ClarityAlliance

BITFLOW STABLESWAP SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, ABA, MARCHEV

SEPTEMBER 25TH, 2024

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
 5.1. Impact 4
 5.2. Likelihood 5
 5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
 [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
 [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
 [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
 [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
 [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
 [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
 [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
 [L-03] Pools with an Invalid Fee Structure Can Be Created 20
 [L-04] Pool Validity Check Not Reached for Invalid Pools 21
 [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
 [L-06] Missing Deadline On Swaps 23
 [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
 [L-08] Avoid Using tx-sender for Caller Identification 25
 [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
 [L-10] Lack of Proper Error Handling on Convergence Failure 27
 [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
 [QA-01] Protocol Fee Irregularities 30
 [QA-02] Simplification of the remove-admin Function 31
 [QA-03] Use amplification-coefficient instead of ann in public functions 32
 [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
 [QA-05] Inefficient Iterative Calculations 34
 [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
 [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
 [QA-08] Maintain Proper Codebase Comments 37
 [QA-09] Remove Redundant Begin Blocks 38
 [QA-10] Non-descriptive BPS Constant Names 39
 [QA-11] Misleading ann Variable Name 40
 [QA-12] Rename withdraw-liquidity to remove-liquidity 41
 [QA-13] Use Tabs Instead of Spaces 42
 [QA-14] Use a SUCCESS Constant 43
 [QA-15] Typographical Error 44

1. About Clarity Alliance

Clarity Alliance is a team of expert whitehat hackers specialising in securing protocols on Stacks.

They have disclosed vulnerabilities that have saved millions in live TVL and conducted thorough reviews for some of the largest projects across the Stacks ecosystem.

Learn more about Clarity Alliance at clarityalliance.org.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

2. Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is 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 Clarity Alliance to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Clarity Alliance’s position is that each company and individual are responsible for their own due diligence and continuous security. Clarity Alliance’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree
to analyze.

The assessment services provided by Clarity Alliance are subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis.

Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third parties. Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Clarity Alliance does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

3. Introduction

A time-boxed security review of the Bitflow Stableswap implementation, where Clarity Alliance reviewed the scope, whilst simultaneously building out a testing suite for the protocol.

4. About Bitflow Stableswap

Bitflow StableSwap is the first protocol designed to enable users to efficiently swap stable assets, including stablecoins, within the Bitcoin ecosystem. It operates on the Stacks layer, a platform specifically designed to facilitate smart contracts and decentralized applications on Bitcoin.

5. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1 Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
 5.1. Impact 4
 5.2. Likelihood 5
 5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
 [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
 [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
 [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
 [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
 [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
 [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
 [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
 [L-03] Pools with an Invalid Fee Structure Can Be Created 20
 [L-04] Pool Validity Check Not Reached for Invalid Pools 21
 [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
 [L-06] Missing Deadline On Swaps 23
 [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
 [L-08] Avoid Using tx-sender for Caller Identification 25
 [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
 [L-10] Lack of Proper Error Handling on Conver-gence Failure 27
 [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
 [QA-01] Protocol Fee Irregularities 30
 [QA-02] Simplification of the remove-admin Function 31
 [QA-03] Use amplification-coefficient instead of ann in public functions 32
 [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
 [QA-05] Inefficient Iterative Calculations 34
 [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
 [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
 [QA-08] Maintain Proper Codebase Comments 37
 [QA-09] Remove Redundant Begin Blocks 38
 [QA-10] Non-descriptive BPS Constant Names 39
 [QA-11] Misleading ann Variable Name 40
 [QA-12] Rename withdraw-liquidity to remove-liquidity 41
 [QA-13] Use Tabs Instead of Spaces 42
 [QA-14] Use a SUCCESS Constant 43
 [QA-15] Typographical Error 44

5.2 Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
    5.1. Impact 4
    5.2. Likelihood 5
    5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
  [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
  [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
  [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
  [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
  [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
  [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
  [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
  [L-03] Pools with an Invalid Fee Structure Can Be Created 20
  [L-04] Pool Validity Check Not Reached for Invalid Pools 21
  [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
  [L-06] Missing Deadline On Swaps 23
  [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
  [L-08] Avoid Using tx-sender for Caller Identification 25
  [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
  [L-10] Lack of Proper Error Handling on Convergence Failure 27
  [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
  [QA-01] Protocol Fee Irregularities 30
  [QA-02] Simplification of the remove-admin Function 31
  [QA-03] Use amplification-coefficient instead of ann in public functions 32
  [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
  [QA-05] Inefficient Iterative Calculations 34
  [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
  [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
  [QA-08] Maintain Proper Codebase Comments 37
  [QA-09] Remove Redundant Begin Blocks 38
  [QA-10] Non-descriptive BPS Constant Names 39
  [QA-11] Misleading ann Variable Name 40
  [QA-12] Rename withdraw-liquidity to remove-liquidity 41
  [QA-13] Use Tabs Instead of Spaces 42
  [QA-14] Use a SUCCESS Constant 43
  [QA-15] Typographical Error 44

ClarityAlliance
Security Review
Bitflow Stableswap

6. Security Assessment Summary

Review Commit Hash:
e949a74b25bbe280708685cd7b40aa1b756166d1

- `contracts/stableswap-pool-trait-v-1-1.clar`
- `contracts/stableswap-core-v-1-1.clar`
- `contracts/sip-010-trait-ft-standard-v-1-1.clar`
- `contracts/stableswap-pool-stx-ststx-v-1-1.clar`

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

7. Executive Summary

Over the course of the security review, Kristian Apostolov, ABA, Marchev engaged with Bitflow to review Bitflow Stableswap. In this period of time a total of **31** issues were uncovered.

Protocol Summary

Protocol Name	Bitflow Stableswap
Repository	https://github.com/BitflowFinance/bitflow-stableswap
Date	September 25th, 2024
Protocol Type	Stableswap AMM

Findings Count

Severity	**Amount**
Critical	1
High	2
Medium	2
Low	11
QA	15
Total Findings	**31**

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

Summary of Findings

ID	Title	Severity	Status
[C-01]	Vulnerability in Pool Configuration When Public Pool Creation is Enabled	Critical	Resolved
[H-01]	Liquidity Fee Cannot Be Updated After Pool Creation	High	Resolved
[H-02]	The Fee Schedule is Malfunctioning	High	Resolved
[M-01]	Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled	Medium	Resolved
[M-02]	Stableswap STX-stSTX Pools are not SIP-10 Compliant	Medium	Resolved
[L-01]	Fees are Incorrectly Applied to the Input Amount on Swaps	Low	Acknowledged
[L-02]	Convergence Threshold Cannot Be Set at Pool Creation	Low	Resolved
[L-03]	Pools with an Invalid Fee Structure Can Be Created	Low	Resolved
[L-04]	Pool Validity Check Not Reached for Invalid Pools	Low	Resolved
[L-05]	Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation	Low	Resolved
[L-06]	Missing Deadline On Swaps	Low	Acknowledged
[L-07]	Minimum D Value Check on Invalid Liquidity Operations is Never Reached	Low	Resolved
[L-08]	Avoid Using tx-sender for Caller Identification	Low	Acknowledged
[L-09]	MINIMUM_SHARES Enforcement May Lead to Liquidity Losses	Low	Resolved
[L-10]	Lack of Proper Error Handling on Convergence Failure	Low	Acknowledged
[L-11]	Incorrect D Value Stored in Pool	Low	Resolved
[QA-01]	Protocol Fee Irregularities	QA	Acknowledged
[QA-02]	Simplification of the remove-admin Function	QA	Resolved
[QA-03]	Use amplification-coefficient instead of ann in public functions	QA	Resolved
[QA-04]	Use Consistent Amount Invalidation Logic in LP Operations	QA	Resolved
[QA-05]	Inefficient Iterative Calculations	QA	Acknowledged

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

ClarityAlliance
Security Review
Bitflow Stableswap

Summary of Findings

ID	Title	Severity	Status
[QA-06]	Redundant Third Call to scale-up-amounts in Liquidity Operation	QA	Acknowledged
[QA-07]	Potential Integer Overflow in D Calculation with High Precision Tokens	QA	Acknowledged
[QA-08]	Maintain Proper Codebase Comments	QA	Resolved
[QA-09]	Remove Redundant Begin Blocks	QA	Acknowledged
[QA-10]	Non-descriptive BPS Constant Names	QA	Resolved
[QA-11]	Misleading ann Variable Name	QA	Resolved
[QA-12]	Rename withdraw-liquidity to remove-liquidity	QA	Acknowledged
[QA-13]	Use Tabs Instead of Spaces	QA	Acknowledged
[QA-14]	Use a SUCCESS Constant	QA	Acknowledged
[QA-15]	Typographical Error	QA	Resolved

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

8. Findings

8.1. Critical Findings

[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled

Description

Creating a Stableswap pool involves the following steps:

- Deploying a contract compatible with the `stableswap-pool-trait-v-1-1` trait.
- Calling `stableswap-core-v-1-1::create-pool` with the deployed contract.

