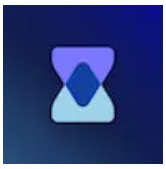

# Timeswap contest
# Findings & Analysis Report

2023-03-10

## Table of contents


- Overview
  - About C4
  - Wardens
- Summary
- Scope
- Severity Criteria
- High Risk Findings (3)
  - [H-01] Rebalance logic is wrong and this distorts the pool's important states
  - [H-02] TimeswapV2LiquidityToken should not use `totalSupply()+1` as tokenId
  - [H-03] The `collect()` function will always TRANSFER ZERO fees, losing _feesPositions without receiving fees!
- Medium Risk Findings (7)
  - [M-01] `_currentIndex` is incorrectly updated; breaking the ERC1155 enumerable implementation

- [M-02] Burning a `ERC1155Enumerable` token doesn't remove it from the enumeration
- [M-03] Fee on transfer tokens will not behave as expected
- [M-04] sqrtDiscriminant can be calculated wrong
- [M-05] unexpected overflow for `FullMath.add512()` which can result in irregular behavior
- [M-06] `_ownedTokensIndex` is SHARED by different owners, as a result, `_removeTokenFromAllTokensEnumeration` might remove the wrong tokenId.
- [M-07] `Mint` function does not update `LiquidityPosition` state of caller before minting LP tokens. This

- Low Risk and Non-Critical Issues

  - 01 User can possibly transfer no `token0` or `token1` to `TimeswapV2Option` contract if corresponding `token0` OR `token1` is a rebasing token
  - 02 `Error.checkEnough` function does not prevent user from sending too many tokens to relevant contract
  - 03 Pool considers option not matured when its maturity and the block timestamp are equal
  - 04 Pool's `ParamLibrary.check` function for `TimeswapV2PoolCollectParam` checks `param.strike == 0` less restrictively than pool's other `ParamLibrary.check` functions
  - 05 Redundant named returns
  - 06 Word/typing typos
  - 07 Confusing NatSpec `@param` usage
  - 08 Incomplete NatSpec Comments
  - 09 Missing NatSpec comments

- Gas Optimizations

  - Gas Optimizations Summary
  - G-01 Gas saving is achieved by removing the `delete` keyword (~60k)

- G-02 Remove `checkDoesNotExist` function
- G-03 Avoid using `state variable` in emit (130 gas)
- G-04 Change `public` state variable visibility to `private`
- G-05 Save gas with the use of the import statement
- G-06 Gas savings can be achieved by changing the model for assigning value to the structure (260 gas)
- G-07 Using `delete` instead of setting struct `0` saves gas
- G-08 In `div 512` function, `quotient 0` aggregate operation is used with unchecked to save gas
- G-09 Avoid using external call
- G-10 Gas overflow during iteration (DoS)
- G-11 Move owner checks to a modifier for gas efficant
- G-12 Use a more recent version of solidity
- G-13 Use nested if and, avoid multiple check combinations
- G-14 Sort Solidity operations using short-circuit mode
- G-15 `>=` costs less gas than `>`
- G-16 Using UniswapV3 `mulDiv` function is gas-optimized
- G-17 Using Openzeppelin Ownable2Step.sol is gas efficient
- G-18 OpenZeppelin's ReentrancyGuard contract is gas-optimized
- G-19 Save gas with the use of the import statement
- G-20 Remove import `forge-std/console.sol`
- G-21 Usage of uints/ints smaller than 32 bytes (256 bits) incurs overhead
- G-22 Use `assembly` to write *address storage values*
- G-23 Setting the *constructor* to `payable`
- G-24 Avoid contract existence checks by using solidity version 0.8.10 or later
- G-25 Optimize names to save gas
- G-26 Upgrade Solidity's optimizer
- G-27 Open the optimizer

- Disclosures

# Overview

## About C4

Code4rena (C4) is an open organization consisting of security researchers, auditors, developers, and individuals with domain expertise in smart contracts.

A C4 audit contest is an event in which community participants, referred to as Wardens, review, audit, or analyze smart contract logic in exchange for a bounty provided by sponsoring projects.

During the audit contest outlined in this document, C4 conducted an analysis of the Timeswap smart contract system written in Solidity. The audit contest took place between January 20—January 27 2023.

## Wardens

59 Wardens contributed reports to the Timeswap contest:

1. 0Kage
2. 0x1f8b
3. 0xAgro
4. 0xGusMcCrae
5. 0xSmartContract
6. 0xackermann
7. 0xcm
8. Awesome
9. Aymen0909
10. Beepidibop
11. Breeje
12. DadeKuma
13. Diana
14. Ellar

15. lllllll
16. lurii3
17. Josiah
18. Moksha
19. Rageur
20. RaymondFam
21. ReyAdmirado
22. Rolezn
23. Ruhum
24. SaeedAlipoor01988
25. Udsen
26. Viktor_Cortess
27. W0RR1O
28. W_Max
29. adriro
30. atharvasama
31. brgltd
32. btk
33. c3phas
34. chaduke
35. codeislight
36. cryptonue
37. ddimitrov22
38. delfin454000
39. descharre
40. eierina
41. fatherOfBlocks
42. georgits
43. gerdusx

44. hansfriese
45. kaden
46. lukris02
47. luxartvinsec
48. martin
49. matrix_0wl
50. mert_eren
51. mookimgo
52. nadin
53. oberon
54. pavankv
55. popular00
56. rbserver
57. shark
58. sorrynotsorry
59. tnevler

This contest was judged by Picodes.

Final report assembled by itsmetechjay.

## Summary

The C4 analysis yielded an aggregated total of 10 unique vulnerabilities. Of these vulnerabilities, 3 received a risk rating in the category of HIGH severity and 7 received a risk rating in the category of MEDIUM severity.

Additionally, C4 analysis included 40 reports detailing issues with a risk rating of LOW severity or non-critical. There were also 24 reports recommending gas optimizations.

All of the issues presented here are linked back to their original finding.

## Scope

The code under review can be found within the C4 Timeswap contest repository, and is composed of 70 smart contracts written in the Solidity programming language and includes 3,605 lines of Solidity code.

## Severity Criteria

C4 assesses the severity of disclosed vulnerabilities based on three primary risk categories: high, medium, and low/non-critical.

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

For more information regarding the severity criteria referenced throughout the submission review process, please refer to the documentation provided on the C4 website, specifically our section on Severity Categorization.

## High Risk Findings (3)

### [H-01] Rebalance logic is wrong and this distorts the pool's important states

*Submitted by* hansfriese

The important states including `long0Balance, long1Balance, long1FeeGrowth, long1ProtocolFees` are wrongly calculated and it breaks the pool's invariant.

#### Proof of Concept

The protocol provides a rebalancing functionality and the main logic is implemented in the library `Pool.sol`. If `param.isLong0ToLong1` is true and the transaction is `TimeswapV2PoolRebalance.GivenLong1`, the protocol calculates the

`long1AmountAdjustFees` first and the actual `long0Amount, longFees` and the final `long1Balance` is decided accordingly.

The problem is it is using the wrong parameter `pool.long0Balance` while it is supposed to use `pool.long1Balance` in the line L679.

This leads to wrong state calculation in the following logic. (especially L685 is setting the `long1Balance` to zero).

Furthermore, the protocol is designed as a permission-less one and anyone can call `TimeswapV2Pool.rebalance()`.

An attacker can abuse this to break the pool's invariant and take profit leveraging that.

```
packages\v2-pool\src\structs\Pool.sol
665:     function rebalance(Pool storage pool, TimeswapV2PoolReb
666:         if (pool.liquidity == 0) Error.requireLiquidity();
667:
668:         // No need to update short fee growth.
669:
670:         uint256 longFees;
671:         if (param.isLong0ToLong1) {
672:             if (param.transaction == TimeswapV2PoolRebalanc
673:                 (long1Amount, longFees) = ConstantSum.calcu
674:
675:                 if (long1Amount == 0) Error.zeroOutput();
676:
677:                 pool.long1Balance -= (long1Amount + longFee
678:             } else if (param.transaction == TimeswapV2PoolR
    //**********************************************************
679:                 uint256 long1AmountAdjustFees = FeeCalculat
    //**********************************************************
680:
681:                 if ((long1Amount = param.delta) == long1Amo
682:                     long0Amount = ConstantSum.calculateGive
683:
684:                     longFees = pool.long1Balance.unsafeSub(
685:                     pool.long1Balance = 0;
686:                 } else {
687:                     (long0Amount, longFees) = ConstantSum.c
688:
689:                     pool.long1Balance -= (long1Amount + lon
```

```
690:                 }
691:
692:                 if (long0Amount == 0) Error.zeroOutput();
693:             }
694:
695:             pool.long0Balance += long0Amount;
696:
697:             (pool.long1FeeGrowth, pool.long1ProtocolFees) =
698:         } else {
699:             if (param.transaction == TimeswapV2PoolRebalanc
700:                 uint256 long0AmountAdjustFees = FeeCalculat
701:
702:                 if ((long0Amount = param.delta) == long0Amo
703:                     long1Amount = ConstantSum.calculateGive
704:
705:                     longFees = pool.long0Balance.unsafeSub(
706:                     pool.long0Balance = 0;
707:                 } else {
708:                     (long1Amount, longFees) = ConstantSum.c
709:
710:                     pool.long0Balance -= (long0Amount + lon
711:                 }
712:
713:                 if (long1Amount == 0) Error.zeroOutput();
714:             } else if (param.transaction == TimeswapV2PoolR
715:                 (long0Amount, longFees) = ConstantSum.calcu
716:
717:                 if (long0Amount == 0) Error.zeroOutput();
718:
719:                 pool.long0Balance -= (long0Amount + longFee
720:             }
721:
722:             pool.long1Balance += long1Amount;
723:
724:             (pool.long0FeeGrowth, pool.long0ProtocolFees) =
725:         }
726:     }
```

## Recommended Mitigation Steps

Fix the L679 as below.

```
uint256 long1AmountAdjustFees = FeeCalculation.removeFees(pool.l
```

vhawk19 (Timeswap) confirmed and resolved:

> Fixed here at this commit.

## [H-02] TimeswapV2LiquidityToken should not use `totalSupply()+1` as tokenId

*Submitted by* mookimgo*, also found by* hansfriese

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2LiquidityToken.sol#L114

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2Token.sol#L103

### Impact

Assuming ERC1155Enumerable is acting normally, there is an **Accounting Issue** about TimeswapV2LiquidityToken and TimeswapV2Token's tokenId.

Different liquidities can have the same `tokenId`, leading to serious balance manipulation.

I'm submitting this issue as medium because current implementation ERC1155Enumerable is wrong, which exactly mitigates this issue making it not exploitable. But this issue will become dangerous once we fixed ERC1155Enumerable.

### Proof of Concept

In this PoC, the attacker will do these steps:

1. Add liquidity of token0 and token1, thus receiving TimeswapV2LiquidityToken tokenId 1.
2. Add liquidity of token2 and token3, thus receiving TimeswapV2LiquidityToken tokenId 2.
3. Burn his liquidity from step1, which will make totalSupply decrease (if ERC1155Enumerable has been patched).

4. Add liquidity of token4 and token5, and receive TimeswapV2LiquidityToken tokenId 2. This is wrong tokenId, which should be 3.

Explanation:

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2LiquidityToken.sol#L112

As the comment said, `if the position does not exist, create it` , but the new tokenId is set as `totalSupply() + 1` .

Function totalSupply is defined in `packages/v2-token/src/base/ERC1155Enumerable.sol` , which is simply `\_allTokens.length` : https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/base/ERC1155Enumerable.sol#L37-L38

`\_allTokens.length` can be decreased in `_removeTokenFromAllTokensEnumeration` function, which is called by `_removeTokenEnumeration` , and by `_afterTokenTransfer` . In simple words, when all token amounts for a specific tokenId are burned ( `_idTotalSupply[id] == 0` ), totalSupply should be decreased.

Current implementation of ERC1155Enumerable has a bug, which will never trigger `_removeTokenFromAllTokensEnumeration` : Calling `\_removeTokenEnumeration` needs `amount>0` , but only `_idTotalSupply[id] == 0` can trigger `\_removeTokenFromAllTokensEnumeration` .

```
        function _removeTokenEnumeration(address from, address to, u
            if (to == address(0)) {
                if (_idTotalSupply[id] == 0 && _additionalConditionR
                _idTotalSupply[id] -= amount;
            }
```

Once the above code gets fixed (swapping the if line and `_idTotalSupply[id] -= amount;` line, patch given below), this issue becomes exploitable, making the accounting of LP wrong.

