Lyra SNX PerpsV2 Hedger and Adapter Smart Contract Audit

Lyra, 11 May 2023

Confidential
17 Tyrwhitt Avenue, Johannesburg, 2196, South Africa
hello@iosiro.com

1. Introduction

iosiro was commissioned by Lyra (https://www.lyra.finance/) to conduct a smart contract audit of Lyra's SNX PerpsV2 Hedger smart contracts. The audit was performed by two auditors between 3 and 21 April 2023 using 22 resource days. A retest and final review of changes was conducted on 8 May 2023.

This report is organized into the following sections.

- **Section 2 - Executive summary:** A high-level description of the findings of the audit.
- **Section 3 - Audit details:** A description of the scope and methodology of the audit.
- **Section 4 - Design specification:** An outline of the intended functionality of the smart contracts.
- **Section 5 - Detailed findings:** Detailed descriptions of the findings of the audit.

The information in this report should be used to understand the smart contracts' risk exposure better and as a guide to improving the security posture of the smart contracts by remediating the issues identified. The results of this audit reflect the in-scope source code reviewed at the time of the audit.

The purpose of this audit was to achieve the following:

- Identify potential security flaws.
- Ensure that the smart contracts function according to the documentation provided.

Assessing the off-chain functionality associated with the contracts, for example, backend web application code, was outside of the scope of this audit.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered high risk from cyber attacks. As such, the highest level of security should be observed when interacting with these assets. This requires a forward-thinking approach, which takes into account the new and experimental nature of blockchain technologies. Strategies that should be used to encourage secure code development include:

- Security should be integrated into the development lifecycle, and the level of perceived security should not be limited to a single code audit.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed where possible.

2. Executive summary

This report presents the findings of an audit performed by iosiro on specific peripheral smart contracts for the Lyra Protocol.

Audit findings

The smart contract audit identified several issues across risk levels, including high, medium, and low-risk findings. High-risk findings were related to inconsistent decimals, breaking changes in an upcoming release of Synthetix, and low liquidity in a particular UniswapV3 pool. Medium-risk findings concerned flaws in various functions that may lead to inaccurate risk assessments and calculation errors. Additionally, the use of a hard-coded value that should be configurable was raised as a low-risk issue.

The audit also provided several informational design improvements to enhance the smart contracts' readability, gas efficiency and reduce the protocol's attack surface.

By the conclusion of the retest all high, medium and low risk issues were suitably addressed or remediated.

Recommendations

At a high level, the security posture of the Lyra Protocol could be further strengthened by:

- Performing additional audits at regular intervals, as security best practices, tools, and knowledge change over time. Additional audits throughout the project's lifespan ensure the longevity of the codebase.
- Continue their bug bounty program encouraging the responsible disclosure of security vulnerabilities in the system.

3. Audit details

3.1 Scope

The source code considered in-scope for the assessment is described below. Code from all other files was considered to be out-of-scope. Out-of-scope code that interacts with in-scope code was assumed to function as intended and not introduce any functional or security vulnerabilities for the purposes of this audit.

3.1.1 Smart contracts

Project name: SNX PerpsV2 Integration
Commits: 929eb44, 98eacc0
Final Commit: 366d188
Files: SNXPerpV2Adapter.sol, SNXPerpsV2PoolHedger.sol, UniswapV3Oracle.sol

3.2 Methodology

The audit was conducted using a variety of techniques described below.

3.2.1 Code review

The source code was manually inspected to identify potential security flaws. Code review is a useful approach for detecting security flaws, discrepancies between the specification and implementation, design improvements, and high-risk areas of the system.

3.2.2 Dynamic analysis

The contracts were compiled, deployed, and tested in a test environment, both manually and through the test suite provided. Manual analysis was used to confirm that the code was functional and discover security issues that could be exploited.

3.2.3 Automated analysis

Tools were used to automatically detect the presence of several types of security vulnerabilities, including reentrancy, timestamp dependency bugs, and transaction-ordering dependency bugs. Static analysis results were reviewed manually and any false positives were removed. Any true positive results are included in this report.

Static analysis tools commonly used include Slither, Securify, and MythX. Tools such as the Remix IDE, compilation output, and linters could also be used to identify potential areas of concern.

3.3 Risk ratings

Each issue identified during the audit has been assigned a risk rating. The rating is determined based on the criteria outlined below.

- **High risk:** The issue could result in a loss of funds for the contract owner or system users.
- **Medium risk:** The issue resulted in the code specification being implemented incorrectly.
- **Low risk:** A best practice or design issue that could affect the security of the contract.
- **Informational:** A lapse in best practice or a suboptimal design pattern that has a minimal risk of affecting the security of the contract.
- **Closed:** The issue was identified during the audit and has since been satisfactorily addressed, removing the risk it posed.

4. Design specification

The following section outlines the intended functionality of the system at a high level. This specification is based on the implementation in the codebase. Any perceived points of conflict should be highlighted with the auditing team to determine the source of the discrepancy.