Initially, only trusted addresses or administrators are permitted to call `create-pool`. However, the team can enable public pool creation by setting the `public-pool-creation` variable to `true`, allowing anyone to create pairs.

Within the `create-pool` function, the core contract configures the pool by invoking all three setters specific to the pool trait:

```
(try! (as-contract
  (contract-call? pool-trait update-pool-balances x-amount y-amount total-shares)))
(try! (as-contract (contract-call? pool-trait pool-mint
  (- total-shares MINIMUM_SHARES) caller)))
(try! (as-contract
  (contract-call? pool-trait pool-mint MINIMUM_SHARES pool-contract)))
```

The current pool implementation in `stableswap-pool-stx-ststx-v-1-1` verifies that the caller is the core contract for all these calls:

```
(let (
  (caller tx-sender)
)
  (begin
    (asserts! (is-eq caller CORE_ADDRESS) ERR_NOT_AUTHORIZED)
    ;; ... code ...
```

By crafting a malicious pool contract, an attacker can exploit these implementation details to arbitrarily change the configuration of any existing pool.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

The core issue is that the authorization check is performed against the `tx-sender` instead of the `contract-caller`. Consequently, if public pool creation is enabled, an attacker can create a malicious pool and call `create-pool` with it. Since the core contract calls the malicious pool using `as-contract`, the `tx-sender` is set to the core contract. This allows the call to the malicious pool to occur in the context of the core contract, enabling further execution to any existing pool. As the authorization check is against tx-sender, the malicious call is deemed legitimate.

In the proof of concept, the attacker uses the malicious pool to set the `stableswap-pool-stx-ststx-v-1-1` pool liquidity fees to 100% and designate themselves as the fee recipient.

```
(define-public (update-pool-balances (x-bal uint) (y-bal uint) (d-val uint))
  (let (
    (caller tx-sender)
  )
    (begin
     ;; ... code ...

     ;; @audit attacker can set fees to 100% and change who gets them
     (try!
       (contract-call? .stableswap-pool-stx-ststx-v-1-1 set-liquidity-fee u10000))
     (try! (contract-call? .stableswap-pool-stx-ststx-v-1-1 set-fee-address
       (var-get theft-address)))
```

Through this attack, all existing pool configurations can be altered to values controlled by the attacker, including the specified pool values.

Recommendation

Replace `tx-sender` with `contract-caller` in all instances outside of the SIP-10 transfer function and contract-deployer type variables. After making this change, the use of `as-contract` can be removed from the calls to the pool trait within `create-pool`.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

8.2. High Findings

[H-01] Liquidity Fee Cannot Be Updated After Pool Creation

Description

The `stableswap-pool-stx-ststx-v-1-1` contract permits the updating of liquidity fees through the `set-liquidity-fee` function. This function is restricted and can only be invoked by the `stableswap-core-v-1-1` contract.

However, the `stableswap-core-v-1-1` contract only calls `set-liquidity-fee` during the pool creation process within its `create-pool` function. Unlike other pool parameters, which are configurable and have corresponding setter functions, there is no function available to update the liquidity fee for an existing pool.

Recommendation

Implement a `set-liquidity-fee` function in the `stableswap-core-v-1-1` contract to allow contract administrators to update liquidity fees for existing pools.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[H-02] The Fee Schedule is Malfunctioning

Description

The `.stableswap-core-v-1-1` smart contract is intended to enable Stableswap pools to apply different fees based on the swap direction. It specifies the following fees:

- `y-protocol-fee`
- `y-provider-fee`
- `x-protocol-fee`
- `x-provider-fee`

However, the current implementation incorrectly applies `y-protocol-fee` and `y-provider-fee` for both swap directions. The `x-protocol-fee` and `x-provider-fee` are defined but remain unused within the contract. Consequently, the protocol's fee schedule mechanism is malfunctioning, resulting in the incorrect application of fees.

Recommendation

Ensure the correct fee schedule is applied according to the swap direction. The following functions should be reviewed and updated as necessary:

- `swap-x-for-y`
- `swap-y-for-x`
- `get-dy`
- `get-dx`

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

8.3. Medium Findings

[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled

Description

Due to the absence of support for dynamically deploying contracts via Clarity code, the creation of a Stableswap pool occurs in two distinct steps, each requiring a separate transaction:

- Deploy the pool that implements the `stableswap-pool-trait` trait.
- Set up the Stableswap pool by invoking the `stableswap-core-v-1-1::create-pool` function.

A vulnerability arises when pool creation is publicly enabled (i.e., the `public-pool-creation` flag is set to `true`) because the second step can be front-run. This poses a problem because critical configurations, such as the fee schedule and fee recipient address, are established during pool creation. This vulnerability allows a malicious actor to effectively *take over* deployed pools or, at the very least, disrupt their creation.

Although the mentioned configurations can be subsequently modified and corrected by a protocol admin, with `public-pool-creation` enabled, pool creation should be *permissionless* and not depend on admin intervention, which could potentially be required for each and every pool.

Rectifying pools that have been taken over would become impractical and burdensome for the protocol owners.

Recommendation

Store the contract deployer in a constant within the `stableswap-pool-stx-ststx-v-1-1` contract. When `stableswap-pool-stx-ststx-v-1-1 ::create-pool` is called, also pass the caller principal (taken as `contract-caller`, *not* `tx-sender`) from the core contract. Using these two, verify that the `stableswap-core-v-1-1::create-pool` caller is the same as the `stableswap-pool-stx-ststx-v-1-1` deployer.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
   [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
   [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
   [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
   [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
   [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
   [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
   [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
   [L-03] Pools with an Invalid Fee Structure Can Be Created 20
   [L-04] Pool Validity Check Not Reached for Invalid Pools 21
   [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
   [L-06] Missing Deadline On Swaps 23
   [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
   [L-08] Avoid Using tx-sender for Caller Identification 25
   [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
   [L-10] Lack of Proper Error Handling on Convergence Failure 27
   [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
   [QA-01] Protocol Fee Irregularities 30
   [QA-02] Simplification of the remove-admin Function 31
   [QA-03] Use amplification-coefficient instead of ann in public functions 32
   [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
   [QA-05] Inefficient Iterative Calculations 34
   [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
   [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
   [QA-08] Maintain Proper Codebase Comments 37
   [QA-09] Remove Redundant Begin Blocks 38
   [QA-10] Non-descriptive BPS Constant Names 39
   [QA-11] Misleading ann Variable Name 40
   [QA-12] Rename withdraw-liquidity to remove-liquidity 41
   [QA-13] Use Tabs Instead of Spaces 42
   [QA-14] Use a SUCCESS Constant 43
   [QA-15] Typographical Error 44