Proof of Concept steps:

First, we need to patch two contracts:

- making TimeswapV2LiquidityToken's `\_timeswapV2LiquidityTokenPositionIds` as public for testing, this can be removed when depolying
- ERC1155Enumerable's `\_removeTokenEnumeration` has been patched to behave correctly, which will decrease `totalSupply` when all token amount of a specific tokenId has been burned.

```
diff --git a/packages/v2-token/src/TimeswapV2LiquidityToken.sol b
index 2f71a25..f3910d9 100644
--- a/packages/v2-token/src/TimeswapV2LiquidityToken.sol
+++ b/packages/v2-token/src/TimeswapV2LiquidityToken.sol
@@ -42,7 +42,7 @@ contract TimeswapV2LiquidityToken is ITimeswapV

     mapping(uint256 => TimeswapV2LiquidityTokenPosition) private

-    mapping(bytes32 => uint256) private _timeswapV2LiquidityToke
+    mapping(bytes32 => uint256) public _timeswapV2LiquidityToken

     mapping(uint256 => mapping(address => FeesPosition)) private

diff --git a/packages/v2-token/src/base/ERC1155Enumerable.sol b/p
index 4ec23ff..4f51fb4 100644
--- a/packages/v2-token/src/base/ERC1155Enumerable.sol
+++ b/packages/v2-token/src/base/ERC1155Enumerable.sol
@@ -91,8 +91,8 @@ abstract contract ERC1155Enumerable is IERC1155
     /// @dev Remove token enumeration list if necessary.
     function _removeTokenEnumeration(address from, address to,
         if (to == address(0)) {
-            if (_idTotalSupply[id] == 0 && _additionalCondition(
             _idTotalSupply[id] -= amount;
+            if (_idTotalSupply[id] == 0 && _additionalCondition(
         }

         if (from != address(0) && from != to) {
```

Add a new test file in `2023-01-timeswap/packages/v2-token/test/TimeswapV2LiquidityToken_MultiMint.t.sol`:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity =0.8.8;

import "forge-std/Test.sol";

import "forge-std/console.sol";

import "../src/TimeswapV2LiquidityToken.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@timeswap-labs/v2-option/src/TimeswapV2OptionFactory.sol
import "@timeswap-labs/v2-option/src/interfaces/ITimeswapV2Optio
import {TimeswapV2LiquidityTokenCollectParam} from "../src/struc
import "@timeswap-labs/v2-pool/src/TimeswapV2PoolFactory.sol";
import "@timeswap-labs/v2-pool/src/interfaces/ITimeswapV2Pool.so
import {TimeswapV2PoolMintParam} from "@timeswap-labs/v2-pool/sr
import {TimeswapV2PoolMintChoiceCallbackParam, TimeswapV2PoolMin

import {TimeswapV2OptionMintCallbackParam, TimeswapV2OptionSwapC

// import "@timeswap-labs/v2-option/src/TimeswapV2OptionFactory.
// // import "@timeswap-labs/v2-option/src/interfaces/ITimeswapV
import "@openzeppelin/contracts/token/ERC1155/utils/ERC1155Holde
import {TimeswapV2LiquidityTokenPosition, PositionLibrary} from
import {TimeswapV2PoolMint} from "@timeswap-labs/v2-pool/src/enu
import {TimeswapV2OptionMint} from "@timeswap-labs/v2-option/src

import {StrikeConversion} from "@timeswap-labs/v2-library/src/St
import {DurationCalculation} from "@timeswap-labs/v2-pool/src/li
import {FullMath} from "@timeswap-labs/v2-library/src/FullMath.s

contract HelperERC20 is ERC20 {
    constructor(string memory _name, string memory _symbol) ERC2
        _mint(msg.sender, type(uint256).max);
    }
}
struct Timestamps {
    uint256 maturity;
    uint256 timeNow;
}
struct MintOutput {
    uint160 liquidityAmount;
    uint256 long0Amount;
    uint256 long1Amount;
    uint256 shortAmount;
    bytes data;
}
```

```
contract TimeswapV2LiquidityTokenTest is Test, ERC1155Holder {
    ITimeswapV2Option opPair;
    ITimeswapV2Option opPair2;
    ITimeswapV2Option opPair3;
    ITimeswapV2Option opPairCurrent;
    TimeswapV2OptionFactory optionFactory;
    TimeswapV2PoolFactory poolFactory;
    ITimeswapV2Pool pool;
    ITimeswapV2Pool pool2;
    ITimeswapV2Pool pool3;
    ITimeswapV2Pool poolCurrent;
    using PositionLibrary for TimeswapV2LiquidityTokenPosition;

    uint256 chosenTransactionFee = 5;
    uint256 chosenProtocolFee = 4;

    HelperERC20 token0;
    HelperERC20 token1;
    HelperERC20 token2;
    HelperERC20 token3;
    HelperERC20 token4;
    HelperERC20 token5;
    HelperERC20 token0Current;
    HelperERC20 token1Current;
    TimeswapV2LiquidityToken mockLiquidityToken;

    function timeswapV2PoolMintChoiceCallback(TimeswapV2PoolMint
        vm.assume(param.longAmount < (1 << 127));
        long0Amount = StrikeConversion.turn(param.longAmount / 2
        long1Amount = StrikeConversion.turn(param.longAmount / 2
        vm.assume(
            param.longAmount < StrikeConversion.combine(long0Amo
        );
    }

    function timeswapV2PoolMintCallback(TimeswapV2PoolMintCallba
        // have to transfer param.long0Amount, param.long1Amount
        console.log(param.long0Amount, param.long1Amount);
        TimeswapV2OptionMintParam memory mparam = TimeswapV2Opti
            strike: param.strike,
            maturity: param.maturity,
            long0To: msg.sender,
            long1To: msg.sender,
            shortTo: msg.sender,
            transaction: TimeswapV2OptionMint.GivenTokensAndLong
```

```
            amount0: param.long0Amount,
            amount1: param.long1Amount,
            data: ""
        });
        opPairCurrent.mint(mparam);
        console.log("opPair mint ok");
    }

    function timeswapV2OptionMintCallback(TimeswapV2OptionMintCa
        data = param.data;
        //console.log("token0 bal:", token0.balanceOf(address(th
        //console.log("token1 bal:", token1.balanceOf(address(th
        token0Current.transfer(msg.sender, param.token0AndLong0A
        token1Current.transfer(msg.sender, param.token1AndLong1A
    }

    function timeswapV2LiquidityTokenMintCallback(TimeswapV2Liqu
        TimeswapV2PoolMintParam memory param1 = TimeswapV2PoolMi
            strike: param.strike,
            maturity: param.maturity,
            to: address(this),
            transaction: TimeswapV2PoolMint.GivenLiquidity,
            delta: param.liquidityAmount,
            data: ""}
        );

        poolCurrent.mint(param1);
        poolCurrent.transferLiquidity(param.strike, param.maturi
        data = bytes("");
    }

    function setUp() public {
        optionFactory = new TimeswapV2OptionFactory();
        token0 = new HelperERC20("Token A", "A");
        token1 = new HelperERC20("Token B", "B");
        token2 = new HelperERC20("Token C", "C");
        token3 = new HelperERC20("Token D", "D");
        token4 = new HelperERC20("Token E", "E");
        token5 = new HelperERC20("Token F", "F");
        if (address(token1) < address(token0)) {
            (token0, token1) = (token1, token0);
        }
        if (address(token3) < address(token2)) {
            (token2, token3) = (token3, token2);
        }
        if (address(token5) < address(token4)) {
```

```
            (token4, token5) = (token5, token4);
        }
        address opAddress = optionFactory.create(address(token0)
        opPair = ITimeswapV2Option(opAddress);
        address opAddress2 = optionFactory.create(address(token2
        opPair2 = ITimeswapV2Option(opAddress2);
        address opAddress3 = optionFactory.create(address(token4
        opPair3 = ITimeswapV2Option(opAddress3);
        poolFactory = new TimeswapV2PoolFactory(address(this), cl
        pool = ITimeswapV2Pool(poolFactory.create(opAddress));
        pool2 = ITimeswapV2Pool(poolFactory.create(opAddress2));
        pool3 = ITimeswapV2Pool(poolFactory.create(opAddress3));
        mockLiquidityToken = new TimeswapV2LiquidityToken(addres
    }

    function testMint(uint256 strike, uint160 amt, uint256 matur
        setUp();

        // vm.assume(strike != 0 && (maturity < type(uint96).max
        vm.assume(to != address(0));
        vm.assume(
            maturity < type(uint96).max &&
                amt < type(uint160).max &&
                amt != 0 &&
                to != address(0) &&
                strike != 0 &&
                maturity > block.timestamp &&
                maturity > 10000 && rate>0
        );

        console.log("init");
        pool.initialize(strike, maturity, rate);
        pool2.initialize(strike, maturity, rate);
        pool3.initialize(strike, maturity, rate);

        //TimeswapV2PoolMintParam memory param = TimeswapV2PoolM

        //MintOutput memory response;
        //(response.liquidityAmount, response.long0Amount, respo
        uint256 id1;
        uint256 id2;
        {
            token0Current = token0;
            token1Current = token1;
            poolCurrent = pool;
            opPairCurrent = opPair;
```

```
        TimeswapV2LiquidityTokenMintParam memory liqTokenMint
            token0: address(token0Current),
            token1: address(token1Current),
            strike: strike,
            maturity: maturity,
            to: address(this),
            liquidityAmount: amt,
            data: ""
        });

        mockLiquidityToken.mint(liqTokenMintParam);
        //console.log(mockLiquidityToken.balanceOf(address(t
        TimeswapV2LiquidityTokenPosition memory timeswapV2Li
            token0: address(token0Current),
            token1: address(token1Current),
            strike: strike,
            maturity: maturity
        });

        bytes32 key1 = timeswapV2LiquidityTokenPosition.toKe
        id1 = mockLiquidityToken._timeswapV2LiquidityTokenPo
        console.log("key1:");
        console.logBytes32(key1);
        console.log("id1:", id1);
        assertEq(mockLiquidityToken.balanceOf(address(this),
        assertEq(mockLiquidityToken.totalSupply(), 1);
        //console.log("_idTotalSupply id1:", mockLiquidityTo
        console.log("========");
    }


    {
        token0Current = token2;
        token1Current = token3;
        poolCurrent = pool2;
        opPairCurrent = opPair2;
        TimeswapV2LiquidityTokenMintParam memory liqTokenMint
            token0: address(token0Current),
            token1: address(token1Current),
            strike: strike,
            maturity: maturity,
            to: address(this),
            liquidityAmount: amt,
            data: ""
        });
```

```
        mockLiquidityToken.mint(liqTokenMintParam2);
        //console.log(mockLiquidityToken.balanceOf(address(t
        TimeswapV2LiquidityTokenPosition memory timeswapV2Li
            token0: address(token0Current),
            token1: address(token1Current),
            strike: strike,
            maturity: maturity
        });

        bytes32 key2 = timeswapV2LiquidityTokenPosition2.toK
        id2 = mockLiquidityToken._timeswapV2LiquidityTokenPo
        console.log("key2:");
        console.logBytes32(key2);
        console.log("id2:", id2);
        assertEq(mockLiquidityToken.balanceOf(address(this),
        assertEq(mockLiquidityToken.totalSupply(), 2);
        console.log("========");
    }

    TimeswapV2LiquidityTokenBurnParam memory burnParam = Tim
        token0: address(token0),
        token1: address(token1),
        strike: strike,
        maturity: maturity,
        to: address(this),
        liquidityAmount: amt,
        data: ""
    });
    mockLiquidityToken.burn(burnParam);
    console.log("balanceOf id1:", mockLiquidityToken.balance
    //console.log("_idTotalSupply id1:", mockLiquidityToken._
    console.log("current totalSupply():", mockLiquidityToken

    {
        token0Current = token4;
        token1Current = token5;
        poolCurrent = pool3;
        opPairCurrent = opPair3;
        TimeswapV2LiquidityTokenMintParam memory liqTokenMin
            token0: address(token0Current),
            token1: address(token1Current),
            strike: strike,
            maturity: maturity,
            to: address(this),
            liquidityAmount: amt,
            data: ""
```

```
            });

            mockLiquidityToken.mint(liqTokenMintParam3);
            //console.log(mockLiquidityToken.balanceOf(address(t
            TimeswapV2LiquidityTokenPosition memory timeswapV2Li
                token0: address(token0Current),
                token1: address(token1Current),
                strike: strike,
                maturity: maturity
            });

            bytes32 key3 = timeswapV2LiquidityTokenPosition3.toK
            uint256 id3 = mockLiquidityToken._timeswapV2Liquidit
            console.log("key3:");
            console.logBytes32(key3);
            console.log("id3:", id3);
            //assertEq(mockLiquidityToken.balanceOf(address(this
            if (id2 == id3) {revert("id3 should not equal to id2
            console.log("========");
        }

        console.log("yo");
    }
}
```

Here is the log for the above test: `forge test --match-path test/TimeswapV2LiquidityToken_MultiMint.t.sol -vv`

```
Running 1 test for test/TimeswapV2LiquidityToken.t.sol:TimeswapV
[FAIL. Reason: id3 should not equal to id2 Counterexample: calld
Logs:
  init
  270988320095665171922088772806210007597798872523852389880971033
  opPair mint ok
  key1:
  0x3ad1cfe6142808456d576d32877db082ef58ce80e40fb5019d9e5f73aebf
  id1: 1
  ========
  270988320095665171922088772806210007597798872523852389880971033
  opPair mint ok
  key2:
  0x4b911bdfb2c97775c28fae58288d53335ea7b59d3675acf0460ff4083897
  id2: 2
  ========
```

```
  balanceOf id1: 0
  current totalSupply(): 1
  270988320095665171922088772806210007597798872523852389880971033
  opPair mint ok
  key3:
  0x6b43d3a16273d9e9f13739b825952b03e59127b9d41c4e0d9d58d635e8d21
  id3: 2

Test result: FAILED. 0 passed; 1 failed; finished in 91.76ms

Failing tests:
Encountered 1 failing test in test/TimeswapV2LiquidityToken.t.sol
[FAIL. Reason: id3 should not equal to id2 Counterexample: calld

Encountered a total of 1 failing tests, 0 tests succeeded
```

## Recommended Mitigation Steps

Do not use `totalSupply()` or other maybe-decreasing variables for new tokenId.

Patch file can be like this:

```
diff --git a/packages/v2-token/src/TimeswapV2LiquidityToken.sol b
index 2f71a25..94e4006 100644
--- a/packages/v2-token/src/TimeswapV2LiquidityToken.sol
+++ b/packages/v2-token/src/TimeswapV2LiquidityToken.sol
@@ -32,6 +32,7 @@ contract TimeswapV2LiquidityToken is ITimeswapV

     address public immutable optionFactory;
     address public immutable poolFactory;
+    uint256 public tokenIdCounter;

     constructor(address chosenOptionFactory, address chosenPoolF
         optionFactory = chosenOptionFactory;
@@ -111,7 +112,7 @@ contract TimeswapV2LiquidityToken is ITimeswa

         // if the position does not exist, create it
         if (id == 0) {
-            id = totalSupply() + 1;
+            id = ++tokenIdCounter;
             _timeswapV2LiquidityTokenPositions[id] = timeswapV2
             _timeswapV2LiquidityTokenPositionIds[key] = id;
         }
```

Picodes (judge) increased severity to High

vhawk19 (Timeswap) confirmed and resolved:

> Fixed in PR.

## [H-03] The `collect()` function will always TRANSFER ZERO fees, losing _feesPositions without receiving fees!

*Submitted by* chaduke, *also found by* Beepidibop *and* 0xcm

Detailed description of the impact of this finding. The `collect()` function will always transfer ZERO fees. At the same time, non-zero `_feesPosition` will be burned.

