Nested Finance contest Findings & Analysis Report

2022-09-22

Table of contents

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 Nested Finance smart contract system written in Solidity. The audit contest took place between June 15—June 18 2022.

Wardens

40 Wardens contributed reports to the Nested Finance contest:

1. 0xDjango
2. Meera
3. 0xNazgul
4. IllIllI
5. Chom
6. joestakey
7. simon135
8. PierrickGT
9. codexploder
10. MiloTruck
11. _Adam
12. delfin454000
13. fatherOfBlocks
14. oyc_109
15. TerrierLover
16. 0xf15ers (remora and twojoy)
17. BowTiedWardens (BowTiedHeron, BowTiedPickle, m4rio_eth, Dravee, and BowTiedFirefox)

18. Dravee
19. cccz
20. 0xkatana
21. ElKu
22. minhquanym
23. Waze
24. 0xFar5eer
25. hansfriese
26. cryptphi
27. c3phas
28. 0xKitsune
29. UnusualTurtle
30. robee
31. sach1r0
32. 0x1f8b
33. asutorufos
34. JC
35. Picodes
36. SooYa

This contest was judged by Jack the Pug.

Final report assembled by liveactionllama. Summary of low risk/non-critical reports and gas optimizations compiled by defsec.

Summary

The C4 analysis yielded an aggregated total of 1 unique vulnerabilities. Of these vulnerabilities, 0 received a risk rating in the category of HIGH severity and 1 received a risk rating in the category of MEDIUM severity.

Additionally, C4 analysis included 22 reports detailing issues with a risk rating of LOW severity or non-critical. There were also 27 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 Nested Finance contest repository, and is composed of 18 smart contracts written in the Solidity programming language and includes 1,733 lines of Solidity code.

Severity Criteria

C4 assesses the severity of disclosed vulnerabilities according to a methodology based on OWASP standards.

Vulnerabilities are divided into 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

Further information regarding the severity criteria referenced throughout the submission review process, please refer to the documentation provided on the C4 website.

Medium Risk Findings (1)

[M-01] User can bypass `entryFee` by sending arbitrary `calldata` to ParaSwap operator

Submitted by OxDjango

NestedFactory.sol#L466
ParaswapOperator.sol#L34

Any user is able to bypass the `entryFee` collection when using `NestedFactory.create()` by passing in arbitrary calldata when using the ParaSwap router. High level, a user can pass in calldata to swap from a miniscule amount of input token to an ERC777 with themselves as the recipient and will gain control of execution, at which time they can send a large amount of output token back to the Nested Factory.

If the user sends `1 wei` of input token, the Nested Factory will return an `entryFee` of `0` due to precision loss. The amount of output token returned to the contract via the direct transfer from the user will then be deposited in the vault.

Proof of Concept

Steps

- User calls `NestedFactory.create()` with a single input order. This input order will define the parameters of the call to Paraswap.
- The single order defines the following in pseudocode:

    1. inputToken: Any token, but we'll use address(0) ETH
    2. amount: 1 wei
    3. Order(operator=Paraswap, token=USDC, calldata=calldata)

The calldata used in the call to paraswap would swap from ETH to any ERC777 (NOT USDC), with an attack contract address set as the `beneficiary`***. Upon transferring the swapped ERC777 to the user's attack contract, the contract would immediately send e.g. 1,000,000 USDC directly back to the Nested Factory contract.***

- The Paraswap operator checks the balances of the buy and sell tokens. Note that the buy token is defined in the Order token parameter, not the calldata passed to Paraswap. Since the operator will check the balance of USDC, it looks like we've swapped 1 wei ETH for 1,000,000 USDC.
- The Paraswap operator returns the swap amounts back to Nested Factory.

- Nested Factory deposits the 1,000,000 USDC to the vault for the user without charging any `entryFee`.

NOTE: I use 1 wei as an extreme example. You would have to ensure that you're swapping at least enough to receive 1 wei of the ERC777 to transfer to the attack contract.

Recommended Mitigation Steps

Allowing a user to pass arbitrary call data to a router is risky because routers allow several paths for an attacker to gain control of execution. Originally, I believed this exploit to be possible simply by swapping to ETH, which would perform an external call to the `beneficiary`, but Paraswap actually only forwards 10,000 gas when performing ETH transfers. If Nested plans to include a vanilla Uniswap router operator, this would be an issue. Here is the Paraswap transfer logic:

```
    function transferTokens(
            address token,
            address payable destination,
            uint256 amount
        )
        internal
        {
            if (amount > 0) {
                if (token == ETH_ADDRESS) {
                    (bool result, ) = destination.call{value: amount
                    require(result, "Failed to transfer Ether");
                }
                else {
                    IERC20(token).safeTransfer(destination, amount);
                }
            }

        }
```

Therefore, it might be worth exploring the option of allowing the user to only choose from a list of predefined function signatures when making calls to Paraswap. The final `Order` param that is passed to the operator would be built within the contract by concatenating the function, input, and output tokens. Even then, if the output token truly is an ERC777, the user would be able to intercept control and directly transfer more of the ERC777.

A large-scale fix would be to charge the entry fee on the amount of output tokens after performing the swap. I'm not sure if this falls in line with Nested's plans though.

maximebrugel (Nested) acknowledged

Jack the Pug (judge) commented:

> I find this to be a valid Medium issue. I have a few things to add:

1. It applies not only to the ParaSwap operator but also the other operators, ie, 0x;
2. Not just the `entryFee`, the `exitFees` can also be bypassed in a similar way;
3. Not necessarily using a ERC777, the attacker can also use a malicious ERC20 they deployed on their own;

> The root cause is that:

> `entryFee` and `exitFees` should be charged on the token that gets in and out the `Reserve`, not the inputToken/outputToken of the swap.

Low Risk and Non-Critical Issues

For this contest, 22 reports were submitted by wardens detailing low risk and non-critical issues. The report submitted by **0xNazgul** received the top score from the judge.

The low risk and non-critical findings below include reports submitted by 0xNazgul, as well as: Meera, Chom, IllIllI, codexploder, joestakey, simon135, 0xf15ers, BowTiedWardens, Dravee, cccz, _Adam, 0xDjango, 0xFar5eer, delfin454000, fatherOfBlocks, hansfriese, MiloTruck, oyc_109, PierrickGT, TerrierLover, and cryptphi.

Low Risk Issue Summary

[L-01] Known vulnerabilities exist in currently used @openzeppelin/contracts version
[L-02] Impractical Entry/Exit fees are allowed
[L-03] poolCoinAmount validation
[L-04] Low level calls with solidity version 0.8.14 can result in optimiser bug.
[L-05] Unchecked return value of transferFrom can allow a user to withdraw native token for free.

[L-01] Known vulnerabilities exist in currently used @openzeppelin/contracts version

As some known vulnerabilities exist in the current @openzeppelin/contracts version, consider updating package.json with at least @openzeppelin/contracts@4.4.2 here:

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/package.json#L65

```
    "@openzeppelin/contracts": "^4.3.2",
```

While vulnerabilities are known, the current scope isn't affected (this might not hold true for the whole solution).

[L-02] Impractical Entry/Exit fees are allowed

https://github.com/code-423n4/2022-06-nested/blob/main/contracts/NestedFactory.sol#L159

It seems that Owner is allowed to set entry/exit fees to be 100% of amount. An entry fees of 100% will be impractical and will lead to all order amount to be gone in fees at `NestedFactory.sol#L378`

Proof of Concept

1. Admin call setEntryFees function and set _entryFees as 10000.
2. entryFees becomes 100%.
3. Now assume an order is submitted via _submitInOrders.
4. Entry fees will be deducted from amount spent.

