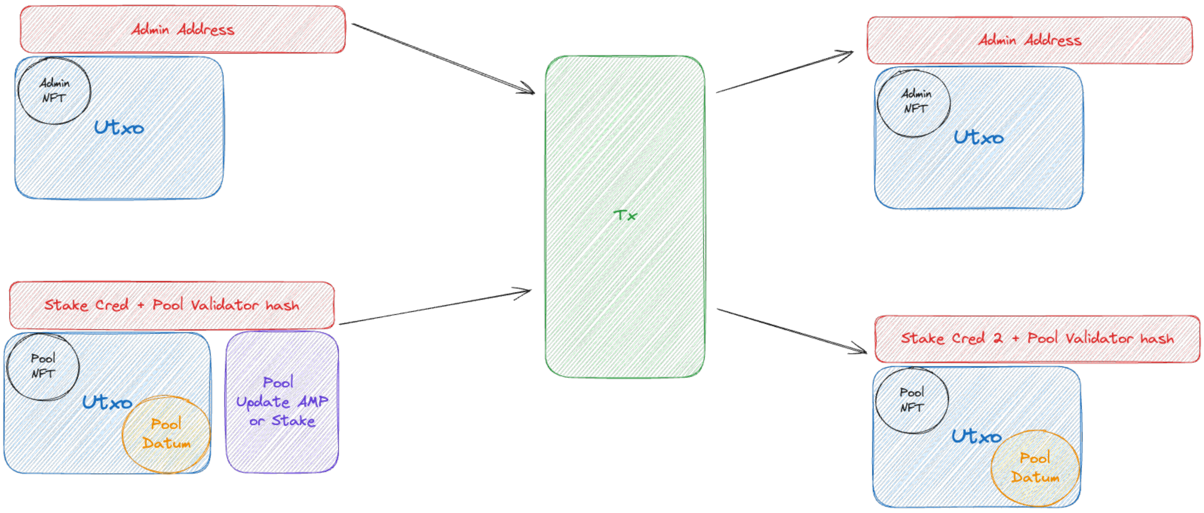
1.3.5.1 - Expected Failure Scenarios

- Admin NFT is **not** provided in the inputs
- Pool input does **not** contain a Pool NFT
- Pool output does **not** contain a Pool NFT
- Pool input does **not** contain a Pool datum
- Pool output does **not** contain a Pool datum
- Pool output contains any assets other than the “Pool assets”, ADA, and Pool NFT
- More than one Pool input
- Mints any assets

- Pool datum is changed
- Pool output asset amounts do not match the Pool datum balances
- More than one script is present
- Pool output goes to a different address than the Pool input

1.3.6 - Update Amplification Coefficient or Update Pool's Stake Credential

This transaction can perform two key functions which is either updating the Amplification Coefficient or modifying the Pool's Stake Credential. The process involves spending the Pool UTxO, adjusting the Amp configuration in the Pool's Datum, and transporting the Datum and Value within the Pool UTxO to a new UTxO, which retains the same payment credential while introducing a new stake credential.



1.3.6.1 - Expected Failure Scenarios

- Admin NFT is **not** provided in the inputs
- Pool input does **not** contain a Pool NFT
- Pool output does **not** contain a Pool NFT
- Pool input does **not** contain a Pool datum
- Pool output does **not** contain a Pool datum
- Pool output contains any assets other than the "Pool assets", ADA, and Pool NFT
- More than one Pool input
- Mints any assets
- Pool datum except amp is changed
- Pool output asset amounts do not match the Pool input asset amounts
- More than one script is present
- Pool output goes to a different Payment Credential than the Pool input

1.4 - Files Audited

Below is a list of all files audited in this report, any files **not** listed here were **not** audited. The final state of the files for the purposes of this report is considered to be commit `d0927de88db4f0bcbe08005dc8e2400290112c39`.