```
    _feesPositions[id][msg.sender].burn(long0Fees, long1Fees, shortF
```

As a result, the contracts will be left in an inconsistent state. The user will burn `_feesPositions` without receiving the fees!

### Proof of Concept

Provide direct links to all referenced code in GitHub. Add screenshots, logs, or any other relevant proof that illustrates the concept.

The `collect()` function will always transfer ZERO fees in the following line:

```
    // transfer the fees amount to the recipient
        ITimeswapV2Pool(poolPair).transferFees(param.strike, par
```

This is because, at this moment, the values of `long0Fees`, `long1Fees`, `shortFees` have not been calculated yet, actually, they will be equal to zero. Therefore, no fees will be transferred. The values of `long0Fees`, `long1Fees`, `shortFees` are calculated afterwards by the following line:

```
(long0Fees, long1Fees, shortFees) = _feesPositions[id][msg.sende
```

Therefore, `ITimeswapV2Pool(poolPair).transferFees` must be called after this line to be correct.

## Tools Used

Remix

## Recommended Mitigation Steps

We moved the line `ITimeswapV2Pool(poolPair).transferFees` after `long0Fees`, `long1Fees`, `shortFees` have been calculated first.

```
function collect(TimeswapV2LiquidityTokenCollectParam calldata p
        ParamLibrary.check(param);

        bytes32 key = TimeswapV2LiquidityTokenPosition({token0: 

        // start the reentrancy guard
        raiseGuard(key);

        (, address poolPair) = PoolFactoryLibrary.getWithCheck(o


        uint256 id = _timeswapV2LiquidityTokenPositionIds[key];

        _updateFeesPositions(msg.sender, address(0), id);

        (long0Fees, long1Fees, shortFees) = _feesPositions[id][m

        if (param.data.length != 0)
            data = ITimeswapV2LiquidityTokenCollectCallback(msg.
                TimeswapV2LiquidityTokenCollectCallbackParam({
                    token0: param.token0,
                    token1: param.token1,
                    strike: param.strike,
                    maturity: param.maturity,
                    long0Fees: long0Fees,
                    long1Fees: long1Fees,
                    shortFees: shortFees,
                    data: param.data
```

```
                })
            );

                // transfer the fees amount to the recipient
        ITimeswapV2Pool(poolPair).transferFees(param.strike, par

        // burn the desired fees from the fees position
        _feesPositions[id][msg.sender].burn(long0Fees, long1Fees

        if (long0Fees != 0 || long1Fees != 0 || shortFees != 0)

        // stop the reentrancy guard
        lowerGuard(key);
    }
```

**vhawk19 (Timeswap) confirmed and resolved:**

> Fixed in PR.

## Medium Risk Findings (7)

### [M-01] `_currentIndex` is incorrectly updated; breaking the ERC1155 enumerable implementation

*Submitted by eierina, also found by adriro*

https://github.com/code-423n4/2023-01-timeswap/blob/3be51465583552cce76816a05170fda7da68596a/packages/v2-token/src/base/ERC1155Enumerable.sol#L92-L101

https://github.com/code-423n4/2023-01-timeswap/blob/3be51465583552cce76816a05170fda7da68596a/packages/v2-token/src/base/ERC1155Enumerable.sol#L116-L121

https://github.com/code-423n4/2023-01-timeswap/blob/3be51465583552cce76816a05170fda7da68596a/packages/v2-token/src/base/ERC1155Enumerable.sol#L136-L149

## Impact

When minting and burning tokens,the ERC1155Enumerable implementation does not correctly update the following states:

- uint256[] private _allTokens;
- mapping(uint256 => uint256) private _allTokensIndex;
- mapping(address => uint256) internal _currentIndex;

In particular:

- the _allTokens array length (and therefore the totalSupply()) always increases (never decreases)
- the _allTokensIndex[id] always increases
- the _curentIndex[from] always increases

## Proof of Concept

NOTE: the following test requires some private states of ERC1155Enumerable.sol to be set from private to internal.

```
contract HelperERC1155 is ERC1155Enumerable, ERC1155Holder {

    constructor() ERC1155("Test") {
    }

    function mint(uint256 id, uint256 amount) external {
        _mint(msg.sender, id, amount, bytes(""));
    }

    function burn(uint256 id, uint256 amount) external {
        _burn(msg.sender, id, amount);
    }

    function currentIndex(address owner) external view returns (
        return _currentIndex[owner];
    }

    function allTokensIndex(uint256 id) external view returns (u
        return _allTokensIndex[id];
    }
```

```
    function allTokens(uint256 idx) external view returns (uint2
        return _allTokens[idx];
    }

    function idTotalSupply(uint256 id) external view returns (ui
        return _idTotalSupply[id];
    }
}

contract BugTest is Test, ERC1155Holder {

    function testImplError() public {
        HelperERC1155 token = new HelperERC1155();

        for(uint i=0; i<10; i++){
            token.mint(i, 1+i);
        }

        for(uint i=0; i<10; i++){
            token.burn(i, 1+i);
            assertEq(token.idTotalSupply(i), 0); // OK
            assertEq(token.allTokensIndex(i), i); // NOT OK (sho
        }

        assertEq(token.totalSupply(), 10); // NOT OK (should be
        assertEq(token.currentIndex(address(this)), 10); // NOT
    }

    function testImplFixed() public {
        HelperERC1155 token = new HelperERC1155();

        for(uint i=0; i<10; i++){
            token.mint(i, 1+i);
        }

        for(uint i=0; i<10; i++){
            token.burn(i, 1+i);
            assertEq(token.idTotalSupply(i), 0); // OK
            assertEq(token.allTokensIndex(i), 0); // OK
        }

        assertEq(token.totalSupply(), 0); // OK
        assertEq(token.currentIndex(address(this)), 0); // OK
    }
}
```

Before fix `forge test --match-contract BugTest -vvv` outputs:

```
Running 2 tests for test/Audit2.t.sol:BugTest
[PASS] testImplError() (gas: 2490610)
[FAIL. Reason: Assertion failed.] testImplFixed() (gas: 2560628)
Test result: FAILED. 1 passed; 1 failed; finished in 2.05ms
```

After fix `forge test --match-contract BugTest -vvv` outputs:

```
Running 2 tests for test/Audit2.t.sol:BugTest
[FAIL. Reason: Assertion failed.] testImplError() (gas: 2558695)
[PASS] testImplFixed() (gas: 2489080)
Test result: FAILED. 1 passed; 1 failed; finished in 2.22ms
```

## Recommended Mitigation Steps

Correct the implementation to update states correctly. Patch provided below for reference.

```
diff --git a/packages/v2-token/src/base/ERC1155Enumerable.sol b/
index 4ec23ff..ef67bca 100644
--- a/packages/v2-token/src/base/ERC1155Enumerable.sol
+++ b/packages/v2-token/src/base/ERC1155Enumerable.sol
@@ -91,8 +91,8 @@ abstract contract ERC1155Enumerable is IERC115
     /// @dev Remove token enumeration list if necessary.
     function _removeTokenEnumeration(address from, address to,
         if (to == address(0)) {
-            if (_idTotalSupply[id] == 0 && _additionalCondition
             _idTotalSupply[id] -= amount;
+            if (_idTotalSupply[id] == 0 && _additionalCondition
         }

         if (from != address(0) && from != to) {
@@ -114,8 +114,7 @@ abstract contract ERC1155Enumerable is IERC1
     /// @param to address representing the new owner of the give
     /// @param tokenId uint256 ID of the token to be added to t
     function _addTokenToOwnerEnumeration(address to, uint256 to
-        _currentIndex[to] += 1;
-        uint256 length = _currentIndex[to];
+        uint256 length = _currentIndex[to]++;
         _ownedTokens[to][length] = tokenId;
```

```
            _ownedTokensIndex[tokenId] = length;
        }
@@ -134,7 +133,7 @@ abstract contract ERC1155Enumerable is IERC1
        /// @param from address representing the previous owner of
        /// @param tokenId uint256 ID of the token to be removed fr
        function _removeTokenFromOwnerEnumeration(address from, uin
-           uint256 lastTokenIndex = _currentIndex[from] - 1;
+           uint256 lastTokenIndex = --_currentIndex[from];
            uint256 tokenIndex = _ownedTokensIndex[tokenId];

            if (tokenIndex != lastTokenIndex) {
```

[Picodes (judge) commented](#):

> There are 2 bugs highlighted here:
>
> - the check is incorrectly made before the state update in `_removeTokenEnumeration`
> - the order in which `_currentIndex` is updated
>
> So splitting this finding in 2. *(Note: issue title has been updated accordingly. Also, see newly created issue `#300` .)*

[vhawk19 (Timeswap) confirmed and commented](#):

> Updated the ERC1155Enumerable.sol implementation, which should resolve these issues.

## [M-02] Burning a `ERC1155Enumerable` token doesn't remove it from the enumeration

*Submitted by adriro, also found by eierina, hansfriese, mookimgo, and chaduke*

The `ERC1155Enumerable` base contract used in the `TimeswapV2Token` and `TimeswapV2LiquidityToken` tokens provides a functionality to enumerate all token ids that have been minted in the contract.

The logic to remove the token from the enumeration if the last token is burned is implemented in the `_afterTokenTransfer` hook:

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/base/ERC1155Enumerable.sol#L81-L101

```
    function _afterTokenTransfer(address, address from, address to, 
        for (uint256 i; i < ids.length; ) {
            if (amounts[i] != 0) _removeTokenEnumeration(from, to, i

            unchecked {
                ++i;
            }
        }
    }

    /// @dev Remove token enumeration list if necessary.

    function _removeTokenEnumeration(address from, address to, uint2
        if (to == address(0)) {
            if (_idTotalSupply[id] == 0 && _additionalConditionRemov
            _idTotalSupply[id] -= amount;
        }

        if (from != address(0) && from != to) {
            if (balanceOf(from, id) == 0 && _additionalConditionRemo
        }
    }
```

The `_removeTokenEnumeration` condition to check if the supply is 0 happens before the function decreases the burned amount. This will `_removeTokenFromAllTokensEnumeration` from being called when the last token(s) is(are) burned.

## Impact

The token isn't removed from the enumeration since `_removeTokenFromAllTokensEnumeration` will never be called. This will cause the enumeration to always contain a minted token even though it is burned afterwards. The function `totalSupply` and `tokenByIndex` will report wrong values.

This will also cause the enumeration to contain duplicate values or multiple copies of the same token. If the token is minted again after all tokens were previously burned, the token will be re added to the enumeration.

### Proof of Concept

The following test demonstrates the issue. Alice is minted a token and that token is then burned, the token is still present in the enumeration. The token is minted again, causing the enumeration to contain the token by duplicate.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity =0.8.8;

import "forge-std/Test.sol";
import "../src/base/ERC1155Enumerable.sol";

contract TestERC1155Enumerable is ERC1155Enumerable {
    constructor() ERC1155("") {
    }

    function mint(address to, uint256 id, uint256 amount) externa
        _mint(to, id, amount, "");
    }

    function burn(address from, uint256 id, uint256 amount) exte
        _burn(from, id, amount);
    }
}

contract AuditTest is Test {
    function test_ERC1155Enumerable_BadRemoveFromEnumeration() pu
        TestERC1155Enumerable token = new TestERC1155Enumerable(
        address alice = makeAddr("alice");
        uint256 tokenId = 0;
        uint256 amount = 1;

        token.mint(alice, tokenId, amount);

        // tokenByIndex and totalSupply are ok
        assertEq(token.tokenByIndex(0), tokenId);
        assertEq(token.totalSupply(), 1);

        // now we burn the token
        token.burn(alice, tokenId, amount);
```

```
            // tokenByIndex and totalSupply still report previous va
            // tokenByIndex should throw index out of bounds, and su
            assertEq(token.tokenByIndex(0), tokenId);
            assertEq(token.totalSupply(), 1);

            // Now we mint it again, this will re-add the token to t
            token.mint(alice, tokenId, amount);
            assertEq(token.totalSupply(), 2);
            assertEq(token.tokenByIndex(0), tokenId);
            assertEq(token.tokenByIndex(1), tokenId);
        }
    }
```

### Recommendation

Decrease the amount before checking if the supply is 0.

```
    function _removeTokenEnumeration(address from, address to, uint2
        if (to == address(0)) {
            _idTotalSupply[id] -= amount;
            if (_idTotalSupply[id] == 0 && _additionalConditionRemov
        }

        if (from != address(0) && from != to) {
            if (balanceOf(from, id) == 0 && _additionalConditionRemo
        }
    }
```

**vhawk19 (Timeswap) confirmed and resolved:**

> Resolved in PR.

## [M-03] Fee on transfer tokens will not behave as expected

*Submitted by* RaymondFam, *also found by* rbserver, nadin, kaden, pavankv, mert_eren, SaeedAlipoor01988, *and* Rolezn

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2Option.sol#L145-L148

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2Option.sol#L235

## Impact

According to Whitepaper 1.1 Permissionless:

“In Timeswap, liquidity providers can create pools for any ERC20 pair, without permission. It is designed to be generalized and works for any pair of tokens, at any time frame, and at any market state ...

If fee on transfer token(s) is/are entailed, it will specifically make `mint()` and `swap()` revert in TimeswapV2Option.sol when checking if the token0 or token1 balance target is achieved.

