ClarityAlliance

BITFLOW-XYK SMART CONTRACT AUDIT

Conducted by:
KRISTIAN APOSTOLOV, 0X3B, STORMY

JUNE 17TH, 2024

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

1. About Clarity Alliance

Clarity Alliance is a team of expert whitehat hackers specializing in securing protocols on Stacks. They have disclosed vulnerabilities that have saved millions in live TVL and have conducted thorough reviews for various 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-xyk 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 5
7. Executive Summary 6
8. Findings 8
8.1. High Findings 8
[H-01] add-liquidity DoS 8
[H-02] swap helpers using Y will always revert 9
8.2. Medium Findings 10
[M-01] get-dlp calculates shares incorrectly 10
[M-02] Removing all liquidity from a pool will permanently DoS the pair 11
8.3. Low Findings 12
[L-01] Event spamming due to unvalidated parameter 12
8.4. QA Findings 13
[QA-01] Expression too long 13
[QA-02] get-dlp doesn't use LP limiting ratios 14
[QA-03] functions use variables only once 15
[QA-04] pool-transfer prints unnecessary information 16
[QA-05] min-output is not needed 17
[QA-06] Use constants 18

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-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

3. Introduction

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

4. About Bitflow-xyk

Bitflow-xyk is a constant product AMM modeled after Uniswap V2. It supports SIP-10 pairs as well as SIP-10 to native STX pairs.

The protocol includes built-in helper logic to enable atomic multi-hop swaps seamlessly.

Learn more about Bitflow-xyk at bitflow.finance.

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-xyk 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 5
7. Executive Summary 6
8. Findings 8
8.1. High Findings 8
[H-01] add-liquidity DoS 8
[H-02] swap helpers using Y will always revert 9
8.2. Medium Findings 10
[M-01] get-dlp calculates shares incorrectly 10
[M-02] Removing all liquidity from a pool will permanently DoS the pair 11
8.3. Low Findings 12
[L-01] Event spamming due to unvalidated parameter 12
8.4. QA Findings 13
[QA-01] Expression too long 13
[QA-02] get-dlp doesn't use LP limiting ratios 14
[QA-03] functions use variables only once 15
[QA-04] pool-transfer prints unnecessary information 16
[QA-05] min-output is not needed 17
[QA-06] Use constants 18

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

6. Security Assessment Summary

Review Commit Hash:
`4fa4ec346ec6ce6426f7051e61d2871478d5f64b`

Scope

The following contracts were in the scope of the security review:

- `contracts/xyk-pool-trait-v-1-1.clar`
- `contracts/token-stx-v-1-1.clar`
- `contracts/xyk-pool-stx-aeusdc-v-1-1.clar`
- `contracts/xyk-swap-helper-v-1-1.clar`
- `contracts/sip-010-trait-ft-standard-v-1-1.clar`
- `contracts/xyk-core-v-1-1.clar`

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn’t use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

7. Executive Summary

Over the course of the security review, Kristian Apostolov, 0×3b, Stormy engaged with Bitflow to review Bitflow-xyk. In this period of time a total of **11** issues were uncovered.

Protocol Summary

Protocol Name	Bitflow-xyk
Repository	https://github.com/BitflowFinance/bitflow-xyk
Date	June 17th, 2024
Protocol Type	Constant Product AMM
SLOC	914

Findings Count

Severity	**Amount**
High	2
Medium	2
Low	1
QA	6
Total Findings	**11**

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

Summary of Findings

ID	Title	Severity	Status
[H-01]	add-liquidity DoS	High	Resolved
[H-02]	swap helpers using Y will always revert	High	Resolved
[M-01]	get-dlp calculates shares incorrectly	Medium	Resolved
[M-02]	Removing all liquidity from a pool will permanently DoS the pair	Medium	Resolved
[L-01]	Event spamming due to unvalidated parameter	Low	Resolved
[QA-01]	Expression too long	QA	Resolved
[QA-02]	get-dlp doesn't use LP limiting ratios	QA	Resolved
[QA-03]	functions use variables only once	QA	Resolved
[QA-04]	pool-transfer prints unnecessary information	QA	Resolved
[QA-05]	min-output is not needed	QA	Resolved
[QA-06]	Use constants	QA	Resolved

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

8. Findings

8.1. High Findings

[H-01] `add-liquidity` DoS

Description

`add-liquidity` uses `margin-of-err` to ensure the `add-liquidity` function doesn't change the x:y ratio by more than 1 BPS (0.01%). However if a swap happens before `add-liquidity`, the transaction will revert.