5. Since fees is 100% so feesDeducted=amountSpent which means amountSpent effectively becomes 0 as all of it went for fees.

Recommended Mitigation Steps

Decide a max percentage of fees say 10% which can be charged and then change the require condition accordingly. Same logic need to be applied for exit fees

```
require(_entryFees <= MAX_PERCENTAGE, "NF: FEES_OVERFLOW");
```

[L-03] poolCoinAmount validation

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/StakingLPVaultHelpers.sol

poolCoinAmount must be 2, 3, 4. so, if it not fall in this range it should be reverted but now it doesn't. On every functions in this file add:

```
if (poolCoinAmount < 2 || poolCoinAmount > 4) revert
Change code to

// SPDX-License-Identifier: GPL-3.0-or-later
pragma solidity 0.8.14;

import "./../Withdrawer.sol";
import "./../libraries/ExchangeHelpers.sol";
import "./../libraries/CurveHelpers/CurveHelpers.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "./../interfaces/external/ICurvePool/ICurvePool.sol";
import "./../interfaces/external/ICurvePool/ICurvePoolETH.sol";
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol'
import "./../interfaces/external/IStakingVault/IStakingVault.sol
import "./../interfaces/external/ICurvePool/ICurvePoolNonETH.sol

error InvalidPoolCoinAmount(uint256 poolCoinAmount);

/// @notice Library for LP Staking Vaults deposit/withdraw
library StakingLPVaultHelpers {
    using SafeERC20 for IERC20;

    /// @dev  Add liquidity in a Curve pool with ETH and deposit
    ///       the LP token in a staking vault
    /// @param vault The staking vault address to deposit into
    /// @param pool The Curve pool to add liquitiy in
    /// @param lpToken The Curve pool LP token
    /// @param poolCoinAmount The number of token in the Curve p
    /// @param eth ETH address
    /// @param amount ETH amount to add in the Curve pool
    function _addLiquidityAndDepositETH(
        address vault,
        ICurvePoolETH pool,
        IERC20 lpToken,
        uint256 poolCoinAmount,
        address eth,
        uint256 amount
    ) internal {
        if (poolCoinAmount < 2 || poolCoinAmount > 4) revert Inv

        uint256 lpTokenBalanceBefore = lpToken.balanceOf(address

        if (poolCoinAmount == 2) {
            pool.add_liquidity{ value: amount }(CurveHelpers.get
        } else if (poolCoinAmount == 3) {
            pool.add_liquidity{ value: amount }(CurveHelpers.get
        } else {
            pool.add_liquidity{ value: amount }(CurveHelpers.get
        }

        uint256 lpTokenToDeposit = lpToken.balanceOf(address(thi
        ExchangeHelpers.setMaxAllowance(lpToken, vault);
        IStakingVault(vault).deposit(lpTokenToDeposit);
    }

    /// @dev  Add liquidity in a Curve pool and deposit
    ///       the LP token in a staking vault
    /// @param vault The staking vault address to deposit into
    /// @param pool The Curve pool to add liquitiy in
    /// @param lpToken The Curve pool lpToken
    /// @param poolCoinAmount The number of token in the Curve p
    /// @param token Token to add in the Curve pool liquidity
    /// @param amount Token amount to add in the Curve pool
    function _addLiquidityAndDeposit(
        address vault,
        ICurvePoolNonETH pool,
        IERC20 lpToken,
        uint256 poolCoinAmount,
        address token,
        uint256 amount
    ) internal {
        if (poolCoinAmount < 2 || poolCoinAmount > 4) revert Inv

        uint256 lpTokenBalanceBefore = lpToken.balanceOf(address
        ExchangeHelpers.setMaxAllowance(IERC20(token), address(p

        if (poolCoinAmount == 2) {
            pool.add_liquidity(CurveHelpers.getAmounts2Coins(poo
        } else if (poolCoinAmount == 3) {
            pool.add_liquidity(CurveHelpers.getAmounts3Coins(poo
        } else {
            pool.add_liquidity(CurveHelpers.getAmounts4Coins(poo
        }

        uint256 lpTokenToDeposit = lpToken.balanceOf(address(thi
        ExchangeHelpers.setMaxAllowance(lpToken, vault);
        IStakingVault(vault).deposit(lpTokenToDeposit);
    }

    /// @dev Withdraw the LP token from the staking vault and
    ///      remove the liquidity from the Curve pool
    /// @param vault The staking vault address to withdraw from
    /// @param amount The amount to withdraw
    /// @param pool The Curve pool to remove liquitiy from
    /// @param lpToken The Curve pool LP token
    /// @param poolCoinAmount The number of token in the Curve p
    /// @param outputToken Output token to receive
    function _withdrawAndRemoveLiquidity128(
        address vault,
        uint256 amount,
        ICurvePool pool,
        IERC20 lpToken,
        uint256 poolCoinAmount,
        address outputToken
    ) internal {
        if (poolCoinAmount < 2 || poolCoinAmount > 4) revert Inv

        uint256 lpTokenBalanceBefore = lpToken.balanceOf(address
        IStakingVault(vault).withdraw(amount);

        bool success = CurveHelpers.removeLiquidityOneCoin(
            pool,
            lpToken.balanceOf(address(this)) - lpTokenBalanceBef
            outputToken,
            poolCoinAmount,
            bytes4(keccak256(bytes("remove_liquidity_one_coin(ui
        );

        require(success, "SDCSO: CURVE_RM_LIQUIDITY_FAILED");
    }

    /// @dev Withdraw the LP token from the staking vault and
    ///      remove the liquidity from the Curve pool
    /// @param vault The staking vault address to withdraw from
    /// @param amount The amount to withdraw
    /// @param pool The Curve pool to remove liquitiy from
    /// @param lpToken The Curve pool LP token
    /// @param poolCoinAmount The number of token in the Curve p
    /// @param outputToken Output token to receive
    function _withdrawAndRemoveLiquidity256(
        address vault,
        uint256 amount,
        ICurvePool pool,
        IERC20 lpToken,
        uint256 poolCoinAmount,
        address outputToken
    ) internal {
        if (poolCoinAmount < 2 || poolCoinAmount > 4) revert Inv

        uint256 lpTokenBalanceBefore = lpToken.balanceOf(address
        IStakingVault(vault).withdraw(amount);

        bool success = CurveHelpers.removeLiquidityOneCoin(
            pool,
            lpToken.balanceOf(address(this)) - lpTokenBalanceBef
            outputToken,
            poolCoinAmount,
            bytes4(keccak256(bytes("remove_liquidity_one_coin(ui
        );

        require(success, "SDCSO: CURVE_RM_LIQUIDITY_FAILED");
    }
}
```

[L-04] Low level calls with solidity version 0.8.14 can result in optimiser bug

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governan ce/OwnerProxy.sol#L20

The protocol is using low level calls with solidity version 0.8.14 which can result in optimizer bug.

Proof of Concept

See POC from Certora

Recommended Mitigation Steps

Consider upgrading to solidity 0.8.15.