Filename
validators/lp_minting_policy.ak
validators/order_validator.ak
validators/pool_validator.ak
lib/stableswap/pool_utils.ak
lib/stableswap/types.ak
lib/stableswap/utils.ak

2 - Findings

ID	Title	Severity	Status
MS-001	Batcher could game ‘input_indexes’ to drain the Pool and steal User funds	Critical	Resolved
MS-201	Other (order-like) scripts besides the parameterized Order validator can be used	Minor	Resolved
MS-202	/ operator rounds down to negative infinity	Minor	Resolved
MS-203	Output value checks for ‘apply_orders’ should check value length.	Minor	Resolved
MS-301	Checking list length of 1 could be replaced with `expect [_]`	Info	Resolved
MS-302	Comparing ‘user_inputs’ and ‘user_outputs’ lengths done redundantly	Info	Resolved
MS-303	Useless rebinding of ‘multiples’ in ‘calculate_exchange’	Info	Resolved
MS-304	Redundant boolean comparison ‘byte < zero_ascii_code’ in ‘do_bytearray_to_int’	Info	Resolved
MS-305	Inefficiently checking for presence of other scripts	Info	Resolved
MS-306	Optimize list.at(index) function	Info	Resolved
MS-307	Use list.any instead of list.filter followed by list.length > 0	Info	Resolved

3 - MS-001 Batcher could game ‘input_indexes’ to drain the Pool and steal User funds

Category	Commit	Severity	Status
Logical	70295e2cc151cea75934c3356fb96b62816d23bc	Critical	Resolved

3.1 - Description

The `input_indexes` value is determined by the batcher. The batcher could game this to drain the Pool or steal from User Orders by either creating a `input_indexes` list that is smaller than the `user_inputs` length or by creating a `input_indexes` with non unique indices. For example the batcher could use a list of [3,3,3,3,3,3,3,3] or a list of [3] for `input_indexes`. This would allow the batcher to apply the same Order multiple times or not apply any of the User Orders.

3.2 - Recommendation

`input_indexes` should be checked to be equal length to `user_inputs` and that all indices in the list are unique.

3.3 - Resolution

Resolved in commit `c6ce15beeb556e8b80861236763131c471bcac96`

4 - MS-201 Other (order-like) scripts besides the parameterized Order validator can be used

Category	Commit	Severity	Status
External	0f22b506370716f1203f54e37f40816bad8f52c9	Minor	Resolved

4.1 - Description

Other (order-like) scripts besides the parameterized Order validator can be used by the Batcher as long as the datum structure matches.

This allows for hidden Orders that malicious Batchers can give priority to that no other Batcher or Minswap team member would be aware of. Essentially it creates a hidden market where certain batchers can be bribed to give the best Order flow to Orders at a different Order validator address.

4.2 - Recommendation

Put Order validator hash in Pool validator datum to compare when gathering user inputs during `ApplyOrder`

4.3 - Resolution

Resolved in commit `09696ab3fc082c3cc1a23147e6ac93913b6cc896`

5 - MS-202 / operator rounds down to negative infinity

Category	Commit	Severity	Status
Concern	0f22b506370716f1203f54e37f40816bad8f52c9	Minor	Resolved

5.1 - Description

Aiken’s division operator truncates toward negative infinity not 0, cases where you do division followed by absolute value are affected by the value rounding toward negative infinity.

5.2 - Recommendation

If the value needs to truncate towards 0 then use the quotient builtin function.

5.3 - Resolution

No use cases of negative division were found in the codebase.

6 - MS-203 Output value checks for ‘apply_orders’ should check value length.

Category	Commit	Severity	Status
Concern	0f22b506370716f1203f54e37f40816bad8f52c9	Minor	Resolved

6.1 - Description

Output value ada and the traded assets have their amounts checked in the output, but there is not a further check to prevent other assets from being added to a Order’s designated output. This could be harmful to scripts that expect an exact amount of unique assets present in an input. Basically harming interoperability with other Plutus scripts.

6.2 - Recommendation

Check that the length of output assets is exactly ada and the assets received from the Order execution.