It is undesirable for users to be able to prevent others from adding liquidity by simply swapping. Not only can this cause DOS attacks, but it will also happen naturally in fast markets where swaps occur constantly.

Recommendation

Instead of restricting the user, get the current K and add the liquidity based on it.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

[H-02] swap helpers using Y will always revert

Location: xyk-swap-helper-v-1-1.clar

Description

`xyk-swap-helper-v-1-1` doesn't correctly implement swaps that end with `Y`. This is due to it always forcing the output token from the first pool swap to be `X` in the second pool swap.

However, if the second swap is `Y` → `X`, and the second pool has `X` as the output token from the first swap, the transaction will revert. This occurs due to us attempting to swap `Y` → `X`, though we have an `X` balance.

Example:

```
X      Y                           X    Y
     xx -> STX : USDC => STX for USDC => USDC : BTC
     yy -> STX : USDC => USDC for STX => STX : ???
     xy -> STX : USDC => STX for USDC => USDC : ???
     yx -> STX : USDC => USDC for STX => STX : USDC
```

Notice how in the table we are forced to set pool[B]'s `X` value as the output value from the swap inside pool[A].

For example, if we did `USDC` for `STX`, then pool[B]'s `X` would be `STX`. However, since we are swapping `Y` → `X`, having `X` instead of `Y` will revert the swap.

Recommendation

Configure the output token from pool[A] as Y in pool[B], for any helper function that ends in Y (yy, xy, yx). Thus, the output token from pool[A], needed for the swap, will be the input token in pool[B].

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

8.2. Medium Findings

[M-01] `get-dlp` calculates shares incorrectly

Location: xyk-pool-stx-aeusdc-v-1-1.clar:get-dlp

Description

Shares are calculated by doing sqrti(K), however get-dlp calculates them by doing k - total-shares, resulting in a different value, compared to what is actually minted inside add-liquidity.

Recommendation

Change the formula to be identical to the implementation in add-liquidity.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
  8.1. High Findings 8
    [H-01] add-liquidity DoS 8
    [H-02] swap helpers using Y will always revert 9
  8.2. Medium Findings 10
    [M-01] get-dlp calculates shares incorrectly 10
    [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
  8.3. Low Findings 12
    [L-01] Event spamming due to unvalidated parameter 12
  8.4. QA Findings 13
    [QA-01] Expression too long 13
    [QA-02] get-dlp doesn't use LP limiting ratios 14
    [QA-03] functions use variables only once 15
    [QA-04] pool-transfer prints unnecessary information 16
    [QA-05] min-output is not needed 17
    [QA-06] Use constants 18

[M-02] Removing all liquidity from a pool will permanently DoS the pair

Location: xyk-core-v-1-1.clar:add-liquidity

Description

The protocol has 2 distinct ways of minting LP receipt tokens:

- Minting initial LP tokens when creating a pool through `create-pool`
- Adding more liquidity to a pair through `add-liquity`.

The former uses `sqrti(x * y)` for calculating the receipt tokens to mint, whilst the latter does use `min((x-amount * total-shares / x-balance), (y-amount * total-shares / y-balance))`

The issue with using this approach is that `add-liquity` and subsequently the whole trading pair will become permanently DoS'd due to `[token]-amount * total-shares / [token]-balance)`

The formula directly relies on `[token]-balance` - the balance of either token, to always be `> 0`.

Given that the initial shares minted upon pool creation get withdrawn, the whole trading pair will seize functioning.