[L-05] Unchecked return value of transferFrom can allow a user to withdraw native token for free

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/OwnerProxy.sol#L20

The Withdrawer contract has a function withdraw() which calls an unsafe transferFrom(). A call to transferFrom is frequently done without checking the results. For certain ERC20 tokens, if insufficient tokens are present, no revert occurs but a result of "false" is returned. As explained in https://consensys.net/diligence/audits/2021/01/fei-protocol/#unchecked-return-value-for-iweth-transfer-call

And in this function case, if the weth.transferFrom() returns false, it would continue the call to withdraw token from the contract and send it to the caller. Thus a user could withdraw free tokens, and eventually some users will be unable to withdraw their tokens.

Proof of Concept

https://github.com/code-423n4/2022-06-nested/blob/main/contracts/Withdrawer.sol#L26

- Alice calls Withdrawer.withdraw() with 100 as input.
- Assume weth.transferFrom() fails, returns false but does not revert.

- weth.withdraw() will run since there was no revert.
- Alice receives 100 eth for free.
- This may be possible in a direct call by Alice or a call from

YearnCurveVaultOperator contract in https://github.com/code-423n4/2022-06-nested/blob/main/contracts/operators/Yearn/YearnCurveVaultOperator.sol#L81

Recommended Mitigation Steps

Check the result of transferFrom and transfer. Or making use of SafeERC20 library: safeTransfer and safeTransferFrom would be recommended.

Non-Critical Issue Summary

[N-01] Missing checks for `address(0x0)` when assigning values to address state variables
[N-02] Adding a return statement when the function defines a named return variable, is redundant
[N-03] Public functions not called by the contract should be declared external instead
[N-04] NatSpec is incomplete
[N-05] Event is missing indexed fields
[N-06] Typos
[N-07] Lack of Event Emission For Critical Functions
[N-08] Too Recent of a Pragma
[N-09] Missing selector check on operator
[N-10] Unused imports
[N-11] Change your imports
[N-12] Add `namesLength > 0` check in `areOperatorsImported()` method
[N-13] Libraries, interfaces, and external imports can be ordered nicely
[N-14] Consider checking the recipient address for existence before making the call
[N-15] Consider using IERC20 type instead of address.
[N-16] setMaxAllowance should be called in the constructor
[N-17] Naming inconsistency - some arguments have `_` at their prefixes but others do not at NestedFactory.sol
[N-18] Use either `_msgSender()` or `msg.sender`
[N-19] `OwnerProxy` can call `selfdestruct()`

[N-20] A magic number should be documented and explained. Use a constant instead

[N-01] Missing checks for `address(0x0)` when assigning values to address state variables

There are 2 instances of this issue:

```
File: contracts/operators/Yearn/YearnCurveVaultOperator.sol   #1

48:            eth = _eth;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnCurveVaultOperator.sol#L48

```
File: contracts/abstracts/OwnableProxyDelegation.sol   #2

65:            _owner = newOwner;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/abstracts/OwnableProxyDelegation.sol#L65

[N-02] Adding a return statement when the function defines a named return variable, is redundant

There are 4 instances of this issue:

```
File: contracts/libraries/CurveHelpers/CurveHelpers.sol   #1

25:                    return amounts;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L25

```
File: contracts/libraries/CurveHelpers/CurveHelpers.sol   #2

45:                     return amounts;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L45

```
File: contracts/libraries/CurveHelpers/CurveHelpers.sol   #3

65:                     return amounts;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L65

```
File: contracts/libraries/CurveHelpers/CurveHelpers.sol   #4

89:                     return success;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L89

[N-03] Public functions not called by the contract should be declared external instead

Contracts are allowed to override their parents' functions and change the visibility from external to public.

There are 2 instances of this issue:

```
File: contracts/governance/OwnerProxy.sol   #1

16:         function execute(address _target, bytes memory _data)
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/OwnerProxy.sol#L16

```
File: contracts/governance/TimelockControllerEmergency.sol   #2

295       function executeEmergency(
296           address target,
297           uint256 value,
298           bytes calldata data
299:      ) public payable onlyRole(EMERGENCY_ROLE) {
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L295-L299

[N-04] NatSpec is incomplete

There are 10 instances of this issue:

File: contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol

```
/// @audit Missing: '@return'
230       /// @param path An array of the two paired token addre
231       /// @param biswapRouter The uniswapV2 router to be use
232       function _swapAndAddLiquidity(
233           uint256 amount,
234           uint256 swapAmountIn,
235           address[] memory path,
236           IBiswapRouter02 biswapRouter
237:      ) private returns (uint256 mintedLpAmount) {

/// @audit Missing: '@param reserveA'
258       /// @dev Calculate the optimal amount of tokenA to swa
259       ///         the same market value of tokenB after the
260       ///         This allows to add as many tokensA and tok
261       ///         to the liquidity to minimize the remaining
262       /// @param investmentA The total amount of tokenA to i
263       /// @param pair The IBiswapPair to be used
264       function _getOptimalSwapAmount(
265           uint256 investmentA,
266             uint256 reserveA,
267             uint256 reserveB,
268             IBiswapRouter02 router,
269             IBiswapPair pair
270:        ) private view returns (uint256 swapAmount) {
```

```
/// @audit Missing: '@param reserveB'
258         /// @dev Calculate the optimal amount of tokenA to swa
259         ///         the same market value of tokenB after the
260         ///         This allows to add as many tokensA and tok
261         ///         to the liquidity to minimize the remaininc
262         /// @param investmentA The total amount of tokenA to i
263         /// @param pair The IBiswapPair to be used
264         function _getOptimalSwapAmount(
265             uint256 investmentA,
266             uint256 reserveA,
267             uint256 reserveB,
268             IBiswapRouter02 router,
269             IBiswapPair pair
270:        ) private view returns (uint256 swapAmount) {
```

```
/// @audit Missing: '@param router'
258         /// @dev Calculate the optimal amount of tokenA to swa
259         ///         the same market value of tokenB after the
260         ///         This allows to add as many tokensA and tok
261         ///         to the liquidity to minimize the remaininc
262         /// @param investmentA The total amount of tokenA to i
263         /// @param pair The IBiswapPair to be used
264         function _getOptimalSwapAmount(
265             uint256 investmentA,
266             uint256 reserveA,
267             uint256 reserveB,
268             IBiswapRouter02 router,
269             IBiswapPair pair
270:        ) private view returns (uint256 swapAmount) {
```

```
/// @audit Missing: '@return'
258         /// @dev Calculate the optimal amount of tokenA to swa
259         ///         the same market value of tokenB after the
260         ///         This allows to add as many tokensA and tok
261         ///         to the liquidity to minimize the remaininc
262         /// @param investmentA The total amount of tokenA to i
263         /// @param pair The IBiswapPair to be used
264         function _getOptimalSwapAmount(
265             uint256 investmentA,
266             uint256 reserveA,
267             uint256 reserveB,
268             IBiswapRouter02 router,
269             IBiswapPair pair
270:        ) private view returns (uint256 swapAmount) {
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L230-L237

File: contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol

```
/// @audit Missing: '@return'
230         /// @param path An array of the two paired token addre
231         /// @param uniswapRouter The uniswapV2 router to be us
232         function _swapAndAddLiquidity(
233             uint256 amount,
234             uint256 swapAmountIn,
235             address[] memory path,
236             IUniswapV2Router02 uniswapRouter
237:        ) private returns (uint256 mintedLpAmount) {

/// @audit Missing: '@param reserveA'
258         /// @dev Calculate the optimal amount of tokenA to swa
259         ///         the same market value of tokenB after the
260         ///         This allows to add as many tokensA and tok
261         ///         to the liquidity to minimize the remaining
262         /// @param investmentA The total amount of tokenA to i
263         function _getOptimalSwapAmount(
264             uint256 investmentA,
265             uint256 reserveA,
266             uint256 reserveB,
267             IUniswapV2Router02 router
268:        ) private pure returns (uint256 swapAmount) {