Example implementation:

```diff
--- a/contracts/stableswap-core-v-1-1.clar
+++ b/contracts/stableswap-core-v-1-1.clar
@@ -466,7 +466,7 @@
        (asserts!
        (is-eq x-balance-scaled y-balance-scaled) ERR_UNEQUAL_POOL_BALANCES)
        (asserts! (> total-shares MINIMUM_SHARES) ERR_MINIMUM_LP_AMOUNT)
        (asserts! (> (len uri) u0) ERR_INVALID_POOL_URI)
-       (try! (as-contract
- (contract-call? pool-trait create-pool x-token-contract y-token-contract fee-address
+       (try! (as-contract
+ (contract-call? pool-trait create-pool x-token-contract y-token-contract fee-address
        (try! (as-contract
          (contract-call? pool-trait set-x-fees x-protocol-fee x-provider-fee)))
        (try! (as-contract
          (contract-call? pool-trait set-y-fees y-protocol-fee y-provider-fee)))
        (try! (as-contract
          (contract-call? pool-trait set-liquidity-fee liquidity-fee)))

--- a/contracts/stableswap-pool-stx-ststx-v-1-1.clar
+++ b/contracts/stableswap-pool-stx-ststx-v-1-1.clar
@@ -10,10 +10,12 @@
 (define-constant ERR_INVALID_PRINCIPAL (err u1003))
 (define-constant ERR_POOL_NOT_CREATED (err u3002))
 (define-constant ERR_POOL_DISABLED (err u3003))
+(define-constant ERR_NOT_POOL_DEPLOYER (err u3004))

 (define-constant CORE_ADDRESS .stableswap-core-v-1-1)

 (define-constant BPS u10000)
+(define-constant CONTRACT_DEPLOYER tx-sender)

 (define-data-var pool-id uint u0)
 (define-data-var pool-name (string-ascii 256) “”)
@@ -296,12 +298,14 @@
       (name (string-ascii 256)) (symbol (string-ascii 256))
       (uri (string-utf8 256))
       (status bool)
+      (core-caller principal)
    )
    (let (
       (caller tx-sender)
    )
       (begin
        (asserts! (is-eq caller CORE_ADDRESS) ERR_NOT_AUTHORIZED)
+       (asserts! (is-eq core-caller CONTRACT_DEPLOYER) ERR_NOT_POOL_DEPLOYER)
        (var-set pool-id id)
        (var-set pool-name name)
        (var-set pool-symbol symbol)
```

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant

Description

The Stableswap pool contract `stableswap-pool-stx-ststx-v-1-1` implements the `stableswap-pool-trait-v-1-1` trait, which is a subset of the SIP-10 trait. However, the current implementation is not SIP-10 compliant due to the way the trait functions are declared in the `stableswap-pool-trait` trait.

The `get-name` and `get-symbol` functions can return up to 256 ASCII characters, whereas SIP-10 specifically limits these to a maximum of 32 characters.

```
(get-name () (response (string-ascii 256) uint))
(get-symbol () (response (string-ascii 256) uint))
```

Another compliance issue is that the SIP-10 `stableswap-pool-stx-ststx-v-1-1::transfer` function returns incorrect error code ranges. The standard specifies that error codes should start from 1 and increase incrementally, with the first four values already defined in the SIP.

However, the mentioned `transfer` function returns three different error codes outside the indicated range. It also overrides the SIP's error code for an amount greater than 0 (`amount is u3: non-positive`) with `ERR_INVALID_AMOUNT` of `u1002` and the `u4: sender is not the same as tx-sender` error with `ERR_NOT_AUTHORIZED` of `u1001`.

```
(asserts! (is-eq caller sender) ERR_NOT_AUTHORIZED)
(asserts! (is-standard sender) ERR_INVALID_PRINCIPAL)
(asserts! (is-standard recipient) ERR_INVALID_PRINCIPAL)
(asserts! (> amount u0) ERR_INVALID_AMOUNT)
```

This issue is also present in the `token-stx-v-1-1::transfer` function.

Third-party protocols may experience unexpected side effects due to these issues when integrating with the Stableswap pool.

Recommendation

Modify the `stableswap-pool-stx-ststx-v-1-1` and `stableswap-pool-trait-v-1-1` contracts so that the `get-name` and `get-symbol` functions are SIP-10 compliant by limiting their maximum length to 32 characters.

Additionally, since the transfer will already fail if the amount is 0, remove the `ERR_INVALID_AMOUNT` check. Change `ERR_NOT_AUTHORIZED` to return `u4` and have `ERR_INVALID_PRINCIPAL` be the next incremental available value, meaning `u5`.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

8.4. Low Findings

[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps

Description

In the current implementation of the Stableswap core, fees are mistakenly applied to the input token amount instead of the output token amount.

When fees are applied to the input token before the swap, the user receives more output tokens than intended, which reduces the total fees collected by the protocol and the liquidity providers.

This issue is subtle when the pool is balanced and behaves like a constant sum AMM. However, it becomes pronounced when the pool is unbalanced and the curve behaves more like a constant product AMM.

Consider the following example. For simplicity, we will demonstrate the issue with a constant product AMM:

Let's have an x * y = k pool with a fee of 10%, where x = 1000 and y = 1000.

Correct Fee Application

- The user swaps 200 tokens of `X`.
- Without fees, the new reserves of `X` and `Y` are calculated as follows:

```
x' = 1000 + 200 = 1200
y' = k/x = (1000 * 1000) / 1200 = 833
```

- The amount of `Y` the user should receive before any fees is thus:

```
dy = 1000 - 833 = 167
```

The user receives 150 tokens of `Y`, and 17 tokens are collected as fees.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
  5.1. Impact 4
  5.2. Likelihood 5
  5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
  [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
  [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
  [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
  [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
  [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
  [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
  [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
  [L-03] Pools with an Invalid Fee Structure Can Be Created 20
  [L-04] Pool Validity Check Not Reached for Invalid Pools 21
  [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
  [L-06] Missing Deadline On Swaps 23
  [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
  [L-08] Avoid Using tx-sender for Caller Identification 25
  [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
  [L-10] Lack of Proper Error Handling on Convergence Failure 27
  [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
  [QA-01] Protocol Fee Irregularities 30
  [QA-02] Simplification of the remove-admin Function 31
  [QA-03] Use amplification-coefficient instead of ann in public functions 32
  [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
  [QA-05] Inefficient Iterative Calculations 34
  [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
  [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
  [QA-08] Maintain Proper Codebase Comments 37
  [QA-09] Remove Redundant Begin Blocks 38
  [QA-10] Non-descriptive BPS Constant Names 39
  [QA-11] Misleading ann Variable Name 40
  [QA-12] Rename withdraw-liquidity to remove-liquidity 41
  [QA-13] Use Tabs Instead of Spaces 42
  [QA-14] Use a SUCCESS Constant 43
  [QA-15] Typographical Error 44

Incorrect Fee Application

- The user swaps 200 tokens of `X`.
- The 10% fee is applied to the input amount, thus:

```
dy = 200 * 0.9 = 180
```

- The new reserves of `X` and `Y` are calculated as follows:

```
x' = 1000 + 180 = 1180
y' = k/x = (1000 * 1000) / 1180 = 847
```

- The amount of `Y` the user should receive is:

```
dy = 1000 - 847 = 153
```

The user receives 153 tokens of `Y`, which is 3 tokens more compared to the scenario where fees are applied after the swap.

Note: For comparison, in Curve's Stableswap implementation, fees are applied to the output amount.

Recommendation

Modify the fee logic so that fees are applied to the output amount rather than the input amount.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[L-02] Convergence Threshold Cannot Be Set at Pool Creation

Description

When a pool is created, the caller can specify almost all configurable parameters except for the convergence threshold. This threshold is set to a default value of `2` and can only be modified by the administrators of the Stableswap Core contract.

Once pool creation becomes publicly accessible (via the `public-pool-creation` variable), users should have the ability to adjust the default value, at least during the initial pool creation.