## Proof of Concept

File: TimeswapV2Option.sol#L144-L148

```
        // check if the token0 balance target is achieved.
        if (token0AndLong0Amount != 0) Error.checkEnough(IERC20(

        // check if the token1 balance target is achieved.
        if (token1AndLong1Amount != 0) Error.checkEnough(IERC20(
```

File: TimeswapV2Option.sol#L234-L235

```
        // check if the token0 or token1 balance target is achie
        Error.checkEnough(IERC20(param.isLong0ToLong1 ? token1 :
```

File: Error.sol#L148-L153

```
    /// @dev Reverts when token amount not received.
    /// @param balance The balance amount being subtracted.
    /// @param balanceTarget The amount target.
    function checkEnough(uint256 balance, uint256 balanceTarget)
        if (balance < balanceTarget) revert NotEnoughReceived(ba
    }
```

As can be seen from the code blocks above, `checkEnough()` is meant to be reverting when token amount has not been received. But in the case of deflationary tokens, the error is going to be thrown even though the token amount has been received due to the fee factor making `balance < balanceTarget`, i.e the contract balance of token0 and/or token1 always less than `currentProcess.balance0Target` or `currentProcess.balance1Target`.

### Recommended Mitigation Steps

Consider:

1. Whitelisting token0 and token1 ensuring no fee-on-transfer token is allowed when deploying a new Timeswap V2 Option pair contract, or
2. Calculating the balance before and after the transfer to the recipient during the process, and use the difference between those two balances as the amount received rather than using the input amount ( `token0AndLong0Amount` or `token1AndLong1Amount` ) if deflationary token is going to be allowed in the protocol.

**vhawk19 (Timeswap) acknowledged and commented**:

> Not supported by design.

## [M-04] sqrtDiscriminant can be calculated wrong

*Submitted by* **sorrynotsorry**

Due to the wrong calculation of short and long tokens during the `leverage` and `deleverage` process, the users can suffer financial loss while the protocol will lose fees.

### Proof of Concept

The protocol uses `leverage` function to deposit short tokens and receive long tokens. On the opposite, `deleverage` function serves for depositing long tokens and receiving short tokens.

**Leverage Function of TimeswapV2Pool contract**

## Deleverage Function of TimeswapV2Pool contract

Both functions call the PoolLibrary's `leverage` and `deleverage` functions after input sanitization.

## Leverage Function of PoolLibrary contract

## Deleverage Function of PoolLibrary contract

PoolLibrary's `leverage` and `deleverage` functions update the state of the pool first for the fee growth and compute the `long0Amount`, `long1Amount`, and `shortAmount`. It also checks the transaction type according to the passed parameter types as per the `Transaction` contract's enum types below and calls `ConstantProduct`'s appropriate function accordingly;

```
/// @dev The different kind of deleverage transactions.
enum TimeswapV2PoolDeleverage {
    GivenDeltaSqrtInterestRate,
    GivenLong,
    GivenShort,
    GivenSum
}

/// @dev The different kind of leverage transactions.
enum TimeswapV2PoolLeverage {
    GivenDeltaSqrtInterestRate,
    GivenLong,
    GivenShort,
    GivenSum
}
```

If the transaction type is `GivenSum`, both `leverage` and `deleverage` functions of PoolLibrary call `ConstantProduct.updateGivenSumLong` for the sum amount of the long position in the base denomination to be withdrawn, and the short position to be deposited.

```
} else if (param.transaction == TimeswapV2PoolDeleverage.GivenSu
        (pool.sqrtInterestRate, longAmount, shortAmount, shortFe
```

```
...
```

Link

```
} else if (param.transaction == TimeswapV2PoolLeverage.GivenSum)
        (pool.sqrtInterestRate, longAmount, shortAmount, ) = Con
```

Link

`updateGivenSumLong` updates the new square root interest rate given the sum of long positions in base denomination change and short position change;

```
function updateGivenSumLong(
    uint160 liquidity,
    uint160 rate,
    uint256 sumAmount,
    uint96 duration,
    uint256 transactionFee,
    bool isAdd
) internal pure returns (uint160 newRate, uint256 longAmount
    uint256 amount = getShortOrLongFromGivenSum(liquidity, r

    if (isAdd) (newRate, ) = getNewSqrtInterestRateGivenShor
    else newRate = getNewSqrtInterestRateGivenLong(liquidity

    fees = FeeCalculation.getFeesRemoval(amount, transaction
    amount -= fees;

    if (isAdd) {
        shortAmount = amount;
        longAmount = sumAmount - shortAmount;
    } else {
        longAmount = amount;
        shortAmount = sumAmount - longAmount;
    }
}
```

Link

And `updateGivenSumLong` calls `getShortOrLongFromGivenSum` in order to return the `amount` which represents the short amount or long amount calculated.

```
function getShortOrLongFromGivenSum(uint160 liquidity, uint1
    uint256 negativeB = calculateNegativeB(liquidity, rate,
    uint256 sqrtDiscriminant = calculateSqrtDiscriminant(liq
    amount = (negativeB - sqrtDiscriminant).shr(1, false);
}
```

Link

And the formula needs `sqrtDiscriminant` value to calculate the `amount` and it calls `calculateSqrtDiscriminant` accordingly

`calculateSqrtDiscriminant` function performs a bunch of checks and carries out mathematical functions to return the SqrtDiscriminant by utilizing FullMath and Math libraries.

```
sqrtDiscriminant = FullMath.sqrt512(b0, b1, true);
```

Link

The `sqrt` formula in the Math contract uses the modified version of Babylonian Method when flags are included.

```
function sqrt(uint256 value, bool roundUp) internal pure ret
    if (value == type(uint256).max) return result = type(uin
    if (value == 0) return 0;
    unchecked {
        uint256 estimate = (value + 1) >> 1;
        result = value;
        while (estimate < result) {
            result = estimate;
            estimate = (value / estimate + estimate) >> 1;
        }
    }

    if (roundUp && value % result != 0) result++;
```

```
}
```

[Link]

However, when the parameter `roundUp` is passed as `true`, this results in inconsistent behavior for different values. And it's being passed as true as can be seen [here]).

In order to show some examples let's pass the numbers as values and flag them true by using Math's `sqrt` function.

| Values | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Results | 1 | 1 | 1 | 2 | 3 | **2** | 3 | **2** | 3 | 4 | 4 | **3** | 4 | 4 | **3** | 4 | 5 | 5 | 5 | **4** | 5 | 5 | 5 | **4** | 5 | |

As can be seen from the table, the results are not distributed logically. And many times the result is steeply lesser than its neighbor results. (E.g Sqrt(6) ->2, Sqrt(15)->3 etc.)

The phenomenon occurs most if the values are small numbers.

So if the parameter `value1` in `FullMath.sqrt512` is passed/calculated as zero value, it has a high chance of providing a wrong calculation as a result with the line below;

```
function sqrt512(uint256 value0, uint256 value1, bool roundU
    if (value1 == 0) result = value0.sqrt(roundUp);
```

This may lead to the wrong calculation of the `sqrtDiscriminant`, hence the wrong calculation of short or long amounts for the given transaction. The users might lose financial value due to this. Accordingly, the protocol might lose unspent fees as well.

While the fewer values are affected more on this one, the pools with fewer token decimals and fewer token amounts are more affected by this error. As an example, a Gemini Dollar pool (59th rank on CMC and having 2 decimals) would be subject to false returns.

### Tools Used

Remix, Excel

### Recommended Mitigation Steps

The team might consider not using `true` flag for `Math.sqrt` function.

**vhawk19 (Timeswap) confirmed and resolved:**

> Fixed in PR.

## [M-05] unexpected overflow for `FullMath.add512()` which can result in irregular behavior

*Submitted by* codeislight

The vulnerability originates from insufficient checking in add512 function, where the AddOverflow revert gets bypassed, essentially the function assumes that an overflow only happens if (addendA1 > sum1), where in the case that it's possible for it to overflow in the case that addendA1 == sum1, which can be resulted through assigning a value that makes ( lt(sum0, addendA0) == 1 ) <=> sum0 < addendA0, which can only be achieved normally by overflowing the least significant addition. Then we can simply break the overflow check by assigning overflowing values which results in add(addendA1, addendB1) > type(256).max && addendA1 <= sum1, then we will manage to bypass the revert check and overflow the most significant part of add512 values.

The previous attack vector can lead to a manipulation in leverage and deleverage functions, in a way that it would result in more tokens for the user.

### Proof of Concept

Inputting the following values results in an overflow:

uint256 addendA0 = 1

uint256 addendA1 = 100

uint256 addendB0 = 115792089237316195423570985008687907853269984665640564039457584007913129639935 (uint256 max value)

uint256 addendB1 = 115792089237316195423570985008687907853269984665640564039457584007913129639935 (uint256 max value)

results in:

sum0 = 0

sum1 = 100

The expected behavior is to revert since, A1 + B1 result in a value that overflows, but instead consider it as a valid behavior due to the insufficient checking.

Abstraction:

A1 - A0

+

B1 - B0

=

S1 - S0

S0 = A0 + B0

S1 = A1 + B1 + ( if S0 overflows [+ 1])

ensure A1 <= S1

revert only on A1 > S1

in the case of S0 overflows:

S1 = A1 + B1 + 1

require(A1 <= S1) is not most suited check, due to the fact that in the case of A1 == S1 check, it can still overflow if S1 = A1 + B1 + 1 overflows. which would bypass A1 > S1 revert check.

The major impact affects the `leverage()` and `deleverage()` results in values which are not expected.

### Recommended Mitigation Steps

Add an equality check for if statement in add512 function.

```
    function add512(uint256 addendA0, uint256 addendA1, uint256 a
        assembly {
            sum0 := add(addendA0, addendB0)
            carry  := lt(sum0, addendA0)
            sum1 := add(add(addendA1, addendB1), carry)
        }
        if (addendA1 > sum1 ||      ((sum1 == addendA1 || sum1 =
) revert AddOverflow(addendA0, addendA1, addendB0, addendB1);
//
    }
```

**Picodes (judge) decreased severity to Medium and commented:**

> No explanation related to how this could lead to errors in `leverage` or `deleverage`.

**vhawk19 (Timeswap) confirmed and resolved:**

> Fixed in PR.

## [M-06] `_ownedTokensIndex` is SHARED by different owners, as a result, `_removeTokenFromAllTokensEnumeration` might remove the wrong tokenId.

*Submitted by* chaduke*, also found by* adriro

The data structure `_ownedTokensIndex` is SHARED by different owners, as a result, `_removeTokenFromAllTokensEnumeration()` might remove the wrong tokenId.

### Proof of Concept

`_ownedTokensIndex` is used to map from token ID to index of the owner tokens list, unfortunately, all owners share the same data structure at the same time (non-fungible tokens). So, the mapping for one owner might be overwritten by another owner when `_addTokenToOwnerEnumeration` is called: https://github.com/code-423n4/2023-01-timeswap/blob/ef4c84fb8535aad8abd6b67cc45d994337ec4514/packages/v2-token/src/base/ERC1155Enumerable.sol#L116-L121. As a result, `_removeTokenFromOwnerEnumeration()` might remove the wrong tokenID.

Removing the wrong tokenID can happen like the following:

1. Suppose Alice owns three tokens A, B, C with indices 1 -> A, 2->B, 3->C
2. Suppose Bob owns token D, 1->D, and will add A to his list via `_addTokenToOwnerEnumeration()`. As a result, we have 1->D, and 2-A, since `_ownedTokensIndex` is shared, we have A->2 in `_ownedTokensIndex`.
3. Next, `_removeTokenFromOwnerEnumeration()` is called to remove A from Alice. However, `tokenIndex` will be 2, which points to B, as a result, instead of deleting A, B is deleted from `_ownedTokens`. Wrong token delete!

   function _removeTokenFromOwnerEnumeration(address from, uint256 tokenId) private { uint256 lastTokenIndex = _currentIndex[from] - 1; uint256 tokenIndex = _ownedTokensIndex[tokenId];

```
        if (tokenIndex != lastTokenIndex) {
            uint256 lastTokenId = _ownedTokens[from][lastTokenIndex];

            _ownedTokens[from][tokenIndex] = lastTokenId;
```

```
            _ownedTokensIndex[lastTokenId] = tokenIndex;
        }

        delete _ownedTokensIndex[tokenId];
        delete _ownedTokens[from][lastTokenIndex];
    }
```

### Tools Used

Remix

### Recommended Mitigation Steps

Redefine `ownedTokensIndex` so that is is not shared:

```
mapping(address => mapping(uint256 => uint256)) private _ownedTo
```

**vhawk19 (Timeswap) confirmed and commented:**

> Updated the ERC1155Enumerable.sol implementation, which should resolve these issues.

## [M-07] `Mint` function does not update `LiquidityPosition` state of caller before minting LP tokens. This

*Submitted by* 0Kage

https://github.com/code-423n4/2023-01-timeswap/blob/ef4c84fb8535aad8abd6b67cc45d994337ec4514/packages/v2-pool/src/structs/Pool.sol#L302

https://github.com/code-423n4/2023-01-timeswap/blob/ef4c84fb8535aad8abd6b67cc45d994337ec4514/packages/v2-pool/src/structs/LiquidityPosition.sol#L60

### Impact

When a LP mints V2 Pool tokens, `mint` function in PoolLibrary gets called. Inside this function `updateDurationWeightBeforeMaturity` updates global `short`, `long0` and `long1` fee growth.

Change in global fee growth necessitates an update to `LiquidityPosition` state of caller (specifically updating fees & fee growth rates) when there are state changes made to that position (in this case, increasing liquidity). This principle is followed in functions such as `burn`, `transferLiquidity`, `transferFees`. However when calling `mint`, this update is missing. As a result, `growth` & `fee` levels in liquidity position of caller are inconsistent with global fee growth rates.