/// @audit Missing: '@param reserveB'
258         /// @dev Calculate the optimal amount of tokenA to swa
259         ///         the same market value of tokenB after the
260         ///         This allows to add as many tokensA and tok
261         ///         to the liquidity to minimize the remaining
262         /// @param investmentA The total amount of tokenA to i
263         function _getOptimalSwapAmount(
264             uint256 investmentA,
265             uint256 reserveA,
        uint256 reserveB,
        IUniswapV2Router02 router
    ) private pure returns (uint256 swapAmount) {
```

```
/// @audit Missing: '@param router'
    /// @dev Calculate the optimal amount of tokenA to swa
    ///         the same market value of tokenB after the
    ///         This allows to add as many tokensA and tok
    ///         to the liquidity to minimize the remaining
    /// @param investmentA The total amount of tokenA to i
    function _getOptimalSwapAmount(
        uint256 investmentA,
        uint256 reserveA,
        uint256 reserveB,
        IUniswapV2Router02 router
    ) private pure returns (uint256 swapAmount) {
```

```
/// @audit Missing: '@return'
    /// @dev Calculate the optimal amount of tokenA to swa
    ///         the same market value of tokenB after the
    ///         This allows to add as many tokensA and tok
    ///         to the liquidity to minimize the remaining
    /// @param investmentA The total amount of tokenA to i
    function _getOptimalSwapAmount(
        uint256 investmentA,
        uint256 reserveA,
        uint256 reserveB,
        IUniswapV2Router02 router
    ) private pure returns (uint256 swapAmount) {
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L230-L237

[N-05] Event is missing `indexed` fields

Each `event` should use three `indexed` fields if there are three or more fields.

There are 8 instances of this issue:

```
File: contracts/operators/Beefy/BeefyVaultStorage.sol
```

```
12:       event VaultAdded(address vault, address tokenOrZapp

16:       event VaultRemoved(address vault);
https:

File: contracts/operators/Yearn/YearnVaultStorage.sol

17:       event VaultAdded(address vault, CurvePool pool);

21:       event VaultRemoved(address vault);
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnVaultStorage.sol#L17

```
File: contracts/governance/TimelockControllerEmergency.sol

37        event CallScheduled(
38            bytes32 indexed id,
39            uint256 indexed index,
40            address target,
41            uint256 value,
42            bytes data,
43            bytes32 predecessor,
44            uint256 delay
45:       );

50:       event CallExecuted(bytes32 indexed id, uint256 indexed

60:       event MinDelayChange(uint256 oldDuration, uint256 newI
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L37-L45

```
File: contracts/abstracts/MixinOperatorResolver.sol

14:       event CacheUpdated(bytes32 name, IOperatorResolver.Ope
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/abstracts/MixinOperatorResolver.sol#L14

[N-06] Typos

- datas vs data

```
abstracts/MixinOperatorResolver.sol:81:    /// @dev Build the ca
```

- setted vs set

```
- abstracts/OwnableProxyDelegation.sol:17:    /// @dev True if t
+ abstracts/OwnableProxyDelegation.sol:17:    /// @dev True if t
```

- liquitiy vs liquidity

```
libraries/StakingLPVaultHelpers.sol:21:    /// @param pool The C
libraries/StakingLPVaultHelpers.sol:52:    /// @param pool The C
libraries/StakingLPVaultHelpers.sol:85:    /// @param pool The C
libraries/StakingLPVaultHelpers.sol:115:    /// @param pool The
```

- WITHDRAWED vs WITHDREW or WITHDRAWN

```
operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol:108:
operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol:108:
operators/Beefy/BeefyVaultOperator.sol:95:        require(vaultA
NestedFactory.sol:51:    /// @dev Fees when funds are withdrawed
NestedFactory.sol:639:    /// @return The withdrawed amount from
```

- dont vs don't

```
NestedFactory.sol:477:    /// @dev Call the operator to submit t
transfered vs transferred
```

```
NestedFactory.sol:534:    /// @return Token transfered (in case
```

[N-07] Lack of Event Emission For Critical Functions

BeefyVaultOperator.sol#L36-L67, BeefyVaultOperator.sol#L79-L108, BeefyZapBiswapLPVaultOperator.sol#L46-L77, BeefyZapBiswapLPVaultOperator.sol#L91-L121, BeefyZapUniswapLPVaultOperator.sol#L46-L77, BeefyZapUniswapLPVaultOperator.sol#L91-L121, ParaswapOperator.sol#L22-L48

Description

Several functions update critical parameters that are missing event emission. These should be performed to ensure tracking of changes of such critical parameters.

Recommendation

Add events to functions that change critical parameters.

[N-08] Too Recent of a Pragma

Context: All Contracts

Description

Using too recent of a pragma is risky since they are not battle tested. A rise of a bug that wasn't known on release would cause either a hack or a need to secure funds and redeploy.

Recommendation

Use a Pragma version that has been used for sometime. I would suggest 0.8.4 for the decrease of risk and still has the gas optimizations implemented.

[N-09] Missing selector check on operator

https://github.com/code-423n4/2022-06-nested/blob/main/contracts/governance/scripts/OperatorScripts.sol#L28
https://github.com/code-423n4/2022-06-nested/blob/main/contracts/OperatorResolver.sol#L20

The addOperator function is not checking that selector of added operator is not bytes4(0)
Same fix is required for requireAndGetOperator function at OperatorResolver.sol#L20

Recommendation

Add below check

```
require(operator.selector != bytes4(0), "AO-SCRIPT: INVALID_SELE
```

[N-10] Unused imports

- Ierc20 is already imported in Inestedfactory.sol
- Feespliter.sol is already imported Inestedfactory.sol
- NestedReverse.sol is already imported Inesteadfactory.sol
- NestedFactory:6
- NestedFactory:12
- safeerc20 imports Ierc20 so you can take out Ierc20 when you import safeerc20.sol
- NestedFactory:13
- IOperatorResolver.sol is already imported in MixinOperatorResolver.sol
- take out IOperatorResolver.sol from OperatorResolver.sol
- OperatorResolver:4
- https://github.com/code-423n4/2022-06-nested/blob/b253ed80f67d1bb2a04e1702f5796fd96a7c521e/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L10
- https://github.com/code-423n4/2022-06-nested/blob/b253ed80f67d1bb2a04e1702f5796fd96a7c521e/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L10
- https://github.com/code-423n4/2022-06-nested/blob/b253ed80f67d1bb2a04e1702f5796fd96a7c521e/contracts/operators/Paraswap/ParaswapOperator.sol#L6

- safeerc20.sol is already in exchangehelper.sol
- https://github.com/code-423n4/2022-06-nested/blob/0dc44d779eaca8f40b7526aabdd81a098dcebf25/contracts/libraries/StakingLPVaultHelpers.sol#L10

[N-11] Change your imports

ex: import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

Instead do your imports like this
import {Ierc20,safeer20} from "import
"@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";

https://github.com/code-423n4/2022-06-nested/blob/b253ed80f67d1bb2a04e1702f5796fd96a7c521e/contracts/NestedFactory.sol#L5

https://github.com/code-423n4/2022-06-nested/blob/b253ed80f67d1bb2a04e1702f5796fd96a7c521e/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L10

[N-12] Add `namesLength > 0` check in `areOperatorsImported()` method

`areOperatorsImported()` in OperatorResolver.sol#L32-L49 returns true when input arrays are empty (ie. `[]`, `[]`).