Having a configuration that cannot be set initially reduces the flexibility for third-party integrators and places an additional burden on existing administrators to modify the setting if necessary.

Recommendation

Include an optional argument in the `stableswap-core-v-1-1::create-pool` function to allow setting the convergence threshold.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[L-03] Pools with an Invalid Fee Structure Can Be Created

Description

Throughout the codebase, there is no validation for the protocol and provider fees—they are accepted as provided.

This could be particularly problematic when `public-pool-creation` is enabled. The absence of validation allows pools with invalid fee structures to be created, where the fees exceed `u1000` basis points (`BPS_1`).

Such invalid fees would cause transactions to revert during swaps because the fee amounts would be greater than the swap input amount, leading to failed swaps.

Recommendation

Implement input validations for protocol and provider fees during pool creation and fee updates.

Ensure these fees do not exceed `BPS_1` (`u1000` basis points).

Additionally, ensure that both provider and protocol fees, cumulatively, do not exceed `BPS_1`, as they are both calculated and withdrawn from the same amount.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[L-04] Pool Validity Check Not Reached for Invalid Pools

Description

When interacting with a pool that has not been properly initialized—meaning the pool contract was deployed, but `stableswap-core-v-1-1::create-pool` was not called—the error `ERR_INVALID_POOL` should be returned. However, during core operations such as swapping, adding, and removing liquidity, the validation performed by `is-valid-pool` is never reached for an uninitialized pool. This is because the transaction reverts before reaching that point due to a divide-by-zero runtime error when calculating the new D value. This makes it more challenging to identify the correct issue when an error occurs.

Recommendation

To address this, move the `is-valid-pool` check/call from the ending `begin` block to the leading `let` block in the functions `get-dy`, `get-dx`, `get-dlp`, `swap-x-for-y`, `swap-y-for-x`, `add-liquidity`, and `withdraw-liquidity`, using a placeholder value.

Here is an example of how to apply this change to the `add-liquidity` function:

```
@@ -663,6 +663,7 @@
   )
   (let (
     (pool-data (unwrap! (contract-call? pool-trait get-pool) ERR_NO_POOL_DATA))
+ (pool-validity-check (asserts! (is-valid-pool (get pool-id pool-data)
+ (contract-of pool-trait)) ERR_INVALID_POOL))
     (pool-contract (contract-of pool-trait))
     (fee-address (get fee-address pool-data))
     (x-token (get x-token pool-data))
@@ -712,7 +713,6 @@
     (caller tx-sender)
   )
     (begin
-       (asserts! (is-valid-pool (get pool-id pool-data)
- (contract-of pool-trait)) ERR_INVALID_POOL)
       (asserts! (is-eq (get pool-status pool-data) true) ERR_POOL_DISABLED)
       (asserts! (is-eq
         (contract-of x-token-trait) x-token) ERR_INVALID_X_TOKEN)
       (asserts! (is-eq
         (contract-of y-token-trait) y-token) ERR_INVALID_Y_TOKEN)
```

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation

Description

When a pool is created in the core contract, the pool symbol is determined by concatenating the symbols of the two tokens using the private function `create-symbol` . The name is then formed by appending `“-LP”` to the symbol.

Since the symbols are arbitrary and externally controlled, two valid token symbols may create an invalid pool token/name symbol by exceeding the current limit of 256 characters, causing the pool creation to revert.

Note: There is a separate issue where the pools themselves are not SIP-10 compliant due to allowing symbols/names longer than 32 characters. Even if that is addressed, without modifying the symbol and name creation logic to trim the resulting concatenation to fit within 32 characters, the pools would either remain non-SIP-10 compliant or simply revert.

Recommendation

Modify the symbol and name creation logic to ensure that the resulting symbol/name does not exceed 32 characters. This can be achieved by implementing a trimming logic.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Conver-gence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[L-06] Missing Deadline On Swaps

Description

The `swap-x-for-y` and `swap-y-for-x` swap functions from the core contract are missing an equivalent deadline parameter. As such, a user may submit a swap which will remain in the mempool for an extended period of time and be eventually executed at at an inconvenient time for users.

Example, users wants to swap 1000 USDa tokens for 1000 USDb tokens. He submits the swap but it is not picked up by any miners. Due to extreme market conditions, USDb depegs. At this point the swap is carried out and since the Stableswap logic does not take into consideration any external oracle prices, the user's swap is now done resulting losses for him.

Recommendation

Add a deadline parameter for the `swap-x-for-y` and `swap-y-for-x` swap functions.

Normally, in decentralized exchanges, the deadline parameter represents a timestamp. Either use the more unreliable block height for a deadline logic or wait until Nakamoto is deployed and use the newly-introduced block timestamp.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached

Description

When adding liquidity, an updated D value is calculated and checked against its previous value to ensure it is higher.

```
(asserts! (> updated-d d-a) ERR_MINIMUM_D_VALUE)
```

However, this check is performed after a subtraction of the values has already occurred:

```
(dlp (/ (* total-shares (- updated-d d-a)) d-a))
```

As a result, the `ERR_MINIMUM_D_VALUE` error will never be triggered when the new D value is lower than the old one, as the transaction would have already reverted with a panic due to the previous subtraction. This makes it more challenging to identify the correct issue in the event of an error.

Recommendation

Move the `ERR_MINIMUM_D_VALUE` check, found in the `add-liquidity` an `get-dlp` functions, from the ending `begin` block to the `let` block, immediately before calculating the LP number to mint, using a placeholder value.

Example:

```
@@ -708,6 +708,7 @@
          (updated-pool-balances-post-fee
             (scale-down-amounts updated-balance-x-post-fee-scaled updated-balance-y-post-fe
          (updated-x-balance-post-fee (get x-amount updated-pool-balances-post-fee))
          (updated-y-balance-post-fee (get y-amount updated-pool-balances-post-fee))
+         (minimum-d-check (asserts! (> updated-d d-a) ERR_MINIMUM_D_VALUE))
          (dlp (/ (* total-shares (- updated-d d-a)) d-a))
          (caller tx-sender)
       )
@@ -712,7 +713,6 @@
          (asserts! (is-eq
             (contract-of y-token-trait) y-token) ERR_INVALID_Y_TOKEN)
          (asserts! (or (> updated-x-amount u0)
             (> updated-y-amount u0)) ERR_INVALID_AMOUNT)
          (asserts! (> min-dlp u0) ERR_INVALID_AMOUNT)
-         (asserts! (> updated-d d-a) ERR_MINIMUM_D_VALUE)
          (asserts! (>= dlp min-dlp) ERR_MINIMUM_LP_AMOUNT)
          (if (> updated-x-amount u0)
```

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
   [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
   [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
   [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
   [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
   [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
   [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
   [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
   [L-03] Pools with an Invalid Fee Structure Can Be Created 20
   [L-04] Pool Validity Check Not Reached for Invalid Pools 21
   [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
   [L-06] Missing Deadline On Swaps 23
   [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
   [L-08] Avoid Using tx-sender for Caller Identification 25
   [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
   [L-10] Lack of Proper Error Handling on Convergence Failure 27
   [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
   [QA-01] Protocol Fee Irregularities 30
   [QA-02] Simplification of the remove-admin Function 31
   [QA-03] Use amplification-coefficient instead of ann in public functions 32
   [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
   [QA-05] Inefficient Iterative Calculations 34
   [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
   [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
   [QA-08] Maintain Proper Codebase Comments 37
   [QA-09] Remove Redundant Begin Blocks 38
   [QA-10] Non-descriptive BPS Constant Names 39
   [QA-11] Misleading ann Variable Name 40
   [QA-12] Rename withdraw-liquidity to remove-liquidity 41
   [QA-13] Use Tabs Instead of Spaces 42
   [QA-14] Use a SUCCESS Constant 43
   [QA-15] Typographical Error 44

[L-08] Avoid Using `tx-sender` for Caller Identification

Description

Throughout the contract, there are instances where `tx-sender` is used instead of `contract-caller` or passing the caller's address. This practice can lead to security vulnerabilities, as users who fall victim to phishing scams and interact with malicious contracts may inadvertently execute sensitive operations within the codebase.