This does not present a direct threat now, as only trusted admin addresses can create pools, though it is intended for pool creation to eventually be permissionless, hence the issue could be utilized maliciously to prevent any new token pair from existing on the protocol.

Recommendation

Consider implementing a minimum amount of LP tokens, which get subtracted from the initial LP amount for the creator, and get minted to the pool contract directly.

The above solution will make having 0 LP supply for any single pool impossible, and thus mitigate the issue.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
  8.1. High Findings 8
    [H-01] add-liquidity DoS 8
    [H-02] swap helpers using Y will always revert 9
  8.2. Medium Findings 10
    [M-01] get-dlp calculates shares incorrectly 10
    [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
  8.3. Low Findings 12
    [L-01] Event spamming due to unvalidated parameter 12
  8.4. QA Findings 13
    [QA-01] Expression too long 13
    [QA-02] get-dlp doesn't use LP limiting ratios 14
    [QA-03] functions use variables only once 15
    [QA-04] pool-transfer prints unnecessary information 16
    [QA-05] min-output is not needed 17
    [QA-06] Use constants 18

8.3. Low Findings

[L-01] Event spamming due to unvalidated parameter

Location: xyk-core-v-1-1.clar

Description

The `pool-trait` parameter across the swap and liquidity mutation flows does not have proper input validation exercised on it. A rogue pool can be passed in order to spam fraudulent events through the core contract.

Recommendation

Consider implementing proper input validation by checking whether the pool is within the `pools` mapping.

CONTENTS

1. About Clarity Alliance 2

2. Disclaimer 3

3. Introduction 4

4. About Bitflow-xyk 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 5

7. Executive Summary 6

8. Findings 8

8.1. High Findings 8

[H-01] add-liquidity DoS 8

[H-02] swap helpers using Y will always revert 9

8.2. Medium Findings 10

[M-01] get-dlp calculates shares incorrectly 10

[M-02] Removing all liquidity from a pool will permanently DoS the pair 11

8.3. Low Findings 12

[L-01] Event spamming due to unvalidated parameter 12

8.4. QA Findings 13

[QA-01] Expression too long 13

[QA-02] get-dlp doesn’t use LP limiting ratios 14

[QA-03] functions use variables only once 15

[QA-04] pool-transfer prints unnecessary information 16

[QA-05] min-output is not needed 17

[QA-06] Use constants 18

8.4. QA Findings

[QA-01] Expression too long

Location: token-stx-v-1-1.clar:set-token-uri

Description

`set-token-uri` , `set-pool-uri` and `create-pool` use the assert below to check if the URI length is greater than 0:
`(asserts! (is-eq (> (len uri) u0) true) ERR_INVALID_TOKEN_URI)`

Recommendation

This check can be shortened in order to save gas:

`(asserts! (> (len uri) u0) ERR_INVALID_TOKEN_URI)`

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

[QA-02] `get-dlp` doesn’t use LP limiting ratios

Location: xyk-pool-stx-aeusdc-v-1-1.clar:get-dlp

Description

`get-dlp` doesn’t use `new-ratio`, `current-ratio`, or `margin-of-err`.

Recommendation

You can remove them. Alternatively, use them to check if `add-liquidity` will pass with the given inputs (i.e., revert if `margin-of-err` is exceeded).

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
   8.1. High Findings 8
      [H-01] add-liquidity DoS 8
      [H-02] swap helpers using Y will always revert 9
   8.2. Medium Findings 10
      [M-01] get-dlp calculates shares incorrectly 10
      [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
   8.3. Low Findings 12
      [L-01] Event spamming due to unvalidated parameter 12
   8.4. QA Findings 13
      [QA-01] Expression too long 13
      [QA-02] get-dlp doesn't use LP limiting ratios 14
      [QA-03] functions use variables only once 15
      [QA-04] pool-transfer prints unnecessary information 16
      [QA-05] min-output is not needed 17
      [QA-06] Use constants 18

[QA-03] functions use variables only once

Location: xyk-core-v-1-1.clar

Description

`set-pool-status` and the functions below initialize `pool-created`, `pool-contract`, `pool-name`, and `pool-id`, but they use them only once. There is no need to initialize a variable if it's used only once, as this increases the gas cost of the operation.