```
    function areOperatorsImported(bytes32[] calldata names, Operator
        external
        view
        override
        returns (bool)
    {
        uint256 namesLength = names.length;
        require(namesLength == destinations.length, "OR: INPUTS_LENG
        for (uint256 i = 0; i < namesLength; i++) {
            if (
                operators[names[i]].implementation != destinations[i
                operators[names[i]].selector != destinations[i].sele
            ) {
                return false;
            }
        }
        return true;
    }
```

Recommendation

```
require(namesLength > 0 "empty names/destinations");
```

[N-13] Libraries, interfaces, and external imports can be ordered nicely

E.g. group all libraries first, then interfaces, then OZ imports

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/StakingLPVaultHelpers.sol#L4-L12

[N-14] Consider checking the recipient address for existence before making the call

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L283
https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L300
https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L325
https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L358

Consider checking the recipient address for existence before making the call.
If the address does not exist, call will return true and the user will not get the tokens to his wallet.

More information:
https://docs.soliditylang.org/en/develop/control-structures.html#:~:text=Warning-,The%20low%2Dlevel%20functions,-call%2C%20delegatecall

[N-15] Consider using IERC20 type instead of address

Consider using `IERC20` type instead of address or `IERC20[]` type instead of `address[]`.

Affected code:

1. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L248
2. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L257
3. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L291
4. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L371
5. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L422
6. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L423
7. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L460
8. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L461

9. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L487
10. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L488
11. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L587
12. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L640
13. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/abstracts/MixinOperatorResolver.sol#L91
14. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/abstracts/MixinOperatorResolver.sol#L92
15. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/abstracts/MixinOperatorResolver.sol#L101
16. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L19
17. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L39
18. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L59
19. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/CurveHelpers/CurveHelpers.sol#L82
20. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libra

ries/StakingLPVaultHelpers.sol#L62

21. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/StakingLPVaultHelpers.sol#L95
22. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/libraries/StakingLPVaultHelpers.sol#L125
23. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/BeefyVaultStorage.sol#L24
24. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L133
25. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L140
26. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L141
27. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L153
28. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L181
29. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L191
30. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L192
31. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L133

32. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L140
33. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L141
34. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L153
35. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L181
36. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L191
37. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L192
38. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Paraswap/ParaswapOperator.sol#L11
39. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnCurveVaultOperator.sol#L117
40. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnCurveVaultOperator.sol#L226
41. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnCurveVaultOperator.sol#L274
42. https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnVaultStorage.sol#L9

[N-16] setMaxAllowance should be called in the constructor

BeefyVaultOperator.sol

deposit: setMaxAllowance should be called in the constructor

In the `deposit` function on line 48, we call `ExchangeHelpers.setMaxAllowance(token, vault);` to allow the vault to spend token.

Each time assets are deposited in the vault, we shouldn't have to allow it to spend the token again.

I recommend to call `ExchangeHelpers.setMaxAllowance(token, vault)` only once in the constructor for each vault and token. I also recommend to add a `setMaxAllowance` function only callable by the owner of the operator that would allow to set the max allowance in case the allowance has decreased.

Recommendation

```
    for (uint256 i; i < vaultsLength; i++) {
        operatorStorage.addVault(vaults[i], tokens[i]);
        ExchangeHelpers.setMaxAllowance(tokens[i], vaults[i]);
    }
```

BeefyZapBiswapLPVaultOperator.sol and BeefyZapUniswapLPVaultOperator.sol

zapAndStakeLp: setMaxAllowance should be called in the constructor

In the `_zapAndStakeLp` function on line 189 and subsequently on line 194 and 195 we call `ExchangeHelpers.setMaxAllowance();` to allow the vault to spend token.

Each time assets are deposited in the vault, we shouldn't have to allow it to spend the token again.
I recommend to call `ExchangeHelpers.setMaxAllowance()` only once in the constructor for each vault and token. I also recommend to add a setMaxAllowance function only callable by the owner of the operator that would allow to set the max allowance in case the allowance has decreased.

This will also avoid calling `ExchangeHelpers.setMaxAllowance(IERC20(swapToken), router);` in the `_withdrawAndSwap` function on line 162.

Recommendation

```
    for (uint256 i; i < vaultsLength; i++) {
        operatorStorage.addVault(vaults[i], routers[i]);

        IBiswapPair pair = IBiswapPair(IBeefyVaultV6(vaults[i]).want

        ExchangeHelpers.setMaxAllowance(IERC20(address(pair)), addre
        ExchangeHelpers.setMaxAllowance(IERC20(pair.token0()), route
        ExchangeHelpers.setMaxAllowance(IERC20(pair.token1()), route
    }
```

YearnCurveVaultOperator.sol

depositETH: setMaxAllowance should be called in the constructor

In the `depositETH` function on line 78, we call `ExchangeHelpers.setMaxAllowance(IERC20(address(weth)), address(withdrawer));` to allow the withdrawer to spend weth.

Each time assets are deposited in the vault, we shouldn't have to allow the withdrawer to spend weth again.
I recommend to call `ExchangeHelpers.setMaxAllowance(IERC20(address(weth)), address(withdrawer))` only once in the constructor. I also recommend to add a `setMaxAllowance` function only callable by the owner of the operator that would allow to set the max allowance in case the allowance has decreased.

Recommendation

```
    ExchangeHelpers.setMaxAllowance(IERC20(_weth), address(_withdraw
```

StakingLPVaultHelpers.sol

addLiquidityAndDepositETH: setMaxAllowance should be called in addVault

In the `addLiquidityAndDepositETH` function on [line 45](), we call `ExchangeHelpers.setMaxAllowance(lpToken, vault);` to allow the vault to spend lpToken.

Each time assets are deposited in the vault, we shouldn't have to allow the vault to spend lpToken again.
I recommend to call `ExchangeHelpers.setMaxAllowance(lpToken, vault);` only once in the addVault function of the YearnVaultStorage. I also recommend to add a setMaxAllowance function only callable by the owner that would allow to set the max allowance in case the allowance has decreased.

This will also avoid calling it in the `_addLiquidityAndDeposit` function on line [77]()

Recommendation

```
    ExchangeHelpers.setMaxAllowance(curvePool.lpToken, vault);
```

[N-17] Naming inconsistency - some arguments have `_` at their prefixes but others do not at NestedFactory.sol

Throughout the file `NestedFactory.sol` , arguments of functions have _ at their prefixes like function `setFeeSplitter(FeeSplitter _feeSplitter)` . However, following 2 arguments do not have `_` at their prefixes which are not consistent.

https://github.com/code-423n4/2022-06-nested/blob/main/contracts/NestedFactory.sol#L121

https://github.com/code-423n4/2022-06-nested/blob/main/contracts/NestedFactory.sol#L133

[N-18] Use either `_msgSender()` or `msg.sender`

Throughout the file `NestedFactory.sol` , `_msgSender()` is used to get the sender. However, following 2 places use msg.sender which seem not consistent.

https://github.com/code-423n4/2022-06-nested/blob/main/contracts/NestedFactory.sol#L89

[N-19] `OwnerProxy` can call `selfdestruct()`

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/OwnerProxy.sol#L9-L36

Impact

OwnerProxy's selfdestruct

Proof of Concept

While only the owner (the timelock) can call the execute function, this doesn't mean it can't be compromised or phished to call a malicious `_target`, which could contain a call to `selfdestruct()`.

As `selfdestruct()` would be a simple OPCODE in the context of the OwnerProxy contract (which is the one using `delegatecall()` in `execute()`), this would destroy the contract.

This is a known bug in the community (see the Parity Multisig Hack): delegatecalls from contracts are dangerous.

Recommended Mitigation Steps

Consider making OwnerProxy a library instead of a contract to protect it from being selfdestructed and to further protect its state (that can also be manipulated as a contract)

Alternatively, consider deploying the OwnerProxy contract using CREATE2 so that the contract could be re-created at the same pre-computed address, if need be.

[N-20] A magic number should be documented and explained. Use a constant instead.

Similar issue in the past: here

1:

```
operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol:240:
operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol:251:
operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol:252:
operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol:240:
operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol:251:
operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol:252:
```

10000:

```
NestedFactory.sol:378:        feesAmount = (amountSpent * entryF
NestedFactory.sol:443:            feesAmount = (amountBought * (
```

I suggest using `constant` variables as this would make the code more maintainable and readable while costing nothing gas-wise (constants are replaced by their value at compile-time).

[N-21] Lack of solhint To Ignore Warnings

Context: [OwnerProxy.sol#L21-L36](), [OperatorScripts.sol#L58-L60]()

Description:

solhint is useful to help ignore warnings that aren't really issues. For example the code base has some assembly blocks which promts the warning `Linter: Avoid to use inline assembly. It is acceptable only in rare cases`. Adding in `/* solhint-disable no-inline-assembly */` above it will ignore this warning.

Recommendation

Use solhint to ignore warnings that aren't really issues.

maximebrugel (Nested) responded:

> **OK:**

- [L-01] Known vulnerabilities exist in currently used @openzeppelin/contracts version
- [L-02] Impractical Entry/Exit fees are allowed
- [L-03] poolCoinAmount validation**
- [L-04] Low level calls with solidity version 0.8.14 can result in optimiser bug.
- [L-05] Unchecked return value of transferFrom can allow a user to withdraw native token for free.
- [N-01] Missing checks for `address(0x0)` when assigning values to address state variables
- [N-02] Adding a return statement when the function defines a named return variable, is redundant
- [N-03] Public functions not called by the contract should be declared external instead
- [N-04] NatSpec is incomplete
- [N-06] Typos
- [N-07] Lack of Event Emission For Critical Functions
- [N-09] Missing selector check on operator
- [N-10] Unused imports
- [N-12] Add `namesLength > 0` check in `areOperatorsImported()` method
- [N-13] Libraries, interfaces, and external imports can be ordered nicely
- [N-15] Consider using IERC20 type instead of address.
- [N-17] Naming inconsistency - some arguments have `_` at their prefixes but others do not at NestedFactory.sol
- [N-18] Use either `_msgSender()` or `msg.sender`
- [N-19] `OwnerProxy` can call `selfdestruct()`
- [N-20] A magic number should be documented and explained. Use a constant instead

> **Acknowledged:**

- [N-05] Event is missing indexed fields
- [N-08] Too Recent of a Pragma

- [N-11] Change your imports
- [N-14] Consider checking the recipient address for existence before making the call
- [N-16] setMaxAllowance should be called in the constructor

> **Not an Issue for Us:**

- [N-21] Lack of solhint To Ignore Warnings

Gas Optimizations

For this contest, 27 reports were submitted by wardens detailing gas optimizations. The report submitted by IllIllI received the top score from the judge.

The gas optimizations below include reports submitted by IllIllI, as well as: Meera, 0xkatana, ElKu, joestakey, minhquanym, PierrickGT, Waze, MiloTruck, c3phas, _Adam, 0xKitsune, Chom, UnusualTurtle, 0xNazgul, delfin454000, fatherOfBlocks, robee, sach1r0, simon135, 0x1f8b, asutorufos, JC, oyc_109, Picodes, SooYa, *and* TerrierLover.

Summary

[G-1] Using `calldata` instead of `memory` for read-only arguments in external functions saves gas
[G-2] Using `storage` instead of `memory` for structs/arrays saves gas
[G-3] Multiple accesses of a mapping/array should use a local variable cache
[G-4] It costs more gas to initialize non-constant/non-immutable variables to zero than to let the default of zero be applied
[G-5] Splitting `require()` statements that use `&&` saves gas
[G-6] Using `private` rather than `public` for `constants`, saves gas
[G-7] Division by two should use bit shifting
[G-8] `require()` or `revert()` statements that check input arguments should be at the top of the function
[G-9] Functions guaranteed to revert when called by normal users can be marked `payable`
[G-10] The `require` statements could be put at the beginning part of a block of statements if it doesn’t affect the logic to save gas.

[G-01] Using `calldata` instead of `memory` for read-only arguments in external functions saves gas

When a function with a `memory` array is called externally, the `abi.decode()` step has to use a for-loop to copy each index of the `calldata` to the `memory` index. Each iteration of this for-loop costs at least 60 gas (i.e. `60 * <mem_array>.length`). Using calldata directly, obliviates the need for such a loop in the contract code and runtime execution.

If the array is passed to an `internal` function which passes the array to another `internal` function where the array is modified and therefore `memory` is used in the `external` call, it's still more gas-efficient to use `calldata` when the external function uses modifiers, since the modifiers may prevent the `internal` functions from being called. `Structs` have the same overhead as an array of length one

There are 3 instances of this issue:

```
File: contracts/governance/scripts/OperatorScripts.sol   #1

28:       function addOperator(IOperatorResolver.Operator mem
https:
```

```
File: contracts/governance/scripts/OperatorScripts.sol   #2

52:       function deployAddOperators(bytes memory bytecode,
https:
```

```
File: contracts/governance/scripts/OperatorScripts.sol   #3
```

```
52:          function deployAddOperators(bytes memory bytecode,
```

52 https:

[G-02] Using `storage` instead of `memory` for structs/arrays saves gas

When fetching data from a `storage` location, assigning the data to a `memory` variable causes all fields of the struct/array to be read from storage, which incurs a Gcoldsload (2100 gas) for each field of the struct/array. If the fields are read from the new `memory` variable, they incur an additional `MLOAD` rather than a cheap stack read. Instead of declearing the variable with the `memory` keyword, declaring the variable with the storage keyword and caching any fields that need to be re-read in stack variables, will be much cheaper, only incuring the Gcoldsload for the fields actually read. The only time it makes sense to read the whole struct/array into a `memory` variable, is if the full struct/array is being returned by the function, is being passed to a function that requires `memory`, or if the array/struct is being read from another `memory` array/struct

There is 1 instance of this issue:

```
File: contracts/NestedFactory.sol   #1

123:          bytes32[] memory operatorsCache = operators;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L123

[G-03] Multiple accesses of a mapping/array should use a local variable cache

The instances below point to the second+ access of a value inside a mapping/array, within a function. Caching a mapping's value in a local **storage** variable when the value is accessed [multiple times](https://gist.github.com/IllIllI000/ec23a57daa30a8f8ca8b9681c8ccefb0), **saves ~42 gas** per access due to not having to recalculate the key's keccak256 hash (**Gkeccak256 - 30 gas**) and that calculation's associated stack operations. Caching an array's struct avoids recalculating the array offsets into memory.