For example, a user interacting with a malicious contract could unknowingly allow that contract to initiate liquidity pool withdrawals on their behalf via the `stableswap-core-v-1-1::withdraw-liquidity` function.

Similarly, if an admin interacts with a malicious contract, the contract could call the `stableswap-core-v-1-1::set-fee-address` function and redirect the fee receivers to its own address, resulting in a loss of funds for the protocol team.

It is important to note that these scenarios require negligence on the part of the admin or user.

Recommendation

Use `contract-caller` instead of `transfer` in all instances, except within the SIP-10 transfer function and contract-deployer type variables.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
    5.1. Impact 4
    5.2. Likelihood 5
    5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
  [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
  [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
  [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
  [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
  [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
  [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
  [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
  [L-03] Pools with an Invalid Fee Structure Can Be Created 20
  [L-04] Pool Validity Check Not Reached for Invalid Pools 21
  [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
  [L-06] Missing Deadline On Swaps 23
  [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
  [L-08] Avoid Using tx-sender for Caller Identification 25
  [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
  [L-10] Lack of Proper Error Handling on Convergence Failure 27
  [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
  [QA-01] Protocol Fee Irregularities 30
  [QA-02] Simplification of the remove-admin Function 31
  [QA-03] Use amplification-coefficient instead of ann in public functions 32
  [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
  [QA-05] Inefficient Iterative Calculations 34
  [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
  [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
  [QA-08] Maintain Proper Codebase Comments 37
  [QA-09] Remove Redundant Begin Blocks 38
  [QA-10] Non-descriptive BPS Constant Names 39
  [QA-11] Misleading ann Variable Name 40
  [QA-12] Rename withdraw-liquidity to remove-liquidity 41
  [QA-13] Use Tabs Instead of Spaces 42
  [QA-14] Use a SUCCESS Constant 43
  [QA-15] Typographical Error 44

[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses

Description

In the StableSwap `stableswap-core-v-1-1` contract, there is an assertion that requires the total shares of a liquidity provider (LP) to exceed a minimum threshold:

```
(asserts! (> total-shares MINIMUM_SHARES) ERR_MINIMUM_LP_AMOUNT)
```

The issue arises because the `MINIMUM_SHARES` amount does not consider the precision/decimals of the pool tokens. Consequently, this mechanism has varying effects depending on the precision of the tokens involved in the pool.

For low-precision tokens (e.g., `decimals == 2`), the initial liquidity requirement can be problematic, as it may create a significant barrier to entry for creating the pool.

Additionally, this minimum amount of shares is minted directly to the pool itself, effectively reducing the liquidity available to the pool creator:

```
(try! (as-contract (contract-call? pool-trait pool-mint
  (- total-shares MINIMUM_SHARES) caller)))
(try! (as-contract
  (contract-call? pool-trait pool-mint MINIMUM_SHARES pool-contract)))
```

As a result, this could lead to a significant loss of funds for the pool creator if they are unaware of this behavior.

Conversely, if the pool consists of high-precision tokens, the same `MINIMUM_SHARES` value becomes insufficient to represent even a minimal amount of liquidity. For example, in a pool with two tokens, `sETH` and `stETH`, each having `decimals() = 18`, this results in a minimal liquidity requirement of `0.000000000001` tokens, which is worth approximately $2.6e^{-9}$, or effectively dust.

The current implementation does not account for the varying decimal places of different tokens, leading to disproportionate minimum liquidity requirements based on token precision. This inconsistency can hinder the efficient bootstrapping of liquidity pools and negatively impact the overall user experience.

Recommendation

To address the inconsistency in enforcing `MINIMUM_SHARES`, implement scaling of the `MINIMUM_SHARES` based on token decimals. The `MINIMUM_SHARES` constant should be adjusted to match the precision of the token with the higher `decimals`.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
 5.1. Impact 4
 5.2. Likelihood 5
 5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
 [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
 [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
 [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
 [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
 [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
 [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
 [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
 [L-03] Pools with an Invalid Fee Structure Can Be Created 20
 [L-04] Pool Validity Check Not Reached for Invalid Pools 21
 [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
 [L-06] Missing Deadline On Swaps 23
 [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
 [L-08] Avoid Using tx-sender for Caller Identification 25
 [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
 [L-10] Lack of Proper Error Handling on Convergence Failure 27
 [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
 [QA-01] Protocol Fee Irregularities 30
 [QA-02] Simplification of the remove-admin Function 31
 [QA-03] Use amplification-coefficient instead of ann in public functions 32
 [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
 [QA-05] Inefficient Iterative Calculations 34
 [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
 [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
 [QA-08] Maintain Proper Codebase Comments 37
 [QA-09] Remove Redundant Begin Blocks 38
 [QA-10] Non-descriptive BPS Constant Names 39
 [QA-11] Misleading ann Variable Name 40
 [QA-12] Rename withdraw-liquidity to remove-liquidity 41
 [QA-13] Use Tabs Instead of Spaces 42
 [QA-14] Use a SUCCESS Constant 43
 [QA-15] Typographical Error 44

[L-10] Lack of Proper Error Handling on Convergence Failure

Description

In the StableSwap `stableswap-core-v-1-1` contract, the invariant `D` convergence calculation uses the Newton-Raphson method to iteratively solve for the invariant. In rare cases where the pool becomes significantly unbalanced, the Newton-Raphson method may fail to converge within 384 iterations.

When convergence is not achieved, the current implementation mistakenly returns `0`. This results in a runtime panic in almost all relevant pool operations due to a division by zero error. The only exception is withdrawing liquidity, which is correctly handled.

In this state, users would encounter a divide by zero error without understanding the cause, potentially causing panic among the liquidity providers (LPs).

Recommendation

Modify the implementation so that when the Newton-Raphson method fails to converge, it reverts or throws an explicit error instead of returning 0 and eventually causing a division by zero error. This approach clearly indicates that the pool is in an invalid state, signaling to LPs to safely withdraw their funds using the `withdraw-liquidity` function.

A similar approach is employed by Curve's original StableSwap implementation:

```
# convergence typically occurs in 4 rounds or less, this should be unreachable!
#
   if it does happen the pool is borked and LPs can withdraw via `remove_liquidity`
raise
```

From an implementation perspective, modify the `get-d` function to return a `(response UnknownType uint)` type and have it return an error if the `fold-d-for-loop` end result is 0. In all places except `withdraw-liquidity`, further revert the execution.

CONTENTS

1. About Clarity Alliance	2
2. Disclaimer	3
3. Introduction	4
4. About Bitflow Stableswap	4
5. Risk Classification	4
5.1. Impact	4
5.2. Likelihood	5
5.3. Action required for severity levels	5
6. Security Assessment Summary	6
7. Executive Summary	7
8. Findings	7
8.1. Critical Findings	10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled	10
8.2. High Findings	12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation	12
[H-02] The Fee Schedule is Malfunctioning	13
8.3. Medium Findings	14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled	14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant	16
8.4. Low Findings	17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps	17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation	19
[L-03] Pools with an Invalid Fee Structure Can Be Created	20
[L-04] Pool Validity Check Not Reached for Invalid Pools	21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation	22
[L-06] Missing Deadline On Swaps	23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached	24
[L-08] Avoid Using tx-sender for Caller Identification	25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses	26
[L-10] Lack of Proper Error Handling on Convergence Failure	27
[L-11] Incorrect D Value Stored in Pool	28
8.5. QA Findings	30
[QA-01] Protocol Fee Irregularities	30
[QA-02] Simplification of the remove-admin Function	31
[QA-03] Use amplification-coefficient instead of ann in public functions	32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst	33
[QA-05] Inefficient Iterative Calculations	34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation	35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens	36
[QA-08] Maintain Proper Codebase Comments	37
[QA-09] Remove Redundant Begin Blocks	38
[QA-10] Non-descriptive BPS Constant Names	39
[QA-11] Misleading ann Variable Name	40
[QA-12] Rename withdraw-liquidity to remove-liquidity	41
[QA-13] Use Tabs Instead of Spaces	42
[QA-14] Use a SUCCESS Constant	43
[QA-15] Typographical Error	44

[L-11] Incorrect D Value Stored in Pool

Description

In the `swap-x-for-y` and `swap-y-for-x` functions, after a swap is executed, the X and Y tokens are stored in the `pool-trait` pool along with the D invariant value.