Recommendation

Ensure initialized variables are used more than once, or else they can be inlined.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
    8.1. High Findings 8
        [H-01] add-liquidity DoS 8
        [H-02] swap helpers using Y will always revert 9
    8.2. Medium Findings 10
        [M-01] get-dlp calculates shares incorrectly 10
        [M-02] Removing all liquidity from a pool will permanently DoS the pair 11
    8.3. Low Findings 12
        [L-01] Event spamming due to unvalidated parameter 12
    8.4. QA Findings 13
        [QA-01] Expression too long 13
        [QA-02] get-dlp doesn't use LP limiting ratios 14
        [QA-03] functions use variables only once 15
        [QA-04] pool-transfer prints unnecessary information 16
        [QA-05] min-output is not needed 17
        [QA-06] Use constants 18

ClarityAlliance
Security Review

BITFLOW-XYK

[QA-04] `pool-transfer` prints unnecessary information

Location: xyk-pool-stx-aeusdc-v-1-1.clar

Description

`pool-transfer` and all other `CORE_ADDRESS` restricted functions print unnecessary information. The print statement below includes `caller`, but since the caller can only be `CORE_ADDRESS`, this is redundant. The current print statement will always show `caller: CORE_ADDRESS`, making the caller variable unnecessary.

```
(print {action: “pool-transfer”, caller: caller, data: {token: token-contract, amount: amount, recipient: recipient}})
```

Recommendation

Change the print statement to: `(print {action: “pool-transfer”, data: {token: token-contract, amount: amount, recipient: recipient}})`

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Bitflow-xyk 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 5
7. Executive Summary 6
8. Findings 8
8.1. High Findings 8
[H-01] add-liquidity DoS 8
[H-02] swap helpers using Y will always revert 9
8.2. Medium Findings 10
[M-01] get-dlp calculates shares incorrectly 10
[M-02] Removing all liquidity from a pool will permanently DoS the pair 11
8.3. Low Findings 12
[L-01] Event spamming due to unvalidated parameter 12
8.4. QA Findings 13
[QA-01] Expression too long 13
[QA-02] get-dlp doesn't use LP limiting ratios 14
[QA-03] functions use variables only once 15
[QA-04] pool-transfer prints unnecessary information 16
[QA-05] min-output is not needed 17
[QA-06] Use constants 18

[QA-05] `min-output` is not needed

Location: xyk-swap-helper-v-1-1.clar:swap-helper-xx

Description

The `min-output` assert inside `swap-helper-xx` is not needed as the check is already handled inside the second swap: `(asserts! (> swap-b min-output) ERR_MINIMUM_OUTPUT)`

Recommendation

Remove the assert.

CONTENTS

1. About Clarity Alliance 2

2. Disclaimer 3

3. Introduction 4

4. About Bitflow-xyk 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 5

7. Executive Summary 6

8. Findings 8

8.1. High Findings 8

[H-01] add-liquidity DoS 8

[H-02] swap helpers using Y will always revert 9

8.2. Medium Findings 10

[M-01] get-dlp calculates shares incorrectly 10

[M-02] Removing all liquidity from a pool will permanently DoS the pair 11

8.3. Low Findings 12

[L-01] Event spamming due to unvalidated parameter 12

8.4. QA Findings 13

[QA-01] Expression too long 13

[QA-02] get-dlp doesn’t use LP limiting ratios 14

[QA-03] functions use variables only once 15

[QA-04] pool-transfer prints unnecessary information 16

[QA-05] min-output is not needed 17

[QA-06] Use constants 18

ClarityAlliance
Security Review

BITFLOW-XYK

[QA-06] Use constants

Location: xyk-core-v-1-1.clar

Description

The value `u10000` is used in multiple places inside `xyk-core-v-1-1`. Such values should be marked as constants to improve code readability.

Recommendation

Define a new constant and use it in places where `u10000` is hardcoded: `(define-constant BPS u10000)`