There are 4 instances of this issue:

```
File: contracts/operators/Yearn/YearnVaultStorage.sol   #1

/// @audit vaults[vault] on line 33
34:             require(vaults[vault].lpToken == address(0), "YVS:
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnVaultStorage.sol#L34

```
File: contracts/governance/scripts/OperatorScripts.sol   #2

/// @audit operators[i] on line 68
69:                 operatorsToImport[i] = IOperatorResolver.Opera
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/scripts/OperatorScripts.sol#L69

```
File: contracts/OperatorResolver.sol   #3

/// @audit operators[<etc>] on line 42
43:                     operators[names[i]].selector != destinatic
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/OperatorResolver.sol#L43

```
File: contracts/OperatorResolver.sol   #4

/// @audit destinations[i] on line 42
43:                     operators[names[i]].selector != destinatic
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/OperatorResolver.sol#L43

[G-04] It costs more gas to initialize non-constant/non-immutable variables to zero than to let the default of zero be applied

Not overwriting the default for stack variables saves 8 gas. Storage and memory variables have larger savings

There are 18 instances of this issue:

```
File: contracts/governance/TimelockControllerEmergency.sol

84:            for (uint256 i = 0; i < proposers.length; ++i) {

89:            for (uint256 i = 0; i < executors.length; ++i) {

234:           for (uint256 i = 0; i < targets.length; ++i) {

324:           for (uint256 i = 0; i < targets.length; ++i) {
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L84

```
File: contracts/abstracts/MixinOperatorResolver.sol

37:            for (uint256 i = 0; i < requiredOperators.length;

56:            for (uint256 i = 0; i < requiredOperators.length;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/abstracts/MixinOperatorResolver.sol#L37

```
File: contracts/OperatorResolver.sol

40:            for (uint256 i = 0; i < namesLength; i++) {

60:            for (uint256 i = 0; i < names.length; i++) {

75:            for (uint256 i = 0; i < destinations.length; i++)
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/OperatorResolver.sol#L40

```
File: contracts/NestedFactory.sol

124:            for (uint256 i = 0; i < operatorsCache.length; i+

136:            for (uint256 i = 0; i < operatorsLength; i++) {

196:            for (uint256 i = 0; i < batchedOrdersLength; i++)

256:            for (uint256 i = 0; i < tokensLength; i++) {

315:            for (uint256 i = 0; i < batchedOrdersLength; i++)

333:            for (uint256 i = 0; i < batchedOrdersLength; i++)

369:            for (uint256 i = 0; i < batchLength; i++) {

412:            for (uint256 i = 0; i < batchLength; i++) {

651:            for (uint256 i = 0; i < _batchedOrders.length; i+
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L124

[G-05] Splitting `require()` statements that use `&&` saves gas

See this issue which describes the fact that there is a larger deployment gas cost, but with enough runtime calls, the change ends up being cheaper.

There are 7 instances of this issue:

```
File: contracts/operators/Beefy/BeefyVaultOperator.sol

54:            require(vaultAmount != 0 && vaultAmount >= minVaul
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/BeefyVaultOperator.sol#L54

```
File: contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator

64:            require(vaultAmount != 0 && vaultAmount >= minVaul

65:            require(depositedAmount != 0 && amountToDeposit >=
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L64

```
File: contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperato

64:            require(vaultAmount != 0 && vaultAmount >= minVaul

65:            require(depositedAmount != 0 && amountToDeposit >=
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L64

```
File: contracts/operators/Paraswap/ParaswapOperator.sol

16:            require(_tokenTransferProxy != address(0) && _augu
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Paraswap/ParaswapOperator.sol#L16

```
File: contracts/NestedFactory.sol

66              require(
67                  address(_nestedAsset) != address(0) &&
68                      address(_nestedRecords) != address(0) &&
69                      address(_reserve) != address(0) &&
70                      address(_feeSplitter) != address(0) &&
71                      address(_weth) != address(0) &&
72                      _operatorResolver != address(0) &&
73                      address(_withdrawer) != address(0),
74                  "NF: INVALID_ADDRESS"
75:             );
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L66-L75

[G-06] Using `private` rather than `public` for `constants`, saves gas

If needed, the value can be read from the verified contract source code. Saves 3406-3606 gas in deployment gas due to the compiler not having to create non-payable getter functions for deployment calldata, not having to store the bytes of the value outside of where it's used, and not adding another entry to the method ID table

There are 4 instances of this issue:

```
File: contracts/governance/TimelockControllerEmergency.sol   #1

25:        bytes32 public constant TIMELOCK_ADMIN_ROLE = keccak25
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governan [ce/TimelockControllerEmergency.sol#L25](https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L25)

```
File: contracts/governance/TimelockControllerEmergency.sol   #2

26:       bytes32 public constant PROPOSER_ROLE = keccak256("PRO
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L26

```
File: contracts/governance/TimelockControllerEmergency.sol   #3

27:       bytes32 public constant EXECUTOR_ROLE = keccak256("EXE
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L27

```
File: contracts/governance/TimelockControllerEmergency.sol   #4

28:       bytes32 public constant EMERGENCY_ROLE = keccak256("EM
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L28

[G-07] Division by two should use bit shifting

`<x> / 2` is the same as `<x> >> 1`. The `DIV` opcode costs 5 gas, whereas `SHR` only costs 3 gas.

There are 2 instances of this issue:

```
File: contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator
```

```
275:            uint256 halfInvestment = investmentA / 2;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapBiswapLPVaultOperator.sol#L275

```
File: contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperatc

273:            uint256 halfInvestment = investmentA / 2;
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/lp/BeefyZapUniswapLPVaultOperator.sol#L273

[G-08] `require()` or `revert()` statements that check input arguments should be at the top of the function

Checks that involve constants should come before checks that involve state variables.

There is 1 instance of this issue:

```
File: contracts/governance/TimelockControllerEmergency.sol   #1

244:            require(delay >= getMinDelay(), "TimelockControlle
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/TimelockControllerEmergency.sol#L244

[G-09] Functions guaranteed to revert when called by normal users can be marked `payable`

If a function modifier such as onlyOwner is used, the function will revert if a normal user tries to pay the function. Marking the function as payable will lower the gas cost for legitimate callers because the compiler will not include checks for whether a payment was provided. The extra opcodes avoided are `CALLVALUE(2),DUP1(3),ISZERO(3),PUSH2(3),JUMPI(10),PUSH1(3),DUP1(3),REVERT(0),JUMPDEST(1),POP(2)`, which costs an average of about 21 gas per call to the function, in addition to the extra deployment cost.