Inconsistent state leads to incorrect calculations of long0/long1 and short fees of LP holders which in turn can lead to loss of fees. Since this impacts actual rewards for users, I've marked it as MEDIUM risk.

## Proof of Concept

Let's say, Bob has following sequence of events

- MINT at T0: Bob is a LP who mints N pool tokens at T0
- MINT at T1: Bob mints another M pool tokens at T1. At this point, had the protocol correctly updated fees before minting new pool tokens, Bob's fees & growth rate would be a function of current liquidity (N), global updated short fee growth rate at t1 (s*t1) and Bob's previous growth rate at t*\0 (b_t0)
- BURN at T2: Bob burns N + M tokens at T2. At this point, Bob's fees should be a function of previous liquidity (N+M), global short fee growth rate (s*t2) and Bob's previous growth rate at t*\1(b*t1) -> since this update never happened, Bob's previous growth rate is wrongly referenced b*t0 instead of b_t1.

Bob could collect a lower fees because of this state inconsistency.

## Recommended Mitigation Steps

Update the liquidity position state right before minting.

After line 302 of Pool.sol, update the LiquidityPosition by adding

```
liquidityPosition.update(pool.long0FeeGrowth, pool.long1FeeGro
```

vhawk19 (Timeswap) confirmed

## Low Risk and Non-Critical Issues

For this contest, 36 reports were submitted by wardens detailing low risk and non-critical issues. The report highlighted below by **rbserver** received the top score from the judge.

*The following wardens also submitted reports:* ddimitrov22, Udsen, Breeje, matrix_0wl, hansfriese, 0xAgro, Josiah, luxartvinsec, Diana, tnevler, delfin454000, Awesome, mookimgo, cryptonue, shark, lllllll, 0x1f8b, fatherOfBlocks, 0xSmartContract, brgltd, oberon, lukris02, Viktor_Cortess, Moksha, DadeKuma, 0xGusMcCrae, popular00, chaduke, georgits, descharre, martin, RaymondFam, btk, Rolezn, *and* SaeedAlipoor01988 .

### [01] User can possibly transfer no `token0` or `token1` to `TimeswapV2Option` contract if corresponding `token0` OR `token1` is a rebasing token

When calling the following `TimeswapV2Option.mint` function, `msg.sender` uses the `ITimeswapV2OptionMintCallback.timeswapV2OptionMintCallback` function to transfer the relevant `token0` and/or `token1` to the `TimeswapV2Option` contract. Similarly, when calling the `TimeswapV2Option.swap` function below, `msg.sender` uses the `ITimeswapV2OptionSwapCallback.timeswapV2OptionSwapCallback` function to transfer the relevant `token0` or `token1` to the `TimeswapV2Option` contract. When `token0` or `token1` is a rebasing token, it is possible that the user uses these callback functions to trigger such token's rebasing event that increases its balance owned by the `TimeswapV2Option` contract.

Then, when the `TimeswapV2Option.mint` and `TimeswapV2Option.swap` functions call `Error.checkEnough` , the rebasing token's balance owned by the `TimeswapV2Option` contract can possibly exceed the corresponding balance target. As a result, the user is able to mint or swap option positions without sending any of such rebasing token to the `TimeswapV2Option` contract.

As a mitigation, this protocol can behave like other protocols that do not support rebasing tokens and use a blocklist to block such tokens from being used as `token0` or `token1` for any options.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2Option.sol#L109-L154

```
function mint(
    TimeswapV2OptionMintParam calldata param
) external override noDelegateCall returns (uint256 token0An
    ...

    Option storage option = options[param.strike][param.matu

    // does main mint logic calculation
    (token0AndLong0Amount, token1AndLong1Amount, shortAmount

    // update token0 and token1 balance target for any previ
    processing.updateProcess(token0AndLong0Amount, token1And

    // add a new process
    // stores the token0 and token1 balance target required
    Process storage currentProcess = (processing.push() = Pr
        param.strike,
        param.maturity,
        IERC20(token0).balanceOf(address(this)) + token0AndL
        IERC20(token1).balanceOf(address(this)) + token1AndL
    ));

    // ask the msg.sender to transfer token0 and/or token1 t
    data = ITimeswapV2OptionMintCallback(msg.sender).timeswa
        TimeswapV2OptionMintCallbackParam({
            strike: param.strike,
            maturity: param.maturity,
            token0AndLong0Amount: token0AndLong0Amount,
            token1AndLong1Amount: token1AndLong1Amount,
            shortAmount: shortAmount,
            data: param.data
        })
    );

    // check if the token0 balance target is achieved.
    if (token0AndLong0Amount != 0) Error.checkEnough(IERC20(
```

```
        // check if the token1 balance target is achieved.
        if (token1AndLong1Amount != 0) Error.checkEnough(IERC20(

        ...
    }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2Option.sol#L198-L244

```
    function swap(TimeswapV2OptionSwapParam calldata param) exte
        ...

        Option storage option = options[param.strike][param.matu

        // does main swap logic calculation
        (token0AndLong0Amount, token1AndLong1Amount) = option.sw

        // update token0 and token1 balance target for any previ
        processing.updateProcess(token0AndLong0Amount, token1And

        // add a new process
        // stores the token0 and token1 balance target required
        Process storage currentProcess = (processing.push() = Pr
            param.strike,
            param.maturity,
            param.isLong0ToLong1 ? IERC20(token0).balanceOf(addr
            param.isLong0ToLong1 ? IERC20(token1).balanceOf(addr
        ));

        // transfer token to recipient.
        IERC20(param.isLong0ToLong1 ? token0 : token1).safeTrans

        // ask the msg.sender to transfer token0 or token1 to th
        data = ITimeswapV2OptionSwapCallback(msg.sender).timeswa
            TimeswapV2OptionSwapCallbackParam({
                strike: param.strike,
                maturity: param.maturity,
                isLong0ToLong1: param.isLong0ToLong1,
                token0AndLong0Amount: token0AndLong0Amount,
                token1AndLong1Amount: token1AndLong1Amount,
                data: param.data
            })
        );
```

```
        // check if the token0 or token1 balance target is achie
        Error.checkEnough(IERC20(param.isLong0ToLong1 ? token1 :

        ...
    }
```

## [02] `Error.checkEnough` function does not prevent user from sending too many tokens to relevant contract

When calling functions like `TimeswapV2Option.mint` or `TimeswapV2Option.swap`, the user will use the callback function like `ITimeswapV2OptionMintCallback.timeswapV2OptionMintCallback` or `ITimeswapV2OptionSwapCallback.timeswapV2OptionSwapCallback` to send some of the corresponding tokens to the relevant contract, such as `TimeswapV2Option`. Calling functions like `TimeswapV2Option.mint` or `TimeswapV2Option.swap` will revert if its call to the following `Error.checkEnough` function reverts when the token amount transferred through the callback function is not enough. However, if user sends too many tokens through the callback function, the `Error.checkEnough` function does not revert; when this happens, the extra token amount after the corresponding token balance target is met will be locked in the relevant receiving contract like `TimeswapV2Option` so the user loses such extra amount.

As a mitigation, the `balance < balanceTarget` condition in the `Error.checkEnough` function can be updated to `balance != balanceTarget`. Alternatively, some logic can be added for returning the sent extra token amount back to the user.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/Error.sol#L151-L153

```
    function checkEnough(uint256 balance, uint256 balanceTarget)
        if (balance < balanceTarget) revert NotEnoughReceived(ba
    }
```

## [03] Pool considers option not matured when its maturity and the block timestamp are equal

As shown by the following comparisons between the option's maturity and the block timestamp in the following `TimeswapV2Pool.initialize` function and the pool's various `ParamLibrary.check` functions, the pool considers the corresponding option not matured when its maturity and the block timestamp are equal. However, this is inconsistent with the option's definition, which considers the option matured when its maturity and the block timestamp are equal as the option's various `ParamLibrary.check` functions below show. The `TimeswapV2Pool` contract's `mint`, `burn`, `deleverage`, `leverage`, and `rebalance` functions all have the `can be only called before the maturity` comment indicating that these functions are supposed to be callable only when the corresponding option is not matured. Users who checked these comments can believe that these functions are callable when the option's maturity and the block timestamp are equal; yet, calling these functions at such timing will revert due to the duration to the option's maturity is already 0 and the inconsistency between the pool and option's definitions of whether the option is matured. As a result, in this case, the user's calls of these `TimeswapV2Pool` contract's functions will revert unexpectedly with the used gas being wasted, and the user experience becoming degraded.

As a mitigation, the pool and option's definitions of whether the option is matured need to be updated to be consistent.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2Pool.sol#L175-L181

```
function initialize(uint256 strike, uint256 maturity, uint16
    if (maturity < blockTimestamp(0)) Error.alreadyMatured(m
    ...
}
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/structs/Param.sol#L142-L194

```
...
function check(TimeswapV2PoolMintParam memory param, uint96 
    ...
    if (param.maturity < blockTimestamp) Error.alreadyMature
    ...
}
```

```
...
function check(TimeswapV2PoolBurnParam memory param, uint96 l
    ...
    if (param.maturity < blockTimestamp) Error.alreadyMature
    ...
}

...
function check(TimeswapV2PoolDeleverageParam memory param, u
    ...
    if (param.maturity < blockTimestamp) Error.alreadyMature
    ...
}

...
function check(TimeswapV2PoolLeverageParam memory param, uin
    ...
    if (param.maturity < blockTimestamp) Error.alreadyMature
    ...
}

...
function check(TimeswapV2PoolRebalanceParam memory param, ui
    ...
    if (param.maturity < blockTimestamp) Error.alreadyMature
    ...
}
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/structs/Param.sol#L103-L151

```
function check(TimeswapV2OptionMintParam memory param, uint9
    ...
    if (param.maturity <= blockTimestamp) Error.alreadyMatur
    ...
}

...
function check(TimeswapV2OptionBurnParam memory param, uint9
    ...
    if (param.maturity <= blockTimestamp) Error.alreadyMatur
    ...
}
```

```
...
function check(TimeswapV2OptionSwapParam memory param, uint9
    ...
    if (param.maturity <= blockTimestamp) Error.alreadyMatur
    ...
}

...
function check(TimeswapV2OptionCollectParam memory param, ui
    ...
    if (param.maturity > blockTimestamp) Error.stillActive(p
    ...
}
```

## [04] Pool's `ParamLibrary.check` function for `TimeswapV2PoolCollectParam` checks `param.strike == 0` less restrictively than pool's other `ParamLibrary.check` functions

As shown below, calling the pool's `ParamLibrary.check` function for `TimeswapV2PoolCollectParam` will not revert when `param.strike` is 0 while one or more of `param.long0Requested`, `param.long1Requested`, or `param.shortRequested` is not 0. However, calling the pool's other `ParamLibrary.check` functions for other `param` will revert whenever `param.strike == 0` is true. To handle the `param.strike == 0` check consistently, please consider updating the `&& param.strike == 0` condition to `|| param.strike == 0` in the pool's `ParamLibrary.check` function for `TimeswapV2PoolCollectParam`.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/structs/Param.sol#L133-L194

```
function check(TimeswapV2PoolCollectParam memory param) inte
    ...
    if (param.long0Requested == 0 && param.long1Requested ==
}

...
function check(TimeswapV2PoolMintParam memory param, uint96 l
```

```
        ...
        if (param.delta == 0 || param.strike == 0) Error.zeroInp
    }

    ...
    function check(TimeswapV2PoolBurnParam memory param, uint96 l
        ...
        if (param.delta == 0 || param.strike == 0) Error.zeroInp
    }

    ...
    function check(TimeswapV2PoolDeleverageParam memory param, u
        ...
        if (param.delta == 0 || param.strike == 0) Error.zeroInp
    }

    ...
    function check(TimeswapV2PoolLeverageParam memory param, uin
        ...
        if (param.delta == 0 || param.strike == 0) Error.zeroInp
    }

    ...
    function check(TimeswapV2PoolRebalanceParam memory param, ui
        ...
        if (param.delta == 0 || param.strike == 0) Error.zeroInp
    }
```

## [05] Redundant named returns

When a function has unused named returns and used return statements, these named returns become redundant. To improve readability and maintainability, these variables for the named returns can be removed while keeping the return statements for the functions associated with the following lines.

```
v2-pool\src\TimeswapV2Pool.sol
  240: function mint(TimeswapV2PoolMintParam calldata param) exte
  248: ) external override returns (uint160 liquidityAmount, uin
  305: function burn(TimeswapV2PoolBurnParam calldata param) exte
  313: ) external override returns (uint160 liquidityAmount, uin
```

## [06] Word/typing typos

`ot` can be changed to `to` in the following comment.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/StrikeConversion.sol#L22

```
        /// @param amount The amount ot be converted. Token0 amount
```

`overidden` can be changed to `overridden` in the following comment.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2Option.sol#L69

```
        // Can be overidden for testing purposes.
```

`positionss` can be changed to `positions` in the following comment.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/interfaces/callbacks/ITimeswapV2PoolMintCallback.sol#L10

```
        /// @dev The liquidity positionss will already be minted to
```

## [07] Confusing NatSpec `@param` usage

Because `data` is the returned variable of the following functions, `@return` can be used instead of `@param` in the corresponding NatSpec comment to avoid confusion.

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/interfaces/callbacks/ITimeswapV2PoolMintCallback.sol#L13-L14

```
        /// @param data The bytes of data to be sent to msg.sender.
        function timeswapV2PoolMintChoiceCallback(TimeswapV2PoolMint
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/interfaces/callbacks/ITimeswapV2PoolMintCallback.sol#L17-L18

```
        /// @param data The bytes of data to be sent to msg.sender.
        function timeswapV2PoolMintCallback(TimeswapV2PoolMintCallba
```

## [08] Incomplete NatSpec Comments

NatSpec comments provide rich code documentation. The following functions are some examples that miss the `@param` and/or `@return` comments.

