OCTOBER 2021

PRIMITIVE V2

Audit Report

Security Auditor

SHW

Introduction

Primitive is an options market platform that allows users to create new option types, to then mint, exercise, or close them during their entire life cycle

SUMMARY

* Protocol: **Primitive Finance V2**

* Scope: **Smart contract in the following repos, excluding the tests**

* Repo: `**primitive-v2-core**`

  - Branch: **`main`**

  - Commit: `**0d7a078bcfcab42dddfcf2830ad0f93c54362251`**

* Repo: **`primitive-v2-periphery`**

  - Branch: `**develop`**

  - Commit: `**06c27b1386688162ee4fcc08cb3ecd2ab9e3d1c0`**

* Publish Date: **10/30/2021**

FINDINGS

CRITICAL SEVERITY

No critical-severity issues.

HIGH SEVERITY

No high-severity issues.

MEDIUM SEVERITY

No medium-severity issues.

LOW SEVERITY

[L-01] Incorrect Event Parameter

In the internal `deploy` function of `PrimitiveFactory`, the following line of code

```
if (stableDecimals > 18 || stableDecimals < 6) revert DecimalsError(riskyDecimals);
```

It should be

```
if (stableDecimals > 18 || stableDecimals < 6) revert DecimalsError(stableDecimals);
```

since it is the `stableDecimals` variable that has an invalid value.

[L-02] Sanity Checks on the Engine Address

The `deposit` function of `MarginManager` does not check whether the engine exists before calling its functions, which causes the transaction to revert (since no code on the address) without an explicit error message. Consider adding an explicit check before calling the methods of the engine. The same issue also exists in the `withdraw` function of `MarginManager` and the `swap` function of `SwapManager`.

[L-03] Error Message of SafeApprove

The `safeApprove` function of the `TransferHelper` library throws an error of `TransferError` when the `approve` function call fails. According to the code comments, the `TransferError`

> **@notice** Thrown when a transfer reverts

Since the `approve` call is unrelated to a token transfer, it may be reasonable to use another error message instead, e.g., `ApproveError`.

[L-04] Uninitialized Cache of Engine

The `PositionManager` contract relies on a call to `_allocate` (done by `PrimitiveHouse`) to initialize the cache of a `poolId` to its engine. The view functions, e.g., `getMetadata`, generate information strings of the `poolId` based on the cache. However, if a pool is created directly through the engine instead of `PrimitiveHouse`, and users call `allocate` directly from the engine, the cache is never initialized, causing the view functions of `PositionManager` to revert.

[L-05] Valid Range of Pool Creation Parameters

According to the comments of the `SigmaError` error, the `sigma` variable should be strictly greater than 100 and less than 1e7. However, in the `create` function of `PrimitiveEngine`, `sigma` is checked by

if (sigma > 1e7 || sigma < 100) revert SigmaError(sigma);

```which does not include the boundary values.

Besides, as the comment says, the `MinLiquidityError` error is thrown when the liquidity is lower than the minimum amount of liquidity. However, if `delLiquidity` equals `MIN_LIQUIDITY`, a `MinLiquidityError` error is thrown as well.

if (delLiquidity <= MIN_LIQUIDITY) revert MinLiquidityError(delLiquidity);

**[NC-01] Incorrect type of variable used in Event parameter**

In [Create event] cal.strike, cal.sigma, cal.maturity and cal.gamma they are Of uint128, uint64, uint32, uint32 respectively, but in IPrimitiveEngine.sol:13 these parameters are of uint256

**[G-01] Packing of variables**

Gas Optimization
```