Three Sigma

Code Audit

OSTIUM

Ostium Onchain perpetuals for Real World Assets

Disclaimer

Code Audit

Ostium Onchain perpetuals for Real World Assets

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Code Audit

Ostium Onchain perpetuals for Real World Assets

Table of Contents

Summary

Code Audit

Ostium Onchain perpetuals for Real World Assets

Summary

Three Sigma audited Ostium in a 10 person week engagement. The audit was conducted from 19-02-2024 to 22-03-2024.

Protocol Description

Ostium is the first decentralized perpetuals exchange purpose-engineered for Real World Assets. A stablecoin-settled trading engine, dynamic fee structure, and proprietary in-house oracle enables leveraged, synthetic trading on a wide range of traditional market assets – from oil to natural gas, soybeans, and more – all fully onchain.

Scope

Code Audit

Ostium Onchain perpetuals for Real World Assets

Scope

Filepath	nSLOC	ERCs	Dependencies	Contents
src/abstract/Delegatable.sol	47	-	-	-
src/lib/ChainUtils.sol	13	-	-	-
src/lib/TradeUtils.sol	128	-	-	-
src/OstiumLinkUpKeep.sol	189	-	LINK	-
src/OstiumLockedDepositNft.sol	36	721	-	-
src/OstiumOpenPnl.sol	152	-	-	-
src/OstiumPairInfos.sol	535	-	-	-
src/OstiumPairsStorage.sol	223	-	-	-
src/OstiumPriceRouter.sol	34	-	-	-
src/OstiumPriceUpKeep.sol	169	-	-	Funds
src/OstiumRegistry.sol	84	-	-	-
src/OstiumTimelockManager.sol	7	-	-	-
src/OstiumTimelockOwner.sol	7	-	-	-
src/OstiumTradesUpKeep.sol	315	-	-	-
src/OstiumTrading.sol	486	-	-	-
src/OstiumTradingCallbacks.sol	535	-	-	-
src/OstiumTradingStorage.sol	361	-	-	-
src/OstiumVault.sol	532	4626	-	Funds
src/OstiumWhitelist.sol	43	-	-	-
SUM	3896			

Assumptions

External dependencies are considered secure.

Methodology

Code Audit

Ostium Onchain perpetuals for Real World Assets

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit we report our findings using as a guideline Immunefi's vulnerability taxonomy, which can be found at immunefi.com/severity-updated/. The final classification takes into account the severity, according to the previous link, and likelihood of the exploit. The following table summarizes the general expected classification according to severity and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Severity / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Code Audit

Ostium Onchain perpetuals for Real World Assets

Project Dashboard

Application Summary

Name	Ostium
Commit	1b70cffbac621d39e321c159e45c048b703add92
Language	Solidity
Platform	Arbitrum

Engagement Summary

Timeline	19-02-2024 to 22-03-2024
№ of Auditors	2
Review Time	10 person weeks

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	1	1	0
High	2	2	0
Medium	12	8	4
Low	17	8	9
None	25	16	9

Category Breakdown

Suggestion	1
Documentation	0
Bug	36
Optimization	13
Good Code Practices	8

Code Maturity Evaluation

Code Audit

Ostium Onchain perpetuals for Real World Assets

Code Maturity Evaluation

Code Maturity Evaluation Guidelines

Category	Evaluation
Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system.
Arithmetic	The proper use of mathematical operations and semantics.
Centralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Code Stability	The extent to which the code was altered during the audit.
Upgradeability	The presence of parameterizations of the system that allow modifications after deployment.
Function Composition	The functions are generally small and have clear purposes.
Front-Running	The system's resistance to front-running attacks.
Monitoring	All operations that change the state of the system emit events, making it simple to monitor the state of the system. These events need to be correctly emitted.
Specification	The presence of comprehensive and readable codebase documentation.
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage.