Please consider completing the NatSpec comments for functions like these.

```
v2-library\src\Error.sol
  125: function inactiveOptionChoice(uint256 strike, uint256 mat

v2-library\src\StrikeConversion.sol
  16: function convert(uint256 amount, uint256 strike, bool zero
  26: function turn(uint256 amount, uint256 strike, bool toOne,
  35: function combine(uint256 amount0, uint256 amount1, uint256

v2-option\src\interfaces\ITimeswapV2Option.sol
  169: function burn(TimeswapV2OptionBurnParam calldata param) e
  190: function collect(TimeswapV2OptionCollectParam calldata pa

v2-option\src\interfaces\ITimeswapV2OptionFactory.sol
  28: function getByIndex(uint256 id) external view returns (add

v2-pool\src\interfaces\callbacks\ITimeswapV2PoolBurnCallback.sol
  13: function timeswapV2PoolBurnChoiceCallback(TimeswapV2PoolBu

v2-pool\src\interfaces\callbacks\ITimeswapV2PoolMintCallback.sol
  14: function timeswapV2PoolMintChoiceCallback(TimeswapV2PoolMi
  18: function timeswapV2PoolMintCallback(TimeswapV2PoolMintCall
```

## [09] Missing NatSpec comments

NatSpec comments provide rich code documentation. The following functions are some examples that miss NatSpec comments. Please consider adding NatSpec comments for functions like these.

```
v2-option\src\TimeswapV2Option.sol
  56: function addOptionEnumerationIfNecessary(uint256 strike, u
  70: function blockTimestamp() internal view virtual returns (u

v2-pool\src\TimeswapV2Pool.sol
  57: function addPoolEnumerationIfNecessary(uint256 strike, uin
  66: function raiseGuard(uint256 strike, uint256 maturity) priv
  71: function lowerGuard(uint256 strike, uint256 maturity) priv
  82: function blockTimestamp(uint96 durationForward) internal v
  252: function mint(

v2-pool\src\TimeswapV2PoolDeployer.sol
  25: function deploy(address poolFactory, address optionPair, u

v2-pool\src\structs\LiquidityPosition.sol
  85: function collectTransactionFees(
```

## Gas Optimizations

For this contest, 24 reports were submitted by wardens detailing gas optimizations. The report highlighted below by **0xSmartContract** received the top score from the judge.

*The following wardens also submitted reports:* Rageur, atharvasama, Udsen, c3phas, Aymen0909, matrix_Owl, W_Max, shark, lllllll, 0x1f8b, 0xackermann, fatherOfBlocks, kaden, ReyAdmirado, Beepidibop, Viktor_Cortess, Iurii3, descharre, chaduke, RaymondFam, W0RR1O, Rolezn, *and* SaeedAlipoor01988 .

## Gas Optimizations Summary

| Number | Optimization Details | Context |
|---|---|---|
| [G-01] | Gas saving is achieved by removing the `delete` keyword (~60k) | 1 |
| [G-02] | Remove `checkDoesNotExist` function | 1 |
| [G-03] | Avoid using `state variable` in emit (130 gas) | 1 |
| [G-04] | Change `public` state variable visibility to `private` | 2 |

| Number | Optimization Details | Context |
|---|---|---|
| [G-05] | Save gas with the use of the import statement | 1 |
| [G-06] | Gas savings can be achieved by changing the model for assigning value to the structure (260 gas) | 2 |
| [G-07] | Using `delete` instead of setting struct `0` saves gas | 10 |
| [G-08] | In `div 512` function, `quotient 0` aggregate operation is used with unchecked to save gas | 1 |
| [G-09] | Avoid using external call | 1 |
| [G-10] | Gas overflow during iteration (DoS) | 1 |
| [G-11] | Move owner checks to a modifier for gas efficant | 2 |
| [G-12] | Use a more recent version of solidity | All contracts |
| [G-13] | Use nested if and, avoid multiple check combinations | 19 |
| [G-14] | ] Sort Solidity operations using short-circuit mode | 3 |
| [G-15] | `>=` costs less gas than `>` | 4 |
| [G-16] | Using UniswapV3 `mulDiv` function is gas-optimized | 1 |
| [G-17] | Using Openzeppelin Ownable2Step.sol is gas efficient | 1 |
| [G-18] | OpenZeppelin's ReentrancyGuard contract is gas-optimized | |
| [G-19] | Save gas with the use of the import statement | |
| [G-20] | Remove import `forge-std/console.sol` | 1 |
| [G-21] | Usage of uints/ints smaller than 32 bytes (256 bits) incurs overhead | 23 |
| [G-22] | Use `assembly` to write *address storage values* | 2 |
| [G-23] | Setting the *constructor* to `payable` | 8 |
| [G-24] | Avoid contract existence checks by using solidity version 0.8.10 or later | 57 |
| [G-25] | Optimize names to save gas | All contracts |
| [G-26] | Upgrade Solidity's optimizer | 3 |
| [G-27] | Open the optimizer | 1 |

Total: 27 issues

## [G-01] Gas saving is achieved by removing the `delete` keyword (~60k)

30k gas savings were made by removing the `delete` keyword. The reason for using the `delete` keyword here is to reset the struct values (set to default value) in every operation. However, the struct values do not need to be zero each time the function is run. Therefore, the  delete" key word is unnecessary. If it is removed, around 30k gas savings will be achieved.

There are two instances of the subject:

```
packages\v2-option\src\TimeswapV2OptionDeployer.sol:
  31      /// @return optionPair The address of the newly deploy
  32:     function deploy(address optionFactory, address token0,
  33:         parameter = Parameter({optionFactory: optionFactor
  34:
  35:         optionPair = address(new TimeswapV2Option{salt: ke
  36:
  37:         // save gas.
- 38:         delete parameter;
  39:     }
  40  }
```

```
packages\v2-pool\src\TimeswapV2PoolDeployer.sol:
  24
  25:     function deploy(address poolFactory, address optionPai
  26:         parameter = Parameter({poolFactory: poolFactory, o
  27:
  28:         poolPair = address(new TimeswapV2Pool{salt: keccak
  29:
- 30:         delete parameter;
  31:     }
  32  }
```

## [G-02] Remove `checkDoesNotExist` function

Using separate internal functions for non-repeating if blocks in more than one function wastes gas.

The `checkDoesNotExist` *internal* function in the `OptionPair.sol` contract is only used in the `create` function of the `TimeswapV2OptionFactory.sol` contract.

```
packages\v2-option\src\TimeswapV2OptionFactory.sol:
  43:     function create(address token0, address token1) externa
  44:         if (token0 == address(0)) Error.zeroAddress();
  45:         if (token1 == address(0)) Error.zeroAddress();
  46:         OptionPairLibrary.checkCorrectFormat(token0, token1
  47:
  48:         optionPair = optionPairs[token0][token1];
  49:         OptionPairLibrary.checkDoesNotExist(token0, token1
  50:
  51:         optionPair = deploy(address(this), token0, token1)
  52:
  53:         optionPairs[token0][token1] = optionPair;
  54:
  55:         emit Create(msg.sender, token0, token1, optionPair
  56:     }
  57 }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2OptionFactory.sol#L43-L57

**Recommendation:**

I suggest that the check on line 50 be done by adding the **if** block as follows.

*packages\v2-option\src\TimeswapV2OptionFactory.sol*

```
  43:     function create(address token0, address token1) externa
  44:         if (token0 == address(0)) Error.zeroAddress();
  45:         if (token1 == address(0)) Error.zeroAddress();
  46:         OptionPairLibrary.checkCorrectFormat(token0, token1
  47:
  48:         optionPair = optionPairs[token0][token1];
- 49:         OptionPairLibrary.checkDoesNotExist(token0, token1
+                 if (optionPair != address(0)) revert OptionPa
  50:
```

```
51:         optionPair = deploy(address(this), token0, token1)
52:
53:         optionPairs[token0][token1] = optionPair;
54:
55:         emit Create(msg.sender, token0, token1, optionPair
56:     }
57  }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2OptionFactory.sol#L43-L57

## [G-03] Avoid using `state variable` in emit (130 gas)

Using a state variable in `SetOwner` emits wastes gas.

1 result - 1 file:

```
packages\v2-pool\src\base\OwnableTwoSteps.sol:
  23     function setPendingOwner(address chosenPendingOwner) ex
  24         Ownership.checkIfOwner(owner);
  25
  26:        if (chosenPendingOwner == address(0)) Error.zeroAdd
  27         chosenPendingOwner.checkIfAlreadyOwner(owner);
  28
  29         pendingOwner = chosenPendingOwner;
  30
  31:         emit SetOwner(pendingOwner);
  32:     }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/base/OwnableTwoSteps.sol#L31

If the following recommendation is taken into account, 130 gas is saved.

```
packages\v2-pool\src\base\OwnableTwoSteps.sol:
 23     function setPendingOwner(address chosenPendingOwner) ext
 24         Ownership.checkIfOwner(owner);
 25
 26         if (chosenPendingOwner == address(0)) Error.zeroAdd
 27         chosenPendingOwner.checkIfAlreadyOwner(owner);
```

```
  28
+ 31:          emit SetOwner(chosenPendingOwner);
  29          pendingOwner = chosenPendingOwner;
  30
- 31:          emit SetOwner(pendingOwner);
  32      }
```

## [G-04] Change `public` state variable visibility to `private`

If it is preferred to change the visibility of the `owner` and `pendingOwnerstate` state variables to `private` , this will save significant gas.

2 result - 1 file:

```
packages\v2-pool\src\base\OwnableTwoSteps.sol:
  14:     address public override owner;

  16:     address public override pendingOwner;
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/base/OwnableTwoSteps.sol#L14-L16

## [G-05] Save gas with the use of the import statement

While the following two critical fee values are assigned in the constructor, there is no zero value control. This means that if both state variables are started with a possible value of `0` , the contract must be deployed again. This possibility means gas consumption.

Zero value control is the most error-prone value control since zero value is assigned in case of no value entry due to EVM design.

In addition, since the immutable value will be changed once, adding a zero value control does not cause high gas consumption.

```
packages\v2-pool\src\TimeswapV2PoolFactory.sol:
  37:     constructor(address chosenOwner, uint256 chosenTransac
  38:         if (chosenTransactionFee > type(uint16).max) rever
```

```
39:         if (chosenProtocolFee > type(uint16).max) revert I
40:
41:         transactionFee = chosenTransactionFee;
42:         protocolFee = chosenProtocolFee;
43:     }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2PoolFactory.sol#L37-L43

**Recommendation:**

It is recommended to perform a zero value check for critical value assignments.

Add zero check for immutable values when assigning values in critical constructor.

## [G-06] Gas savings can be achieved by changing the model for assigning value to the structure (260 gas)

By changing the pattern of assigning value to the structure, gas savings of `~130` per instance are achieved. In addition, this use will provide significant savings in distribution costs.

There are two examples of this issue:

```
packages\v2-pool\src\TimeswapV2PoolDeployer.sol:
  25:     function distribution(pool Factory address, option pai
  26:         parameter = Parameter({poolFactory: poolFactory, o
```

```
packages\v2-option\src\TimeswapV2OptionDeployer.sol:
  32:     function deploy(address optionFactory, address identif
  33:         parameter = Parameter({optionFactory:optionFactory
```

The following model, which is more gas efficient, can be preferred to assign value to the building elements.

```
packages\v2-option\src\TimeswapV2OptionDeployer.sol:
  32:     function deploy(address optionFactory, address identif
```

```
- 33:          parameter = Parameter({optionFactory: optionsFacto
+              parameter.optionFactory = optionFactory;
+              parameter.token0 = token0;
+              parameter.token1 = token1;
```

## [G-07] Using `delete` instead of setting struct `0` saves gas

10 results - 2 files:

```
packages\v2-pool\src\structs\LiquidityPosition.sol:
   93:    liquidityPosition.long0Fees = 0;

  101:    liquidityPosition.long1Fees = 0;

  109:    liquidityPosition.shortFees = 0;
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/structs/LiquidityPosition.sol#L93

```
packages\v2-pool\src\structs\Pool.sol:
  228:    pool.long0ProtocolFees = 0;

  236:    pool.long1ProtocolFees = 0;

  244:    pool.shortProtocolFees = 0;

  633:    pool.long0Balance = 0;

  646:    pool.long1Balance = 0;

  685:    pool.long1Balance = 0;

  706:    pool.long0Balance = 0;
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/structs/Pool.sol#L228

**Recommendation code:**

```
packages\v2-pool\src\structs\Pool.sol#L228
-  228:    pool.long0ProtocolFees = 0;
+ 228:    delete pool.long0ProtocolFees;
```

## [G-08] In `div 512` function, `quotient 0` aggregate operation is used with unchecked to save gas

```
packages/v2-library/src/FullMath.sol:
  158:     function div512(uint256 dividend0, uint256 dividend1,
  159:         (quotient0, quotient1) = div512(dividend0, divide
  160:
  161:         if (roundUp) {
  162:             (uint256 productA0, uint256 productA1) = mul5
  163:             productA1 += (quotient1 * divisor);
  164:             if (dividend1 > productA1 || dividend0 > prod
  165:                 if (quotient0 == type(uint256).max) {
  166:                     quotient0 = 0;
  167:                     quotient1++;
- 168:                 } else quotient0++;
+ 168:                 } else
+                        unchecked {
+                         quotient0++;
+                        }
  169:             }
  170:         }
  171:     }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/FullMath.sol#L165-L168

## [G-09] Avoid using external call

An if block check can be added as follows. With this control, gas saving is achieved by avoiding the use of external calls.

```
packages/v2-pool/src/base/OwnableTwoSteps.sol:
  22      /// @inheritdoc IOwnableTwoSteps
  23:     function setPendingOwner(address chosenPendingOwner) e
- 24:         Ownership.checkIfOwner(owner);
```

```
+ 24:         if (msg.sender == owner) revert NotOwner();
  25:
  26:         if (chosenPendingOwner == address(0)) Error.zeroAd
  27:         chosenPendingOwner.checkIfAlreadyOwner(owner);
  28:
  29:         pendingOwner = chosenPendingOwner;
  30:
  31:         emit SetOwner(pendingOwner);
  32:     }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/base/OwnableTwoSteps.sol#L24

## [G-10] Gas overflow during iteration (DoS)

Each iteration of the cycle requires a gas flow. A moment may come when more gas is required than it is allocated to record one block. In this case, all iterations of the loop will fail.