For instance, in the `swap-x-for-y` function, the `updated-d` value, which determines the new D, is calculated based on `updated-x-balance-scaled` and `updated-y-balance-scaled`. The `updated-x-balance-scaled` considers the existing X balance plus the deposited X amount minus all fees (both protocol and provider fees).

However, when updating the pool balances, the X balance used excludes only the protocol fees, not the provider fees, as provider fees continue to form liquidity and are effectively *reinvested*.

As a result, the persisted D value is based on the total X balance minus all fees, whereas the actual pool state reflects the X balance minus only protocol fees.

This discrepancy causes the D value to be inconsistent with the pool's actual state. Since the D value stored in `pool-trait` is not directly used within the protocol, this issue does not affect the swap calculations. However, it may impact any external parties or integrators that rely on the D value and the X and Y token balances stored in the `pool-trait` pool.

Recommendation

Ensure consistency in the calculation and application of fees when updating pool balances and computing the D value.

Specifically, ensure that:

- The `updated-x-balance-scaled` in `swap-x-for-y` does not include the protocol fee but does include the provider fee.
- The `updated-y-balance-scaled` in `swap-y-for-x` does not include the protocol fee but does include the provider fee.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
    5.1. Impact 4
    5.2. Likelihood 5
    5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

```
@@ -533,7 +533,7 @@
(x-amount-fees-provider-scaled (/ (* x-amount-scaled provider-fee) BPS_1))
(x-amount-fees-total-scaled
  (+ x-amount-fees-protocol-scaled x-amount-fees-provider-scaled))
(dx-scaled (- x-amount-scaled x-amount-fees-total-scaled))
-       (updated-x-balance-scaled (+ x-balance-scaled dx-scaled))
+       (updated-x-balance-scaled (+
+ (+ x-balance-scaled dx-scaled) x-amount-fees-provider-scaled))
(updated-y-balance-scaled (get-y dx-scaled x-balance-scaled y-balance-scaled
  (* amplification-coefficient BPS_3) convergence-threshold))
(updated-y-balance (get y-amount
  (scale-down-amounts u0 updated-y-balance-scaled x-token-trait y-token-trait)))
(dy (- y-balance updated-y-balance))
@@ -607,7 +607,7 @@
(y-amount-fees-provider-scaled (/ (* y-amount-scaled provider-fee) BPS_1))
(y-amount-fees-total-scaled
  (+ y-amount-fees-protocol-scaled y-amount-fees-provider-scaled))
(dy-scaled (- y-amount-scaled y-amount-fees-total-scaled))
-       (updated-y-balance-scaled (+ y-balance-scaled dy-scaled))
+       (updated-y-balance-scaled (+
+ (+ y-balance-scaled dy-scaled) y-amount-fees-provider-scaled))
(updated-x-balance-scaled (get-x dy-scaled y-balance-scaled x-balance-scaled
  (* amplification-coefficient BPS_3) convergence-threshold))
(updated-x-balance (get x-amount
  (scale-down-amounts updated-x-balance-scaled u0 x-token-trait y-token-trait)))
(dx (- x-balance updated-x-balance))
```

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

8.5. QA Findings

[QA-01] Protocol Fee Irregularities

Description

Throughout the codebase, fees are divided into two categories:

- *Provider fee:* This fee is taken on swaps and left in the pool to be distributed among liquidity providers.
- *Protocol fee:* This fee is taken on swaps and when adding liquidity, and it is sent to a `fee-address`.

The fee referred to as the “protocol fee” is actually sent to a fee address that is set per pool.

While pool creation is limited to admins only, the admins can choose a fee address specific to the protocol team, which will provide protocol incentives.

However, once pool creation is made available to the general public, both the fee amount and its receiver can be specified at the time of creation by the pool deployer. As a result, the protocol team itself may not receive any fees at that point.

Note that while the initial fee schedule is chosen by the pool creator, trusted admins can change it unilaterally after deployment. However, doing this on non-protocol pools would be considered a highly controversial action.

Recommendation

If the aforementioned behavior is intended, acknowledge this issue.

Otherwise, implement the following changes:

Do not allow the protocol fees, liquidation fees (if intended to be sent to the protocol), or the fee receiver principle to be changeable at pool creation, only through admin action. This way, when public pool creation is allowed, all pools should default to actual protocol values to ensure the team has a steady revenue stream.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
   [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
   [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
   [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
   [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
   [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
   [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
   [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
   [L-03] Pools with an Invalid Fee Structure Can Be Created 20
   [L-04] Pool Validity Check Not Reached for Invalid Pools 21
   [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
   [L-06] Missing Deadline On Swaps 23
   [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
   [L-08] Avoid Using tx-sender for Caller Identification 25
   [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
   [L-10] Lack of Proper Error Handling on Convergence Failure 27
   [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
   [QA-01] Protocol Fee Irregularities 30
   [QA-02] Simplification of the remove-admin Function 31
   [QA-03] Use amplification-coefficient instead of ann in public functions 32
   [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
   [QA-05] Inefficient Iterative Calculations 34
   [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
   [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
   [QA-08] Maintain Proper Codebase Comments 37
   [QA-09] Remove Redundant Begin Blocks 38
   [QA-10] Non-descriptive BPS Constant Names 39
   [QA-11] Misleading ann Variable Name 40
   [QA-12] Rename withdraw-liquidity to remove-liquidity 41
   [QA-13] Use Tabs Instead of Spaces 42
   [QA-14] Use a SUCCESS Constant 43
   [QA-15] Typographical Error 44

[QA-02] Simplification of the remove-admin Function

Description

The `remove-admin` function can be streamlined by eliminating two redundant `let` declarations for variables that are used only once: `caller-in-list` and `admin-to-remove-in-list`. By removing these `let` declarations and directly attributing values, as done in previous admin-guarded functions, we can achieve both uniformity and a slight optimization in fees.

This optimization can be applied throughout the codebase contracts.

Recommendation

Eliminate the `caller-in-list` and `admin-to-remove-in-list` variables from the `remove-admin` function and replace their usage with direct value attribution.

Example:

```
 (define-public (remove-admin (admin principal))
    (let (
          (admins-list (var-get admins))
-         (caller-in-list (index-of admins-list tx-sender))
-         (admin-to-remove-in-list (index-of admins-list admin))
          (caller tx-sender)
    )
-         (asserts! (is-some caller-in-list) ERR_NOT_AUTHORIZED)
-         (asserts! (is-some admin-to-remove-in-list) ERR_ADMIN_NOT_IN_LIST)
+         (asserts! (is-some (index-of admins-list caller)) ERR_NOT_AUTHORIZED)
+         (asserts! (is-some (index-of admins-list admin)) ERR_ADMIN_NOT_IN_LIST)
          (asserts! (not
             (is-eq admin CONTRACT_DEPLOYER)) ERR_CANNOT_REMOVE_CONTRACT_DEPLOYER)
          (var-set admin-helper admin)
          (var-set admins (filter admin-not-removeable admins-list))
```

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-03] Use amplification-coefficient instead of ann in public functions

Description

The `get-d`, `get-x`, and `get-y` functions in the core contract currently receive an `ann` parameter. This parameter is consistently the product of the `amplification-coefficient` and the `BPS_3` constant:

```
(* amplification-coefficient BPS_3)
```

This approach is redundant since there are no calls to these functions without this multiplication. Additionally, it may cause confusion for integrators, as Curve's original Stableswap implementation uses the `_amp` parameter to represent the amplification coefficient directly when calculating the D variable. In contrast, the current implementation requires the value to be multiplied by a factor before being passed to the public function.