6.3 - Resolution

Resolved in commit `09696ab3fc082c3cc1a23147e6ac93913b6cc896`

7 - MS-301 Checking list length of 1 could be replaced with `expect [_]`

Category	Commit	Severity	Status
Style	0f22b506370716f1203f54e37f40816bad8f52c9	Info	Resolved

7.1 - Description

Aiken has multiple ways to check the amount of items in a list. One alternative to length check is using `expect [_]` to fail if the list does not match the expect pattern.

7.2 - Recommendation

If the program is expected to fail on a length of anything except 1, one alternative syntax is to expect on there being only one item in the list.

7.3 - Resolution

Resolved in commit `09696ab3fc082c3cc1a23147e6ac93913b6cc896`

8 - MS-302 Comparing 'user_inputs' and 'user_outputs' lengths done redundantly

Category	Commit	Severity	Status
Efficiency	0f22b506370716f1203f54e37f40816bad8f52c9	Info	Resolved

8.1 - Description

In `apply_orders` there is a length check that checks that the User Order inputs list is equal in length to the User Order outputs. Getting a list's length is O(n) complexity, `apply_orders` is recursive, and this check only needs to be done once.

8.2 - Recommendation

The list length of user inputs and outputs should happen before calling `apply_orders`. This would ensure the list lengths are equal and then the `apply_orders` can `expect [output, ..outputs] = ...` after checking that there is an input.

8.3 - Resolution

Resolved in commit `09696ab3fc082c3cc1a23147e6ac93913b6cc896`

9 - MS-303 Useless rebinding of 'multiples' in 'calculate_exchange'

Category	Commit	Severity	Status
Style	0f22b506370716f1203f54e37f40816bad8f52c9	Info	Resolved

9.1 - Description

There is a redundant line that says `let multiples = multiples` in `calculate_exchange`.

9.2 - Recommendation

Remove `let multiples = multiples`, the Aiken compiler would automatically optimize away the reassignment, but there is no need for this code to exist.

9.3 - Resolution

Resolved in commit `9dfb392d2c4dfc99e1d5cfb4e85269eefe75a3dc`

10 - MS-304 Redundant boolean comparison ‘byte < zero_ascii_code’ in ‘do_bytearray_to_int’

Category	Commit	Severity	Status
Redundancy	0f22b506370716f1203f54e37f40816bad8f52c9	Info	Resolved

10.1 - Description

In `do_bytearray_to_int` there is a redundant check that does `byte < zero_ascii_code || byte < zero_ascii_code || ...`

10.2 - Recommendation

Remove the redundant `byte < zero_ascii_code`

10.3 - Resolution

Resolved in commit `9dfb392d2c4dfc99e1d5cfb4e85269eefe75a3dc`

11 - MS-305 Inefficiently checking for presence of other scripts

Category	Commit	Severity	Status
Efficiency	d16065d19a670381d7f08d53da291f81061e2928	Info	Resolved

11.1 - Description

In `validate_withdraw_admin_fee` and `validate_update_amp_or_update_stake_credential` there is a check to see if other scripts are present in the transaction. This can be done more efficiently by using the redeemer count.

11.2 - Recommendation

Check that the length of `redeemers` in the `ScriptContext` is `1`

11.3 - Resolution

Resolved in commit `70295e2cc151cea75934c3356fb96b62816d23bc`

12 - MS-306 Optimize list.at(index) function

Category	Commit	Severity	Status
Efficiency	d16065d19a670381d7f08d53-da291f81061e2928	Info	Resolved

12.1 - Description

For cases where you want to get an item at the list index and fail the validator otherwise, list.at(index) is slower due to handling both the success and failure cases. Instead you can use `list_at_index` which is optimized for the success case and fails the validator if the list is going out of bounds. Furthermore `list_at_index` is optimized for skipping indices to quickly get to the desired index. The dependency `list_at_index_step` is used for getting to an item quickly in smaller lists.

12.2 - Recommendation

Replace usage of list.at(index) with the usage of list_at_index(index) if the list is often > 5 items. In the case of less items, use list_at_index_step(index)

12.3 - Resolution

Resolved in commit `0fe13a2a1028bd96316a57140d6b8fd090df2e31`

13 - MS-307 Use list.any instead of list.filter followed by list.length > 0

Category	Commit	Severity	Status
Efficiency	3313526d5cdf-f1792ec2979cbf6cf878c1a57187	Info	Resolved

13.1 - Description

Using list.filter goes through the entire list. This can be avoided in cases where you just need to find the existence of a single item in the list.