4.1 SNX PerpsV2 Adapter

The contract enables the options market to perform and estimate exchanges between quote and base assets using Uniswap V3 Pools. The slippage of each exchange is regulated by validating the parameters against the exchange rate reported by Chainlink's data feeds.

Synthetix is used instead of directly querying the Chainlink aggregator. This approach provides extra validation on the prices returned and ensures that all prices are normalized to 18 decimal places.

The adapter defines all parameters and return values using 18 decimal places to abstract away the underlying assets' complexity.

4.2 SNX Perps V2 Hedger

As Lyra Markets engages in selling options, it exposes the liquidity providers to the price movements of the underlying assets. To mitigate this exposure, the markets open proportional positions using their associated Synthetix PerpsV2 markets.

The hedger employs delayed orders to minimize fees when trading on the SNX PerpsV2 market. This introduces additional complexity, as orders must be submitted and executed subsequently.

The Risk Council controls several risk parameters of the hedger, such as the maximum leverage of the position and minimum order size. A minimal interaction delay and order size are enforced to prevent sandwich attacks and abuse of the hedger.

Lyra's Newport release uses USDC as the default quote asset and must therefore perform exchanges between USDC and sUSD. The hedger exchanges the tokens

through the Curve sUSD/3CRV pool via the Curve Pool Registry. A worst acceptable exchange rate is configurable by the Risk Council to mitigate sandwich attacks.

5. Detailed findings

The following section details the findings of the audit.

5.1 High risk

No high-risk issues identified during the audit were present at the conclusion of the audit.

5.2 Medium risk

No medium-risk issues identified during the audit were present at the conclusion of the audit.

5.3 Low risk

No low-risk issues identified during the audit were present at the conclusion of the audit.

5.4 Informational risk

5.4.1 Design comments

Actions to improve the functionality and readability of the codebase are outlined below.

Rename `uniDeviation`

SNXPerpV2Adapter.sol#L39

`uniDeviation` could be renamed to be more descriptive, for example, `maxSpotPriceDeviationRatio`. It might also be better to have this variable configurable per market by storing it in the `uniswapInfo` struct for each market.

Avoid maximum approvals

SNXPerpV2Adapter.sol#L105
SNXPerpsV2PoolHedger.sol#L134

Performing maximum approvals for the relevant Uniswap and Curve contracts unnecessarily increases the attack surface of the adapter and pool hedger. It is recommended that approvals be set before exchanges and nullified thereafter.

Perform balance checks before and after asset exchanges

SNXPerpV2Adapter.sol#L387
SNXPerpV2Adapter.sol#L342
SNXPerpsV2PoolHedger.sol#L568
SNXPerpsV2PoolHedger.sol#L592

The minimum or exact output parameters of the Uniswap and Curve pools are assumed to function as expected. While this is likely a sound assumption, the contract balances should be checked before and after trades, as a defense-in-depth strategy.

Decimal consistency

[OptionMarket.sol#L189]

`baseLimit` is specified in terms of the base asset's decimal and not 18 decimals as used through the protocol.

To ensure consistent use of 18 decimals, it is recommended that the `baseLimit` is configured and stored using 18 decimals and converted as needed:

```
--- a/contracts/OptionMarket.sol
+++ b/contracts/OptionMarket.sol
@@ -989,7 +989,7 @@ contract OptionMarket is Owned, SimpleInitializable,
ReentrancyGuard {
       // add base limit here to prevent trades that are above the base limit
       uint shortCollateralBalance =
baseAsset.balanceOf(address(shortCollateral));

-      if (shortCollateralBalance + pendingCollateralConverted > baseLimit) {
+      if (shortCollateralBalance + pendingCollateralConverted >
ConvertDecimals.convertFrom18(baseLimit, baseAsset.decimals())) {
              revert BaseLimitExceeded(
              address(this),
              msg.sender
```

Update: The Lyra development team acknowledged the issue during the audit, but decided to not normalize `baseLimit` to 18 decimals as it would simplify Governance proposals.

5.5 Closed

5.5.1 Inconsistent decimals in `SNXPerpV2Adapter` (high risk)

SNXPerpV2Adapter.sol#L259
SNXPerpV2Adapter.sol#L283
SNXPerpV2Adapter.sol#L319

Description

The BaseAdapter interface, which the SNXPerpV2Adapter implements, defines all `uint` parameters as having 18 decimals. Testing showed that decimals used in `exchangeFromExactBase()`, `exchangeToExactBaseWithLimit()`, and `estimateExchangeToExactQuote()` were not correct for all token pairs.

In most cases, this would prevent users from buying options from the Option Market, as the underlying token exchange would revert. In the worst-case scenario, it could lead to significant losses for the protocol and users.

Recommendation

Functions in the SNXPerpV2Adapter contract should be standardized and consistent for all quote/base pairs.

Additionally, the adapter interface parameters should clearly indicate the expected decimal values to remove ambiguity. Using a straightforward naming convention for the adapter interface parameters is recommended, such as `baseAmountD18`, `quoteAmountD18`, `baseAmountAsset`, `quoteAmountAsset`, etc.

The code should be reviewed and tested thoroughly to ensure the issue is resolved.

