xToken - xU3LP Quantstamp Review

May 2021

Poming Lee Notes

Target commit hash:
https://github.com/xtokenmarket/xu3lp/compare/4454137b14e1c9353bd87803b242497a6b715df6..fdaabe9b0da6a23b4540d4aec889956404c5b1f0

Findings

Unexpected result if decimal of token0 or token1 is larger than `TOKEN_DECIMAL_REPRESENTATION`

High
Status: Acknowledged, xToken team commented that “For the first finding, that is purposely so, since the token0 and token1 decimals for the first two waves of tokens are with a maximum of 18 decimals”.
`contracts/xU3LPStable.sol`: function `initialize` should check and revert if either the decimal of token0 or token1 is larger than `TOKEN_DECIMAL_REPRESENTATION`. This suggestion is based on the current implementation of `function getToken0AmountInWei(uint256 amount), `function getToken0AmountInNativeDecimals(uint256 amount)`, `function getToken1AmountInWei(uint256 amount)`, and `function getToken1AmountInNativeDecimals(uint256 amount)`.

Unexpected result if the decimal of token1 is is larger than the decimal of token0

High
Status: Fixed.
This is due to the fact that the `tokenDiffDecimalMultiplier` is computed based on `Utils.subAbs` function.

And everywhere in the code; for instance, `contracts/xU3LPStable.sol`: `L953` has an implicit assumption that the decimal of token0 is larger than the decimal of token1.
Recommendation:
Revert the function call of function `initialize` whenever found that the decimal of token1 is larger than the decimal of token0.

Misuse of amount unit when swapping to obtain enough tokens to withdraw

Status: Acknowledged, xToken team stated that this is intended, in order to maintain the tendency of the contracts to hold 50:50 token0 and token1.
High
1. `contracts\xU3LPStable.sol`: `L495`: should convert the unit of `amountIn` from `token0` to `token1`.
2. `contracts\xU3LPStable.sol`: `L500`: should minus `amountIn` instead of `amountOut`.
3. `contracts\xU3LPStable.sol`: `L509`: should convert the unit of `amountIn` from `token1` to `token0`.
4. `contracts\xU3LPStable.sol`: `L513`: should minus `amountIn` instead of `amountOut`.

Misuse of amount unit when minting XU3LP token(1)

High
Status: Acknowledged, xToken team stated that this is intended, in order to maintain the tendency of the contracts to hold 50:50 token0 and token1.
`contracts\xU3LPStable.sol`: `L136`: should be `_mintInternal(amount.sub(fee));` instead of `_mintInternal(getAmountInAsset1Terms(amount).sub(fee));` because function `calculateMintAmount` is implemented in terms of token0's amount.

Misuse of amount unit when burning XU3LP token(2)

High
Status: Acknowledged, xToken team stated that this is intended, in order to maintain the tendency of the contracts to hold 50:50 token0 and token1.
`contracts\xU3LPStable.sol`: `L154`: since `token0` is the `outputAsset`, `getAmountInAsset0Terms` part in this line should be removed because the resulting amount, without this function, is already `token0` amount. No need to further convert it into `token1` amount.

Constants used for generating a fixed point `1.0001` is misplaced

Medium
Status: Fixed
`contracts\libraries\Utils.sol`: `L28`: should be `.divu(10001, 10000)` instead.

There is no backup oracle nor protection from erroneous price data

Medium
Status: Fixed by adding a sanity check of deviation of 1% price change from latest price.
The oracle data from UniswapV3 that the current system uses does not have any backup currently. The system could fail to work correctly when the oracle is operating abnormally or being manipulated.
Recommendation:
Please add more than one oracle in order to increase the security level. Also, consider adding some sanity checks to the collected price data. At least add an emergent transaction revert implementation whenever the price data collected from UniswapV3 is far from being normal.

The target balanced state is achieved based on amount of underlying tokens instead of the value of underlying tokens

Informative
Status: Mitigated since the ratio of token0 and token1 will be close to 1.
The target balanced state is achieved based on the amount of underlying tokens instead of the value of underlying tokens. There are two places in the contract that show this design choice, they are, function `transferOnBurn` and function `getTargetBufferTokenBalance`. This may lead to more swapping operations when the difference between the values of the underlying tokens is large.
Recommendation:
Rebalance the amounts of the underlying tokens based on the value of them.

End user might be delayed from withdrawing a large amount of tokens by burning XU3LP token even when there is sufficient LP tokens hold by the contract

Status: Acknowledged, xToken team stated that they will put it in blog post and on dapp where users invest.
Low
`contracts\xU3LPStable.sol`: `L168`-`L171`: an end user might not be able to burn XU3LP token

when the contract does not hold enough underlying tokens. This is true even when there are sufficient LP tokens hold by the contract. Although as time passes, the contract would unstake the underlying tokens from UniswapV3 contracts and have more liquidity for the user to withdraw, this possible delay in withdrawing their fund should be known by the user.
Recommendation: It is recommended to make this information clear to the users in public documents.

Privileged roles

Status: Acknowledged, xToken team stated that they will put it in blog post and on dapp where users invest.
Severity: Informational
Description:

There are some actions that could have important consequences for end-users. The owner or the managers of the `contracts\xU3LPStable.sol` can:
1. Call function `withdrawToken` at any point in time, to withdraw any amount of the underlying tokens from the contract.
2. Call function `adminStake` and `adminUnstake` to manipulate the staking status of the contract.
3. Call function `adminSwap` to swap any amount of tokens between `token0` and `token1`.
4. Call function `pauseContract` and function `unpauseContract` to pause or unpause the contract.

Recommendation:
This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Unlock pragma

Status: Fixed
Severity: Informational
Description: Every Solidity file specifies in the header a version number of the format pragma solidity (^)0.8.*. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version and above, hence the term "unlocked".
Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version for anchor-eth-contracts/contracts/*.

Documentation

`contracts\xU3LPStable.sol`: `L608`-`L610`: those comments do not align with the code implementation
Status: Fixed

Adhere to Specification

* For commit `65ecb28`, `contracts/libraries/Utils.sol::calculateSwapAmount`: the `subAbs` is different from the code comment, and never give the sign back later in the code, please make sure if this is intended. If it is, correct the code comment.
Status: Fixed

Best Practice

* ` contracts\xU3LPStable.sol`: function `mintInitial`: consider calling this function at the end of `function initialize` to avoid human error in the deployment process.
Status: Acknowledge, xToken team: 'we will probably chain the calls, or at least call mintInitial immediately after initialise in the deployment script."

* Based on the design in `contracts\xU3LPStable.sol`:`L859`, this contract is designed for stable coin trading pairs only. The xToken team should make sure that the chosen trading pairs should always satisfy the assumption that the ratio is near to `1`.
Status: Acknowledged, xToken team: 'All of the pairs in the first two waves satisfy that assumption.'