13.2 - Recommendation

Use list.any in places where you use list.filter followed by list.length > 0

13.3 - Resolution

Resolved in commit `dcc75cf45e49f2195df37b14192d65c85137c3da`

14 - Appendix

14.1 - Disclaimer

This report is governed by the terms in the agreement between TxPipe (**TXPIPE**) and Minswap Labs (**CLIENT**). This report cannot be shared, referred to, altered, or relied upon by any third party without TXPIP's written consent. This report does not endorse or disapprove any specific project, team, code, technology, asset or similar. It provides no warranty or guarantee about the quality or nature of the technology analyzed.

TXPIPE DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, related to this report, its content, and the related services and products. This report is provided as-is. TxPipe does not take responsibility for any product or service advertised or offered by Client or any third party. **TXPIPE IS NOT RESPONSIBLE FOR MONITORING ANY TRANSACTION BETWEEN YOU AND CLIENT AND/OR ANY THIRD-PARTY PROVIDERS OF PRODUCTS OR SERVICES.**

This report should not be used for making investment or involvement decisions with any project, services or assets. This report provides general information and is not a form of financial, investment, tax, legal, regulatory, or other advice.

TxPipe created this report as an informational review of the due diligence performed on the Client's smart contract. This report provides no guarantee on the security or operation of the smart contract on deployment or post-deployment. **TXPIPE HAS NO DUTY TO MONITOR CLIENT'S OPERATION OF THE PROJECT AND UPDATE THE REPORT ACCORDINGLY.**

The information in this report may not cover all vulnerabilities. This report represents an extensive assessment process intended to help increase the quality of the Client's code. However, blockchain technology and cryptographic assets present a high level of ongoing risk, including unknown risks and flaws.

TxPipe recommends multiple independent audits, a public bug bounty program, and continuous security auditing and monitoring. Errors in the manual review process are possible, and TxPipe advises seeking multiple independent opinions on critical claims. **TXPIPE BELIEVES EACH COMPANY AND INDIVIDUAL IS RESPONSIBLE FOR THEIR OWN DUE DILIGENCE AND CONTINUOUS SECURITY.**

14.2 - Issue Guide

14.2.1 - Severity

Severity	Description
Critical	Critical issues highlight exploits, bugs, loss of funds, or other vulnerabilities that prevent the dApp from working as intended. These issues have no workaround.
Major	Major issues highlight exploits, bugs, or other vulnerabilities that cause unexpected transaction failures or may be used to trick general users of the dApp. dApps with Major issues may still be functional.
Minor	Minor issues highlight edge cases where a user can purposefully use the dApp in a non-incentivized way and often lead to a disadvantage for the user.
Info	Info are not issues. These are just pieces of information that are beneficial to the dApp creator. These are not necessarily acted on or have a resolution, they are logged for the completeness of the audit.

14.2.2 - Status

Status	Description
Resolved	Issues that have been **fixed** by the **project** team.
Acknowledged	Issues that have been **acknowledged** or **partially fixed** by the **project** team. Projects can decide to not **fix** issues for whatever reason.
Identified	Issues that have been **identified** by the **audit** team. These are waiting for a response from the **project** team.

14.3 - Revisions

This report was created using a git based workflow. All changes are tracked in a github repo and the report is produced using typst. The report source is available here. All versions with downloadable PDFs can be found on the releases page.

14.4 - About Us

TxPipe is a blockchain technology company responsible for many projects that are now a critical part of the Cardano ecosystem. Our team built Oura, Scrolls, Pallas, Demeter, and we're the original home of Aiken. We're passionate about making tools that make it easier to build on Cardano. We believe that blockchain adoption can be accelerated by improving developer experience. We develop blockchain tools, leveraging the open-source community and its methodologies.

14.4.1 - Links

- Website
- Email
- Twitter