```
packages/v2-option/src/structs/Process.sol:
  32      /// @param isAddToken1 IsAddToken1 if true. IsSubToken
  33:     function updateProcess(Process[] storage processing, u
+              require(processing.length.length() < maxProcessin
  34:         for (uint256 i; i < processing.length; ) {
  35:             Process storage process = processing[i];
  36:
  37:             if (token0Amount != 0) process.balance0Target
  38:
  39:             if (token1Amount != 0) process.balance1Target
  40:
  41:             unchecked {
  42:                 i++;
  43:             }
  44:         }
  45:      }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/structs/Process.sol#L33-L45

## [G-11] Move owner checks to a modifier for gas efficant

It's better to use a modifier for simple owner checks for an easier inspection of functions. This is also more gas efficient as it does not control with external call.

**The part where `owner` is defined:**

```
packages/v2-library/src/Ownership.sol:
  22:     function checkIfOwner(address owner) internal view {
  23         if (msg.sender != owner) revert NotTheOwner(msg.se
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/Ownership.sol#L22-L23

2 results 2 files:

```
packages/v2-pool/src/TimeswapV2Pool.sol:
  189:         ITimeswapV2PoolFactory(poolFactory).owner().check
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2Pool.sol#L189

```
packages/v2-pool/src/base/OwnableTwoSteps.sol:
  23     function setPendingOwner(address chosenPendingOwner) e
  24:        Ownership.checkIfOwner(owner);
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/base/OwnableTwoSteps.sol#L23-L24

## [G-12] Use a more recent version of solidity

> Solidity 0.8.10 has a useful change that reduced gas costs of external calls which expect a return value.

> In 0.8.15 the conditions necessary for inlining are relaxed. Benchmarks show that the change significantly decreases the bytecode size (which impacts the deployment cost) while the effect on the runtime gas usage is smaller.

> In 0.8.17 prevent the incorrect removal of storage writes before calls to Yul functions that conditionally terminate the external EVM call; Simplify the starting offset of zero-length operations to zero. More efficient overflow checks for multiplication.

The version of 70 contracts included in the scope is `0.8.8`. I recommend that you upgrade the versions of all contracts in scope to the latest version of robustness, '0.8.17'.

## [G-13] Use nested if and, avoid multiple check combinations

Using nested is cheaper than using && multiple check combinations. There are more advantages, such as easier to read code and better coverage reports.

19 results - 9 files:

```
packages\v2-library\src\CatchError.sol:
  15:     if ((length - 4) % 32 == 0 && bytes4(reason) == selecto
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/CatchError.sol#L15

```
packages\v2-library\src\FullMath.sol:
  257:     if (roundUp && mulmod(multiplicand, multiplier, divisor
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/FullMath.sol#L257

```
packages\v2-library\src\Math.sol:
  51:     if (roundUp && dividend % divisor != 0) quotient++;

  62:     if (roundUp && dividend % (1 << divisorBit) != 0) quoti

  81:     if (roundUp && value % result != 0) result++;
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/Math.sol#L51

```
packages\v2-option\src\structs\Param.sol:
  111:   if (param.amount0 == 0 && param.amount1 == 0) Error.zer

  124:   if (param.amount0 == 0 && param.amount1 == 0) Error.zer
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/structs/Param.sol#L111

```
packages\v2-pool\src\TimeswapV2Pool.sol:
  167:   if (long0Fees == 0 && long1Fees == 0 && shortFees == 0)
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2Pool.sol#L167

```
packages\v2-pool\src\libraries\ConstantProduct.sol:
  411:   if (a11 == 0 && a01.unsafeAdd(a10) >= a01) {
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/libraries/ConstantProduct.sol#L411

```
packages\v2-pool\src\structs\Param.sol:
  136:   if (param.long0Requested == 0 && param.long1Requested =
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/structs/Param.sol#L136

```
packages\v2-token\src\base\ERC1155Enumerable.sol:
  60:    if (_idTotalSupply[id] == 0 && _additionalConditionAddT

  64:    if (to != address(0) && to != from) {

  65:    if (balanceOf(to, id) == 0 && _additionalConditionAddTo

  94:    if (_idTotalSupply[id] == 0 && _additionalConditionRemo

  98:    if (from != address(0) && from != to) {
```

```
99:     if (balanceOf(from, id) == 0 && _additionalConditionRem
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/base/ERC1155Enumerable.sol#L60

```
packages\v2-token\src\structs\Param.sol:
  121:   if (param.long0Amount == 0 && param.long1Amount == 0 &&

  128:   if (param.long0Amount == 0 && param.long1Amount == 0 &&

  149:   if (param.long0FeesDesired == 0 && param.long1FeesDesir
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/structs/Param.sol#L121

**Recomendation Code:**

```
protocol\contracts\plugins\aave\StaticATokenLM.sol#L422:
- 149:   if (param.long0FeesDesired == 0 && param.long1FeesDesir
+           if (param.long0FeesDesired == 0) {
+               if (param.long1FeesDesired == 0) {
+                   if (param.shortFeesDesired == 0) {
+                       Error.zeroInput();
+       }
+               }
+           }
```

## [G-14] Sort Solidity operations using short-circuit mode

Short-circuiting is a solidity contract development model that uses `OR/AND` logic to sequence different cost operations. It puts low gas cost operations in the front and high gas cost operations in the back, so that if the front is low, if the cost operation is feasible, you can skip (short-circuit) the subsequent high-cost Ethereum virtual machine operation.

```
//f(x) is a low gas cost operation
//g(y) is a high gas cost operation
```

```
//Sort operations with different gas costs as follows
f(x) || g(y)
f(x) && g(y)
```

3 results - 3 files:

```
packages\v2-pool\src\libraries\ConstantProduct.sol:
  298:     if (product.div(longAmount, false) != rate || product
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/libraries/ConstantProduct.sol#L298

```
packages\v2-library\src\CatchError.sol:
  15:     if ((length - 4) % 32 == 0 && bytes4(reason) == selecto
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/CatchError.sol#L15

```
packages\v2-library\src\FullMath.sol:
  68:     if (subtrahend1 > minuend1 || (subtrahend1 == minuend1
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/FullMath.sol#L68

## [G-15] >= costs less gas than >

The compiler uses opcodes GT and ISZERO for solidity code that uses >, but only requires LT for >=, which saves 3 gas

4 results - 2 files:

```
packages\v2-library\src\Math.sol:
  89:         return value1 < value2 ? value1 : value2;
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/Math.sol#L89

```
packages\v2-library\src\StrikeConversion.sol:
  27:         return strike > type(uint128).max ? (toOne ? conve

  36:         return strike > type(uint128).max ? amount0 + conv

  48:             strike > type(uint128).max
  49                 ? (zeroToOne ? convert(base - amount, strik
  50                 : (zeroToOne ? base - convert(amount, strik
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/StrikeConversion.sol#L27

## [G-16] Using UniswapV3 `mulDiv` function is gas-optimized

```
packages/v2-library/src/FullMath.sol:
  180     /// @return result The result.
  181:    function mulDiv(uint256 multiplicand, uint256 multipl
  182         (uint256 product0, uint256 product1) = mul512(mult
  183
  184         // Handle non-overflow cases, 256 by 256 division
```

Reference: https://github.com/Uniswap/v3-core/blob/412d9b236a1e75a98568d49b1aeb21e3a1430544/contracts/libraries/FullMath.sol#L14

Reference: https://xn--2-umb.com/21/muldiv/

## [G-17] Using Openzeppelin Ownable2Step.sol is gas efficient

The project makes secure Owner changes with OwnableTwoStep.

The project's `acceptOwner()` function:

```
packages\v2-pool\src\base\OwnableTwoSteps.sol:
```

```
34      /// @inheritdoc IOwnableTwoSteps
35      function acceptOwner() external override {
36          msg.sender.checkIfPendingOwner(pendingOwner);
37
38          owner = msg.sender;
39          delete pendingOwner;
40
41          emit AcceptOwner(msg.sender);
42      }
43  }
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/base/OwnableTwoSteps.sol#L35-L43

However, I recommend using the more gas-optimized Openzeppelin in Ownable2Step.sol.

Openzeppelin acceptOwner() function:

```
function acceptOwnership() public virtual {
        address sender = _msgSender();
        require(pendingOwner() == sender, "Ownable2Step: caller
        _transferOwnership(sender);
    }
```

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable2Step.sol

## [G-18] OpenZeppelin's ReentrancyGuard contract is gas-optimized

```
packages/v2-pool/src/libraries/ReentrancyGuard.sol:
  11:     /// @dev The initial state which must be change to NOT_
  12:     uint96 internal constant NOT_INTERACTED = 0;
  13:
  14:     /// @dev The initial and ending state of balanceTarget
  15:     uint96 internal constant NOT_ENTERED = 1;
  16:
  17:     /// @dev The state where the contract is currently bei
```

```
18:     uint96 internal constant ENTERED = 2;
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/libraries/ReentrancyGuard.sol#L12-L18

I recommend using the gas-optimized OpenZeppelin ReentrancyGuard.sol contract.

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol

## [G-19] Save gas with the use of the import statement

With the import statement, it saves gas to specifically import only the parts of the contracts, not the complete ones.

```
packages/v2-token/src/interfaces/IERC1155Enumerable.sol:

6: import "@openzeppelin/contracts/token/ERC1155/IERC1155.sol";
```

**Description:**

Solidity code is also cleaner in another way that might not be noticeable: the struct Point. We were importing it previously with global import but not using it. The Point struct `polluted the source code` with an unnecessary object we were not using because we did not need it.

This was breaking the rule of modularity and modular programming: `only import what you need` Specific imports with curly braces allow us to apply this rule better.

**Recommendation:**

```
import {contract1 , contract2} from "filename.sol";
```

A good example from the ArtGobblers project;

import {Owned} from “solmate/auth/Owned.sol”;

import {ERC721} from “solmate/tokens/ERC721.sol”;

import {LibString} from “solmate/utils/LibString.sol”;

import {MerkleProofLib} from “solmate/utils/MerkleProofLib.sol”;

import {FixedPointMathLib} from “solmate/utils/FixedPointMathLib.sol”;

import {ERC1155, ERC1155TokenReceiver} from “solmate/tokens/ERC1155.sol”;

import {toWadUnsafe, toDaysWadUnsafe} from “solmate/utils/SignedWadMath.sol”;

## [G-20] Remove import `forge-std/console.sol`

It’s used to print the values of variables while running tests to help debug and see what’s happening inside your contracts But since it’s a development tool, it serves no purpose on mainnet.

1 result - 1 file:

```
packages\v2-token\src\TimeswapV2Token.sol:
  5: import "forge-std/console.sol";
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2Token.sol#L5

Also, the following code block should be removed along with the removal of `forge-std/console.sol`.

```
packages/v2-token/src/TimeswapV2Token.sol:
  109:             console.log("reaches right before mint in tim
```

**Recommendation:**

Use only for tests

## [G-21] Usage of uints/ints smaller than 32 bytes (256 bits) incurs overhead

When using elements that are smaller than 32 bytes, your contracts gas usage may be higher. This is because the EVM operates on 32 bytes at a time. Therefore, if the element is smaller than that, the EVM must use more operations in order to reduce the size of the element from 32 bytes to the desired size.

https://docs.soliditylang.org/en/v0.8.11/internals/layout_in_storage.html

Use a larger size then downcast where needed.