There are 27 instances of this issue:

```
File: contracts/operators/Beefy/BeefyVaultStorage.sol

24:        function addVault(address vault, address tokenOrZapper

34:        function removeVault(address vault) external onlyOwner
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Beefy/BeefyVaultStorage.sol#L24

```
File: contracts/operators/Yearn/YearnVaultStorage.sol

29:        function addVault(address vault, CurvePool calldata cu

41:        function removeVault(address vault) external onlyOwner
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/operators/Yearn/YearnVaultStorage.sol#L29

```
File: contracts/governance/OwnerProxy.sol

16:        function execute(address _target, bytes memory _data)
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governance/OwnerProxy.sol#L16

```
File: contracts/governance/TimelockControllerEmergency.sol
```

```
199        function schedule(
200            address target,
201            uint256 value,
202            bytes calldata data,
203            bytes32 predecessor,
204            bytes32 salt,
205            uint256 delay
206:       ) public virtual onlyRole(PROPOSER_ROLE) {

221        function scheduleBatch(
222            address[] calldata targets,
223            uint256[] calldata values,
224            bytes[] calldata datas,
225            bytes32 predecessor,
226            bytes32 salt,
227            uint256 delay
228:       ) public virtual onlyRole(PROPOSER_ROLE) {

255:       function cancel(bytes32 id) public virtual onlyRole(P

274        function execute(
275            address target,
276            uint256 value,
277            bytes calldata data,
278            bytes32 predecessor,
279            bytes32 salt
280:       ) public payable virtual onlyRoleOrOpenRole(EXECUTOR_

295        function executeEmergency(
296            address target,
297            uint256 value,
298            bytes calldata data
299:       ) public payable onlyRole(EMERGENCY_ROLE) {

312        function executeBatch(
313            address[] calldata targets,
314            uint256[] calldata values,
315            bytes[] calldata datas,
316            bytes32 predecessor,
317            bytes32 salt
318:       ) public payable virtual onlyRoleOrOpenRole(EXECUTOR_
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/governan ce/TimelockControllerEmergency.sol#L199-L206

```
File: contracts/abstracts/OwnableProxyDelegation.sol

50:        function renounceOwnership() public virtual onlyOwner

56:        function transferOwnership(address newOwner) public vi
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/abstracts/OwnableProxyDelegation.sol#L50

```
File: contracts/OperatorResolver.sol

52         function importOperators(
53             bytes32[] calldata names,
54             Operator[] calldata operatorsToImport,
55             MixinOperatorResolver[] calldata destinations
56:        ) external override onlyOwner {

74:        function rebuildCaches(MixinOperatorResolver[] calldat
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/OperatorResolver.sol#L52-L56

```
File: contracts/NestedFactory.sol

121:       function addOperator(bytes32 operator) external overri

133:       function removeOperator(bytes32 operator) external ove

152:       function setFeeSplitter(FeeSplitter _feeSplitter) exte

159:       function setEntryFees(uint256 _entryFees) external ove

167:       function setExitFees(uint256 _exitFees) external overr

175:       function unlockTokens(IERC20 _token) external override
```

```
205        function processInputOrders(uint256 _nftId, BatchedInp
206            external
207            payable
208            override
209            nonReentrant
210            onlyTokenOwner(_nftId)
211:           isUnlocked(_nftId)

219        function processOutputOrders(uint256 _nftId, BatchedOu
220            external
221            override
222            nonReentrant
223            onlyTokenOwner(_nftId)
224:           isUnlocked(_nftId)

231        function processInputAndOutputOrders(
232            uint256 _nftId,
233            BatchedInputOrders[] calldata _batchedInputOrders,
234            BatchedOutputOrders[] calldata _batchedOutputOrder
235:       ) external payable override nonReentrant onlyTokenOwne

243        function destroy(
244            uint256 _nftId,
245            IERC20 _buyToken,
246            Order[] calldata _orders
247:       ) external override nonReentrant onlyTokenOwner(_nftIc

278        function withdraw(uint256 _nftId, uint256 _tokenIndex)
279            external
280            override
281            nonReentrant
282            onlyTokenOwner(_nftId)
283:           isUnlocked(_nftId)

301:       function updateLockTimestamp(uint256 _nftId, uint256 _
```

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L121

[G-10] The `require` statements could be put at the

beginning part of a block of statements if it doesn't affect the logic to save gas.**

```
require(_orders.length != 0, "NF: INVALID_ORDERS");
```

These arithmetic operations can be unchecked.

```
uint256 halfInvestment = investmentA / 2;

uint256 halfInvestment = investmentA / 2;
```

This line could be pre-computed and defined as a constant to save gas.

```
bytes4(keccak256(bytes("remove_liquidity_one_coin(uint256,int12
b. bytes4(keccak256(bytes("remove_liquidity_one_coin(uint256,uir
```

[G-11] Unnecessary computation

When emitting an event that includes a new and an old value, it is cheaper in gas to avoid caching the old value in memory. Instead, emit the event, then save the new value in storage.

Proof of Concept

Instances include:

```
OwnableProxyDelegation.sol
function _setOwner
```

Recommended Mitigation

Replace

```
address oldOwner = _owner;
_owner = newOwner;
emit OwnershipTransferred(oldOwner, newOwner)
```

with

```
    emit OwnershipTransferred(_owner_, newOwner)
    _owner = newOwner;
```

[G-12] Mathematical optimizations

`X += Y` costs 22 more gas than `X = X + Y`. This can mean a lot of gas wasted in a function call when the computation is repeated `n` times (loops)

Proof of Concept

Instances include:

- NestedFactory.sol
- amountBought -= amountFees
- amountSpent += *submitOrder(address(tokenSold)*,batchedOrders.orders[i].token,*nftId*,batchedOrders.orders[i],true // always to the reserve)
- ethNeeded += _batchedOrders[i].amount

Recommended Mitigation

use `X = X + Y` instead of `X += Y` (same with `-`).

[G-13] Constructor parameters should be avoided when possible

Constructor parameters are expensive. The contract deployment will be cheaper in gas if they are hard coded instead of using constructor parameters. With the compilers parameters in `hardhat.config.ts`, deployment costs approximately `400` more gas per variable written via a constructor parameter.

Proof of Concept

Instances include:

YearnCurveVaultOperator.sol

[eth = _eth]

[withdrawer = _withdrawer]

Recommended Mitigation

Hardcode storage variables with their initial value instead of writing it during contract deployment with constructor parameters.

[G-14] Can save gas when call `_submitOutOrders()` with `_toReserve = false`

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L443

In case `_toReserve = false`, it still calculates `feesAmount` using `entryFees` but do not use the result anywhere. We can save gas by calculating `feesAmount` only when `_toReserve = true`

Recommended Mitigation Steps

Only calculate `feesAmount` only when `_toReserve = true`

```
    if (_toReserve) {
            feesAmount = (amountBought * entryFees) / 10000;
    _transferToReserveAndStore(_batchedOrders.outputToken, amountBou
    }
```

[G-15] transferToReserveAndStore: balanceReserveAfter can be inlined

In the `_transferToReserveAndStore` function, we store the reserve balance after the transfer in the balanceReserveAfter variable, on line 523.

This variable being only used once, we can inline it and save one mstore.

Recommendation

```
    nestedRecords.store(_nftId, address(_token), _token.balanceOf(re
```

[G-16] Inequality

https://github.com/code-423n4/2022-06-nested/blob/b4a153c943d54755711a2f7b80cbbf3a5bb49d76/contracts/NestedFactory.sol#L286

Non strict inequality are cheaper than strict one. I suggest to use >= or <= instead of > and < if possible.

maximebrugel (Nested) responded:

> **OK:**

- [G-1] Using `calldata` instead of `memory` for read-only arguments in external functions saves gas
- [G-2] Using `storage` instead of `memory` for structs/arrays saves gas
- [G-3] Multiple accesses of a mapping/array should use a local variable cache
- [G-5] Splitting `require()` statements that use `&&` saves gas
- [G-7] Division by two should use bit shifting
- [G-10] The `require` statements could be put at the beginning part of a block of statements if it doesn't affect the logic to save gas.
- [G-12] Mathematical optimizations
- [G-13] Constructor parameters should be avoided when possible
- [G-14] Can save gas when call `_submitOutOrders()` with `_toReserve = false`
- [G-15] transferToReserveAndStore: balanceReserveAfter can be inlined
- [G-16] Inequality

> **Acknowledged:**

- [G-6] Using `private` rather than `public` for `constants`, saves gas
- [G-11] Unnecessary computation

Disclosures

Top