Update

The issues in the affected functions were resolved per the recommendations in commit 98eacc0.

5.5.2 Breaking changes in `submitOffchainDelayedOrderWithTracking()` (high risk)

SNXPerpsV2PoolHedger.sol#L486

Description

At the time of writing, the next release of the Synthetix Protocol (Calph 2.86.0) is set to introduce a breaking change to the submitOffchainDelayedOrderWithTracking() function. Once deployed, the integration between the hedger and the PerpsV2 markets could result in failed transactions or fill prices.

The release will replace the `priceImpactDelta` parameter with a new `desiredFillPrice` parameter of the same type. However, the behavior of the parameters differ significantly. Where `priceImpactDelta` is expressed as a percentage, `desiredFillPrice` is the worst price acceptable at order execution.

Recommendation

Logic to satisfy the new `desiredFillPrice` parameter should be introduced. It is important to note that the ideal `desiredFillPrice` may be higher or lower than the spot price, based on the direction of the trade.

Update

The issue was resolved in commit 98eacc0.

5.5.3 Low WBTC/USDC UniswapV3 Liquidity (high risk)

UniswapV3 WBTC/USDC Pool Information

Description

At the time of writing, the liquidity and volume of the WBTC/USDC Uniswap V3 pool on Optimism were considered extremely low. This would enable an attacker to manipulate the Time Weighted Average Price (TWAP) reported by the pool and cause unexpected reverts. It would also be trivial to sandwich trades performed by the protocol.

Recommendation

For token pairs with limited liquidity, it might be more effective to implement a multi-hop trade and TWAP. This is expected to increase the complexity of trades significantly.

For example, exchanging from WBTC to USDC could be facilitated through the WBTC / WETH and WETH / USDC pools.

At the time of writing, the volume and liquidity of these pools were multiple orders of magnitude greater than that of the WBTC / USDC pool.

Update

The issue was mitigated by including a mechanism to limit the amount of Base Asset to be kept as short collateral. The team indicated that the long running average liquidity for the relevant Uniswap Pairs will be used to determine the limit per market. The `baseLimit` parameter and logic was introduced in commit 98eacc0.

Furthermore, the additional deviation checks using the TWAP of the Uniswap Pools were also removed in commit 98eacc0. It was concluded that the TWAP provided little benefit above the validations already performed on the asset prices returned by Synthetix.

5.5.4 `canHedge()` fails to always consider market depth (medium risk)

[SNXPerpsV2PoolHedger.sol#L255]

Description

The `canHedge()` function returns `true` without considering the market depth when the `expectedHedge` and `currentHedge` are on opposite sides. This can result in failed orders.

For reference, the if statement that returns early is given below:

```
if (Math.abs(expectedHedge) <= Math.abs(currentHedge)) {
  // Delta is shrinking (potentially flipping, but still smaller than current
hedge), so we skip the check
  return true;
}
```

Recommendation

The if statement should be updated to only return early and bypass the remaining validations in `canHedge()` if the `expectedHedge` and `currentHedge` are of the same sign.

Update

The issue was resolved as per the recommendation in commit 98eacc0.

5.5.5 Incorrect assumption in `_addFeeToMargin()` (medium risk)

SNXPerpsV2PoolHedger.sol#L384

Description

`_addFeeToMargin()` assumes that the fee returned from `perpsMarket.orderFee()` includes the keeper fee deposit when submitting an off-chain order. Testing revealed that this is not the case and could result in `hedgeDelta()` using an incorrect `minInitialMargin` and reverting.

Recommendation

The keeper fee should be added to `minInitialMargin` to ensure sufficient margin is transferred to the position before submitting the off-chain order.

Update

The `minKeeperFee` was added to the `minInitialMargin` as per the recommendation in commit 98eacc0.

5.5.6 Division by zero in `estimateExchangeToExactQuote()` (medium risk)

SNXPerpV2Adapter.sol#L273

Description

In `estimateExchangeToExactQuote()`, if there is no Uniswap pool configured, `_getUniswapPrice()` will return zero and result in a division by zero. This is because the function uses the lowest price between `_getUniswapPrice()` and `_getSNXPerpV2Price()`.

Recommendation

A check should be added similar to `estimateExchangeToExactBase()`, whereby the SNX price is used if the Uniswap price is zero.

Update

The issue was addressed per the recommendation in commit 98eacc0.

5.5.6 Hard-coded Uniswap deviation (low risk)

SNXPerpV2Adapter.sol#L362

Description

The Uniswap deviation in `exchangeToExactBaseWithLimit()` is hard coded to a value of `1.03 ether`, which means that the value would not get updated when the state variable `uniDeviation` is updated.

Using a hard-coded value would prevent the protocol from managing its slippage tolerance, which could result in losses if the trades were susceptible to sandwich attacks.

It should be noted that a 3% of slippage could be significant and result in significant losses for the protocol. A more stringent slippage tolerance should be supported, such as 0.5%.

Recommendation

The state variable `uniDeviation` should replace the hard-coded value.

Update

The hard-coded value was replaced as per the recommendation in commit 98eacc0.