23 results - 4 files:

```
packages\v2-library\src\SafeCast.sol:
  20:     result = uint16(value);

  38:     result = uint160(value);
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-library/src/SafeCast.sol#L20

```
packages\v2-pool\src\TimeswapV2Pool.sol:
  104:     return pools[strike][maturity].liquidity;

  109:     return pools[strike][maturity].sqrtInterestRate;

  114:     return pools[strike][maturity].liquidityPositions[owner
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2Pool.sol#L104

```
packages\v2-pool\src\libraries\ConstantProduct.sol:
  67:     liquidityAmount = getLiquidityGivenLong(rate, longAmoun

  82:     liquidityAmount = getLiquidityGivenShort(rate, shortAmo

  104:     liquidityAmount = getLiquidityGivenLong(rate, amount, !
```

```
110:    liquidityAmount = getLiquidityGivenShort(rate, amount, 

142:    newRate = isAdd ? rate + deltaRate : rate - deltaRate;

174:    newRate = getNewSqrtInterestRateGivenLong(liquidity, ra

206:    (newRate, deltaRate) = getNewSqrtInterestRateGivenShort

260:    return FullMath.mulDiv(uint256(rate), longAmount, uint2

269:    return FullMath.mulDiv(shortAmount, uint256(1) << 192, 

296:    return numerator.div(denominator2, true).toUint160();

316:    deltaRate = FullMath.mulDiv(shortAmount, uint256(1) << 
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/libraries/ConstantProduct.sol#L67

```
packages\v2-pool\src\structs\LiquidityPosition.sol:
  66:    liquidityPosition.liquidity += liquidityAmount;

  76:    liquidityPosition.liquidity -= liquidityAmount;

  207:   pool.sqrtInterestRate = rate;

  308:   liquidityAmount = param.delta.toUint160(),

  398:   liquidityAmount = param.delta.toUint160(),

  486:   param.delta.toUint160(),

  572:   param.delta.toUint160(),
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/structs/LiquidityPosition.sol#L66

## [G-22] Use `assembly` to write *address storage values*

2 results - 2 files:

```
packages\v2-pool\src\base\OwnableTwoSteps.sol:
  18     constructor(address chosenOwner) {
  19:          owner = chosenOwner;```

19 https:
20
21
22 ```solidity
23 packages\v2-token\src\TimeswapV2Token.sol:
24   41     constructor(address chosenOptionFactory) ERC1155("Tim
25   42:           optionFactory = chosenOptionFactory;
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2Token.sol#L42

**Recommendation Code:**

```
  41     constructor(address chosenOptionFactory) ERC1155("Times
- 42:           optionFactory = chosenOptionFactory;
+                  assembly {
+                      sstore(optionFactory.slot, chosenOptionFa
+                  }
```

## [G-23] Setting the *constructor* to `payable`

You can cut out 10 opcodes in the creation-time EVM bytecode if you declare a constructor payable. Making the constructor payable eliminates the need for an initial check of `msg.value == 0` and saves `13 gas` on deployment with no security risks.

8 results - 8 files:

```
packages\v2-option\src\NoDelegateCall.sol:
  19:     constructor() {
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/NoDelegateCall.sol#L19

```
packages\v2-option\src\TimeswapV2Option.sol:
  65:     constructor() NoDelegateCall() {
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2Option.sol#L65

```
packages\v2-pool\src\NoDelegateCall.sol:
  19:     constructor() {
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/NoDelegateCall.sol#L19

```
packages\v2-pool\src\TimeswapV2Pool.sol:
  77:     constructor() NoDelegateCall() {
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2Pool.sol#L77

```
packages\v2-pool\src\TimeswapV2PoolFactory.sol:
  37:     constructor(address chosenOwner, uint256 chosenTransac
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2PoolFactory.sol#L37

```
packages\v2-pool\src\base\OwnableTwoSteps.sol:
  18:     constructor(address chosenOwner) {
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/base/OwnableTwoSteps.sol#L18

```
packages\v2-token\src\TimeswapV2LiquidityToken.sol:
  36:     constructor(address chosenOptionFactory, address chose
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2LiquidityToken.sol#L36

```
packages\v2-token\src\TimeswapV2Token.sol:
  41:     constructor(address chosenOptionFactory) ERC1155("Time
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2Token.sol#L41

**Recommendation:**

Set the constructor to `payable`

## [G-24] Avoid contract existence checks by using solidity version 0.8.10 or later

Prior to 0.8.10 the compiler inserted extra code, including EXTCODESIZE (100 gas), to check for contract existence for external calls. In more recent solidity versions, the compiler will not insert these checks if the external call has a return value

57 results - 6 files:

```
packages\v2-token\src\TimeswapV2LiquidityToken.sol:
   66:     amount = balanceOf(owner, _timeswapV2LiquidityTokenPos
   71:     safeTransferFrom(from, to, _timeswapV2LiquidityTokenPo
   81:     uint256 id = _timeswapV2LiquidityTokenPositionIds[posi
  109:     bytes32 key = timeswapV2LiquidityTokenPosition.toKey()
  153:     bytes32 key = TimeswapV2LiquidityTokenPosition({token0
  185:     bytes32 key = TimeswapV2LiquidityTokenPosition({token0
  125:     uint160 liquidityBalanceTarget = ITimeswapV2Pool(poolP
  143:     Error.checkEnough(ITimeswapV2Pool(poolPair).liquidityO
  125:     uint160 liquidityBalanceTarget = ITimeswapV2Pool(poolP
```

```
143:     Error.checkEnough(ITimeswapV2Pool(poolPair).liquidityO
131:     data = ITimeswapV2LiquidityTokenMintCallback(msg.sende
163:     data = ITimeswapV2LiquidityTokenBurnCallback(msg.sende
160:     ITimeswapV2Pool(poolPair).transferLiquidity(param.stri
193:     ITimeswapV2Pool(poolPair).transferFees(param.strike, p
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2LiquidityToken.sol#L66

```
packages\v2-token\src\TimeswapV2Token.sol:
 67:     amount = ERC1155.balanceOf(owner, _timeswapV2TokenPosi
 72:     safeTransferFrom(from, to, _timeswapV2TokenPositionIds
 97:     bytes32 key = timeswapV2TokenPosition.toKey();
127:     bytes32 key = timeswapV2TokenPosition.toKey();
156:     bytes32 key = timeswapV2TokenPosition.toKey();
240:     _burn(msg.sender, _timeswapV2TokenPositionIds[timeswap
254:     _burn(msg.sender, _timeswapV2TokenPositionIds[timeswap
268:     _burn(msg.sender, _timeswapV2TokenPositionIds[timeswap
 87:     long0BalanceTarget = ITimeswapV2Option(optionPair).pos
117:     long1BalanceTarget = ITimeswapV2Option(optionPair).pos
146:     shortBalanceTarget = ITimeswapV2Option(optionPair).pos
186:     if (param.long0Amount != 0) Error.checkEnough(ITimeswa
189:     if (param.long1Amount != 0) Error.checkEnough(ITimeswa
192:     if (param.shortAmount != 0) Error.checkEnough(ITimeswa
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-token/src/TimeswapV2Token.sol#L67

```
packages\v2-pool\src\TimeswapV2Pool.sol:
  189:     ITimeswapV2PoolFactory(poolFactory).owner().checkIfOwn

  267:     if (long0Amount != 0) long0BalanceTarget = ITimeswapV2

  271:     if (long1Amount != 0) long1BalanceTarget = ITimeswapV2

  274:     uint256 shortBalanceTarget = ITimeswapV2Option(optionP

  293:     if (long0Amount != 0) Error.checkEnough(ITimeswapV2Opt

  295:     if (long1Amount != 0) Error.checkEnough(ITimeswapV2Opt

  297:     Error.checkEnough(ITimeswapV2Option(optionPair).positi

  393:     if (long0Amount != 0) long0BalanceTarget = ITimeswapV2

  397:     if (long1Amount != 0) long1BalanceTarget = ITimeswapV2

  411:     if (long0Amount != 0) Error.checkEnough(ITimeswapV2Opt

  413:     if (long1Amount != 0) Error.checkEnough(ITimeswapV2Opt

  444:     uint256 balanceTarget = ITimeswapV2Option(optionPair).

  461:     Error.checkEnough(ITimeswapV2Option(optionPair).positi

  479:     uint256 balanceTarget = ITimeswapV2Option(optionPair).

  511:     ITimeswapV2Option(optionPair).positionOf(param.strike,
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/TimeswapV2Pool.sol#L189

```
packages\v2-option\src\libraries\OptionFactory.sol:
  28:     optionPair = ITimeswapV2OptionFactory(optionFactory).g
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/libraries/OptionFactory.sol#L28

```
packages\v2-pool\src\libraries\PoolFactory.sol:
  32:     poolPair = ITimeswapV2PoolFactory(poolFactory).get(opt

  46:     poolPair = ITimeswapV2PoolFactory(poolFactory).get(opt
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-pool/src/libraries/PoolFactory.sol#L32

```
packages\v2-option\src\TimeswapV2Option.sol:
  128:     IERC20(token0).balanceOf(address(this)) + token0AndLon

  129:     IERC20(token1).balanceOf(address(this)) + token1AndLon

  145:     if (token0AndLong0Amount != 0) Error.checkEnough(IERC2

  148:     if (token1AndLong1Amount != 0) Error.checkEnough(IERC2

  215:     param.isLong0ToLong1 ? IERC20(token0).balanceOf(addres

  216:     param.isLong0ToLong1 ? IERC20(token1).balanceOf(addres

  172:     if (token0AndLong0Amount != 0) IERC20(token0).safeTran

  175:     if (token1AndLong1Amount != 0) IERC20(token1).safeTran

  220:     IERC20(param.isLong0ToLong1 ? token0 : token1).safeTra

  259:     if (token0Amount != 0) IERC20(token0).safeTransfer(par

  262:     if (token1Amount != 0) IERC20(token1).safeTransfer(par
```

https://github.com/code-423n4/2023-01-timeswap/blob/main/packages/v2-option/src/TimeswapV2Option.sol#L128

## [G-25] Optimize names to save gas

Contracts most called functions could simply save gas by function ordering via `Method ID`. Calling a function at runtime will be cheaper if the function is positioned earlier in the order (has a relatively lower Method ID) because `22 gas` are added to the cost of a function for every position that came before it. The caller can save on gas if you prioritize most called functions.

**Context:**

All Contracts

**Recommendation:**

Find a lower `method ID` name for the most called functions for example Call() vs. Call1() is cheaper by `22 gas`.

For example, the function IDs in the `TimeswapV2Pool.sol` contract will be the most used; A lower method ID may be given.

**Proof of Concept:**

https://medium.com/joyso/solidity-how-does-function-name-affect-gas-consumption-in-smart-contract-47d270d8ac92

TimeswapV2Pool.sol function names can be named and sorted according to METHOD ID

```
Sighash   |   Function Signature
========================
fbddf051  =>  addPoolEnumerationIfNecessary(uint256,uint256)
1ea3a4eb  =>  raiseGuard(uint256,uint256)
3a9e8dd9  =>  lowerGuard(uint256,uint256)
f4f89897  =>  blockTimestamp(uint96)
2d883a73  =>  getByIndex(uint256)
6f682a53  =>  numberOfPools()
5c81c9b8  =>  hasLiquidity(uint256,uint256)
b15044ac  =>  totalLiquidity(uint256,uint256)
a8f403b7  =>  sqrtInterestRate(uint256,uint256)
f78333a3  =>  liquidityOf(uint256,uint256,address)
647284f5  =>  feeGrowth(uint256,uint256)
6c867790  =>  feesEarnedOf(uint256,uint256,address)
```

```
72f8f85c  =>  protocolFeesEarned(uint256,uint256)
7ffd3a70  =>  totalLongBalance(uint256,uint256)
3a9d71e7  =>  totalLongBalanceAdjustFees(uint256,uint256)
8fdc5c99  =>  totalPositions(uint256,uint256)
c0e0c4c6  =>  transferLiquidity(uint256,uint256,address,uint160)
d7e2c24a  =>  transferFees(uint256,uint256,address,uint256,uint2
ad118b02  =>  initialize(uint256,uint256,uint160)
47b46959  =>  collectProtocolFees(TimeswapV2PoolCollectParam)
53fa956e  =>  collectTransactionFees(TimeswapV2PoolCollectParam)
55e305f3  =>  collect(uint256,uint256,address,address,address,ui
0150ca41  =>  mint(TimeswapV2PoolMintParam)
6da9a2a4  =>  mint(TimeswapV2PoolMintParam,uint96)
df52795d  =>  mint(TimeswapV2PoolMintParam,bool,uint96)
13576d77  =>  burn(TimeswapV2PoolBurnParam)
731d4f67  =>  burn(TimeswapV2PoolBurnParam,uint96)
bd4952bd  =>  burn(TimeswapV2PoolBurnParam,bool,uint96)
5d0ea1f5  =>  deleverage(TimeswapV2PoolDeleverageParam)
2e1b22ce  =>  deleverage(TimeswapV2PoolDeleverageParam,uint96)
cde7bf11  =>  deleverage(TimeswapV2PoolDeleverageParam,bool,uint
a97a4f78  =>  leverage(TimeswapV2PoolLeverageParam)
37aaeff8  =>  leverage(TimeswapV2PoolLeverageParam,uint96)
b8be2a15  =>  leverage(TimeswapV2PoolLeverageParam,bool,uint96)
c993c3fa  =>  rebalance(TimeswapV2PoolRebalanceParam)
```

## [G-26] Upgrade Solidity's optimizer

Make sure Solidity's optimizer is enabled. It reduces gas costs. If you want to gas optimize for contract deployment (costs less to deploy a contract) then set the Solidity optimizer at a low number. If you want to optimize for run-time gas costs (when functions are called on a contract) then set the optimizer to a high number.

Set the optimization value higher than 800 in your hardhat.config.ts file.

3 results - 3 files:

```
packages\v2-option\hardhat.config.ts:
  27: const config: HardhatUserConfig = {
  28:   paths: {
  29:     sources: "./src",
  30:   },
  31:   solidity: {
  32:     version: "0.8.8",
  33:     settings: {
```

```
34:       optimizer: {
35:         enabled: true,
36:         runs: 200,
37:       },
38:     },
39:   },
```

```
packages\v2-pool\hardhat.config.ts:
  26: const config: HardhatUserConfig = {
  27:   paths: {
  28:     sources: "./src",
  29:   },
  30:   solidity: {
  31:     version: "0.8.8",
  32:     settings: {
  33:       optimizer: {
  34:         enabled: true,
  35:         runs: 200,
  36:       },
  37:     },
  38:   },
```

```
packages\v2-token\hardhat.config.ts:
  26: const config: HardhatUserConfig = {
  27:   paths: {
  28:     sources: "./src",
  29:   },
  30:   solidity: {
  31:     version: "0.8.8",
  32:     settings: {
  33:       optimizer: {
  34:         enabled: true,
  35:         runs: 200,
  36:       },
  37:     },
  38:   },
```

## [G-27] Open the optimizer

Always use the Solidity optimizer to optimize gas costs. It's good practice to set the optimizer as high as possible until it no longer helps reduce gas costs in function calls.

This is advisable since function calls are intended to be executed many more times than contract deployment, which only happens once.

In the light of this information, I suggest you to open the optimizer for `v2-library`.

## Disclosures

C4 is an open organization governed by participants in the community.

C4 Contests incentivize the discovery of exploits, vulnerabilities, and bugs in smart contracts. Security researchers are rewarded at an increasing rate for finding higher-risk issues. Contest submissions are judged by a knowledgeable security researcher and solidity developer and disclosed to sponsoring developers. C4 does not conduct formal verification regarding the provided code but instead provides final verification.

C4 does not provide any guarantee or warranty regarding the security of this project. All smart contract software should be used at the sole risk and responsibility of users.

Top