To improve clarity and prevent confusion among integrators and technical users of the contract, it is recommended to use `amplification-coefficient` as a parameter for these functions.

Recommendation

Replace the `ann` parameter with `amplification-coefficient` in the `get-d`, `get-x`, and `get-y` functions. Additionally, incorporate the `BPS_3` multiplication within these functions.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-04] Use Consistent Amount Invalidation Logic in LP Operations

Description

In the core contract, when adding liquidity, the amounts of X and Y tokens (after fees) are verified to ensure that at least one of them is greater than zero using the following check:

```
(asserts! (or (> updated-x-amount u0)
  (> updated-y-amount u0)) ERR_INVALID_AMOUNT)
```

However, when withdrawing liquidity, the same validation is performed using a different approach:

```
(asserts! (> (+ x-amount y-amount) u0) ERR_INVALID_AMOUNT)
```

The use of two different mechanisms for the same validation reduces code uniformity, and the first check involves more operations than necessary.

Recommendation

In the `add-liquidity` function, modify the `ERR_INVALID_AMOUNT` check to match the implementation used in the `withdraw-liquidity` function.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-05] Inefficient Iterative Calculations

Description

The `.stableswap-core-v-1-1` contract utilizes inefficient iterative calculations, potentially leading to unnecessarily high transaction costs.

Currently, the implementation employs Clarity's `fold` function with a fixed list of 384 items for these calculations:

```
lisp (define-constant index-list (list u1 u2 ... u384))
```

This method is inefficient because the `fold` operation continues even if converged values are found early in the process, which could result in higher transaction costs for users interacting with the contract.

For context, the original Curve implementation uses a `for` loop with a maximum of 255 iterations and includes an early return mechanism, resulting in a more optimal solution. However, Clarity, being a pure functional language, does not support `for` loops, making it challenging to directly replicate this approach.

Recommendation

Given that the Clarity VM supports a maximum call stack depth of only 64, which may be insufficient for implementing a recursive solution that can also handle heavily unbalanced pools, we recommend reducing the number of iterations to 255.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
  5.1. Impact 4
  5.2. Likelihood 5
  5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
  [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
  [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
  [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
  [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
  [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
  [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
  [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
  [L-03] Pools with an Invalid Fee Structure Can Be Created 20
  [L-04] Pool Validity Check Not Reached for Invalid Pools 21
  [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
  [L-06] Missing Deadline On Swaps 23
  [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
  [L-08] Avoid Using tx-sender for Caller Identification 25
  [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
  [L-10] Lack of Proper Error Handling on Convergence Failure 27
  [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
  [QA-01] Protocol Fee Irregularities 30
  [QA-02] Simplification of the remove-admin Function 31
  [QA-03] Use amplification-coefficient instead of ann in public functions 32
  [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
  [QA-05] Inefficient Iterative Calculations 34
  [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
  [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
  [QA-08] Maintain Proper Codebase Comments 37
  [QA-09] Remove Redundant Begin Blocks 38
  [QA-10] Non-descriptive BPS Constant Names 39
  [QA-11] Misleading ann Variable Name 40
  [QA-12] Rename withdraw-liquidity to remove-liquidity 41
  [QA-13] Use Tabs Instead of Spaces 42
  [QA-14] Use a SUCCESS Constant 43
  [QA-15] Typographical Error 44

[QA-06] Redundant Third Call to `scale-up-amounts` in Liquidity Operation

Description

In the `add-liquidity` and `get-dlp` functions of the `stableswap-core-v-1-1` contract, there is a third call to the `scale-up-amounts` function to calculate the updated, scaled X and Y token values:

```
(updated-pool-balances-scaled
  (scale-up-amounts updated-x-balance updated-y-balance x-token-trait y-token-trait))
(updated-x-balance-scaled (get x-amount updated-pool-balances-scaled))
(updated-y-balance-scaled (get y-amount updated-pool-balances-scaled))
```

This third call, which also involves two additional external calls to the tokens' `get-decimals` SIP-10 function, is unnecessary. The scaled updated balances can be directly obtained by adding the `*-amount-scaled` to the `*-balance-scaled` variables.

Recommendation

Simplify both the `add-liquidity` and `get-dlp` functions by calculating the `updated-y-balance-scaled` and `updated-x-balance-scaled` updated-x-balance-scaled variables without calling `scale-up-amounts`.

Example for `get-dlp`:

```
(amplification-coefficient (get amplification-coefficient pool-data))
-         (updated-x-balance (+ x-balance x-amount))
-         (updated-y-balance (+ y-balance y-amount))
          (amounts-added-scaled
            (scale-up-amounts x-amount y-amount x-token-trait y-token-trait))
          (x-amount-scaled (get x-amount amounts-added-scaled))
          (y-amount-scaled (get y-amount amounts-added-scaled))
          (pool-balances-scaled
            (scale-up-amounts x-balance y-balance x-token-trait y-token-trait))
          (x-balance-scaled (get x-amount pool-balances-scaled))
          (y-balance-scaled (get y-amount pool-balances-scaled))
-         (updated-pool-balances-scaled
- (scale-up-amounts updated-x-balance updated-y-balance x-token-trait y-token-trait))
-         (updated-x-balance-scaled (get x-amount updated-pool-balances-scaled))
-         (updated-y-balance-scaled (get y-amount updated-pool-balances-scaled))
+         (updated-x-balance-scaled (+ x-amount-scaled x-balance-scaled))
+         (updated-y-balance-scaled (+ y-amount-scaled y-balance-scaled))
          (d-a (get-d x-balance-scaled y-balance-scaled
            (* amplification-coefficient BPS_3) convergence-threshold))
          (d-b (get-d updated-x-balance-scaled updated-y-balance-scaled
            (* amplification-coefficient BPS_3) convergence-threshold))
```

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens

Description

In the StableSwap core contract (`stableswap-core-v-1-1.clar`), the calculation of the invariant `D` is vulnerable to integer overflow when dealing with tokens that have a large number of decimals:

```
(new-d-partial (> (* current-d new-d-partial-x) (* BPS_3 current-y-balance)))
```

For example, consider a pool consisting of `sbETH` (imaginary Stacks Bridged ETH) and `sbstETH` (imaginary Stacks Bridged stETH), both of which are SIP-010 tokens with `u18` decimals. This pool contains `100 sbETH` and `100 sbstETH`, representing approximately $0.5 million in liquidity. The result of the above calculation is `10000000000000000000000000000000000000000` , a 133-bit number. However, the contract uses `uint` , a 128-bit integer data type, which will result in an overflow. This overflow disrupts the D invariant calculation and compromises the pool's stability.

Recommendation

If the protocol is expected to handle tokens with a large number of decimals, consider implementing multi-precision arithmetic. Although Clarity does not currently support multi-precision arithmetic, it can be adapted from other languages. For instance, use two `uint` variables to represent 256-bit numbers and implement the necessary overflow and carry logic. This approach would only require changes to internal representations, without needing to modify external APIs or interfaces.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

ClarityAlliance
Security Review

Bitflow
Stableswap

[QA-08] Maintain Proper Codebase Comments

Description

The `stableswap-core-v-1-1` contract is a newer version of the previous Bitflow Stableswap implementation. However, the previous version is significantly better commented.

Here are some examples of comments that can be adapted and incorporated into the new version: `[1]`, `[2]`, `[3]`, `[4]`, `[5]`, `[6]`, `[7]`, `[8]`, and `[9]`.

Additionally, in the current version of the protocol, there are some outdated comments that should be removed:

- `[1]`: *should we mint minimum share to pool contract like xyk core?*
  The shares are already correctly minted.
- `[2]`: *ability to enable/disable single sided liquidity?*
  This comment is irrelevant as it pertains to an internal feature discussion.
- `[3]`: *fee-address or protocol-address?*
  This comment is also irrelevant as it pertains to an internal feature discussion.

Recommendation

Add comprehensive comments similar to those in the initial implementation version and remove the outdated comments mentioned above.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-09] Remove Redundant Begin Blocks

Description

Throughout the codebase, after `let` declarations, there are unnecessary `begin` blocks added instead of directly writing the subsequent statements, which is normally allowed by the `let` block. This redundancy is present in almost all contracts, and all instances of these `begin` code blocks can be removed.

Recommendation

Remove the `begin` code blocks from the entire codebase and integrate the inner logic directly after the `let` variable declarations.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
    5.1. Impact 4
    5.2. Likelihood 5
    5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
    [C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
    [H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
    [H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
    [M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
    [M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
    [L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
    [L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
    [L-03] Pools with an Invalid Fee Structure Can Be Created 20
    [L-04] Pool Validity Check Not Reached for Invalid Pools 21
    [L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
    [L-06] Missing Deadline On Swaps 23
    [L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
    [L-08] Avoid Using tx-sender for Caller Identification 25
    [L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
    [L-10] Lack of Proper Error Handling on Convergence Failure 27
    [L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
    [QA-01] Protocol Fee Irregularities 30
    [QA-02] Simplification of the remove-admin Function 31
    [QA-03] Use amplification-coefficient instead of ann in public functions 32
    [QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
    [QA-05] Inefficient Iterative Calculations 34
    [QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
    [QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
    [QA-08] Maintain Proper Codebase Comments 37
    [QA-09] Remove Redundant Begin Blocks 38
    [QA-10] Non-descriptive BPS Constant Names 39
    [QA-11] Misleading ann Variable Name 40
    [QA-12] Rename withdraw-liquidity to remove-liquidity 41
    [QA-13] Use Tabs Instead of Spaces 42
    [QA-14] Use a SUCCESS Constant 43
    [QA-15] Typographical Error 44

[QA-10] Non-descriptive BPS Constant Names

Description

The `stableswap-core-v-1-1` contract employs constants named `BPS_1` , `BPS_2` , etc., which do not provide descriptive names that convey their purpose or functionality.

This lack of descriptive naming conventions makes the code challenging to understand and maintain, potentially leading to misunderstandings or errors during development and maintenance.

Recommendation

Adopt more descriptive constant names that clearly convey the semantics and intent of each constant.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-11] Misleading `ann` Variable Name

Description

The use of `ann` as a standalone variable name or as part of other variables throughout the contract is misleading because it represents `A*n` rather than `A*n^n`. This naming likely originates from the `Ann` variable in Curve's original StableSwap implementation. However, this could cause confusion because `ann` does not reflect the definition of `A*n^n` as presented in the original StableSwap whitepaper.

In Curve's implementation, the variable `Ann` actually represents `A*n^(n-1)` (as explained in this Curve research paper). While this naming convention makes sense in Curve's context, where the model supports an arbitrary number of tokens (represented by `n`), in Bitflow's StableSwap implementation, n is fixed at `2`. Consequently, `ann` effectively represents `A*n^(n-1) = A*n^(2-1) = A*n^1 = A * n`.

This discrepancy between the notation and its actual representation can lead to misunderstandings and confusion among developers and security researchers.

Recommendation

To avoid confusion and improve clarity, rename all variables containing `ann` to `an`. This will accurately reflect the variable's meaning as `A*n`, improving the readability and maintainability of the codebase.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-12] Rename withdraw-liquidity to remove-liquidity

Description

From a contextual standpoint, adding liquidity to the pool is performed through the `add-liquidity` function, not via a `deposit-liquidity` function. However, removing liquidity from a pool is currently executed using the `withdraw-liquidity` function, which is not the direct opposite of adding but rather of depositing.

This inconsistency creates a lack of uniformity and introduces ambiguity, as the term `withdrawing` is typically associated with depositing and borrowing/lending activities, not with liquidity provision.

Recommendation

Rename the `withdraw-liquidity` function to `remove-liquidity`.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-13] Use Tabs Instead of Spaces

Description

For Clarity smart contracts, minimizing code size is crucial. Using tabs instead of spaces for indentation can help reduce code size while maintaining readability.

Spaces increase the code size because each space character occupies 1 byte of storage. In contrast, a tab, which can visually represent any number of spaces (commonly 2 or 4), occupies only 1 byte. Therefore, using tabs instead of spaces can significantly reduce storage requirements. For example, a 2-space indent visually appears the same as a tab but requires 2 bytes, doubling the storage space needed compared to a tab.

Recommendation

Convert all instances of whitespace characters to tabs for indentation.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

[QA-14] Use a SUCCESS Constant

Description

Throughout the codebase, there are instances where the response `(ok true)` is returned. In all these cases, this response indicates that the function call was successful, rather than serving as a contextual flag (e.g., it is not used in functions with names like `is-*`).

Recommendation

To enhance code readability, a constant `SUCCESS` should be created and used in place of the `(ok true)` response.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow Stableswap 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. Critical Findings 10
[C-01] Vulnerability in Pool Configuration When Public Pool Creation is Enabled 10
8.2. High Findings 12
[H-01] Liquidity Fee Cannot Be Updated After Pool Creation 12
[H-02] The Fee Schedule is Malfunctioning 13
8.3. Medium Findings 14
[M-01] Pool Creation Vulnerable to Front-Running When Public Pool Creation is Enabled 14
[M-02] Stableswap STX-stSTX Pools are not SIP-10 Compliant 16
8.4. Low Findings 17
[L-01] Fees are Incorrectly Applied to the Input Amount on Swaps 17
[L-02] Convergence Threshold Cannot Be Set at Pool Creation 19
[L-03] Pools with an Invalid Fee Structure Can Be Created 20
[L-04] Pool Validity Check Not Reached for Invalid Pools 21
[L-05] Pool Symbol and Name Formation Logic Can Cause DoS in Pool Creation 22
[L-06] Missing Deadline On Swaps 23
[L-07] Minimum D Value Check on Invalid Liquidity Operations is Never Reached 24
[L-08] Avoid Using tx-sender for Caller Identification 25
[L-09] MINIMUM_SHARES Enforcement May Lead to Liquidity Losses 26
[L-10] Lack of Proper Error Handling on Convergence Failure 27
[L-11] Incorrect D Value Stored in Pool 28
8.5. QA Findings 30
[QA-01] Protocol Fee Irregularities 30
[QA-02] Simplification of the remove-admin Function 31
[QA-03] Use amplification-coefficient instead of ann in public functions 32
[QA-04] Use Consistent Amount Invalidation Logic in LP Operationst 33
[QA-05] Inefficient Iterative Calculations 34
[QA-06] Redundant Third Call to scale-up-amounts in Liquidity Operation 35
[QA-07] Potential Integer Overflow in D Calculation with High Precision Tokens 36
[QA-08] Maintain Proper Codebase Comments 37
[QA-09] Remove Redundant Begin Blocks 38
[QA-10] Non-descriptive BPS Constant Names 39
[QA-11] Misleading ann Variable Name 40
[QA-12] Rename withdraw-liquidity to remove-liquidity 41
[QA-13] Use Tabs Instead of Spaces 42
[QA-14] Use a SUCCESS Constant 43
[QA-15] Typographical Error 44

ClarityAlliance
Security Review

Bitflow
Stableswap

[QA-15] Typographical Error

Description

Throughout the codebase, the duplicated `admin-not-removeable` private function contains a typographical error. The word `removeable` should be corrected to `removable`.

Recommendation

Correct the typo in all relevant contracts within the codebase.