Code Maturity Evaluation Results

Category	Evaluation
Access Controls	Satisfactory. The codebase has a strong access control mechanism.
Arithmetic	Satisfactory. The codebase uses Solidity version >0.8.0 implementing safe arithmetic.
Centralization	Moderate. The admins/owners of the protocol have some privileges over the protocol (e.g setting parameters or oracle prices).
Code Stability	Satisfactory. The codebase was stable throughout the audit.
Upgradeability	Satisfactory. Most contracts are set up to allow for upgradability.
Function Composition	Satisfactory. Functionalities are well split into different contracts and helpers.
Front-Running	Satisfactory. No significant front-running issues were found.
Monitoring	Satisfactory. Most state changing occurrences emit events.
Specification	Satisfactory. The code matched the design specifications.
Testing and Verification	Satisfactory. Unit and fuzz tests were present for most functionality.

Findings

Code Audit

Ostium Onchain perpetuals for Real World Assets

Findings

3S-OS-C01

In **`OstiumTradingStorage`**, **`firstEmptyTradeIndex()`** and **`firstEmptyOpenLimitIndex()`** overwrite index 0 if not found

Id	3S-OS-C01
Classification	Critical
Severity	Critical
Likelihood	Medium
Category	Bug
Status	Addressed in #21e90e7.

Description

OstiumTradingStorage::firstEmptyTradeIndex() and OstiumTradingStorage::firstEmptyOpenLimitIndex() return index 0 if they can not find an available index. This may lead to overwriting a trade in **`storeTrade()`** if, for example, a trader has 2 trades, opens a third one via market order, but the **`Gov`** frontruns the chainlink bot **`OstiumPriceUpKeep()`** and sets **`_maxTradesPerPair`** to 2. It will overwrite the trade in the callback, more specifically in OstiumTradingCallbacks::registerTrade() and lastly in OstiumTradingStorage::storeTrade().

Recommendation

Revert if it can not find an available open trade / limit order slot instead of returning index **`0`**.

3S-OS-H01

Liquidations can be prevented by updating the **`SL`** timeout before it expires

Id	3S-OS-H02
Classification	High
Severity	High
Likelihood	High
Category	Bug
Status	Addressed in #32a1701.

Description

Traders who continually update their stop losses before the **`SL`** timeout expires will never face liquidation. This is because liquidating via a **`LimitOrder`** of type **`LIQ`** is impossible when the limit order includes a stop loss. Additionally, triggering the stop loss with a **`LimitOrder`** of type **`SL`** is hindered by the timeout check.

A PoC can be found here.

Recommendation

If the trade is liquidatable, disregard the timeout for stop losses.

3S-OS-H02

Uneven **`getPendingAccFundingFees()`** leading to wrong funding rate update

Id	3S-OS-H03
Classification	High
Severity	High
Likelihood	High
Category	Bug
Status	Addressed in .

Description

Ostium uses a velocity funding rate update, so the actual change is the integral of the velocity of the past time. There is a case when this update (which can be viewed as the area of the function, as shown in the docs) is done incorrectly. Namely, when **`absLastFundingRate`** is bigger than **`absNewFundingRate`**, the square part of the area (in the negative y axis), has side **`absNewFundingRate`**, but the docs and code use **`absLastFundingRate`**.

Here is a poc. It can be seen that going from -5000 to 1000 gives a different change than -1000 to 5000.

Recommendation

Update the docs and tweak the code to accomodate for this change, such as:

```
function getPendingAccFundingFees(uint16 pairIndex)
    public
    view
    returns (int256 valueLong, int256 valueShort, int64 fr)
{
    ...
    if (absNewFundingRate > f.maxFundingFeePerBlock) {
        ...
    } else {
        ...
```

```
        uint256 squareHeight = absNewFundingRate < absLastFundingRate ?
absNewFundingRate : absLastFundingRate;
        accumulated_funding_rate_change = int256(
            (squareHeight + (uint256(numBlocksToCharge).ceilDiv(2) *
absLastVelocity))
                * uint256(numBlocksToCharge)
        );
        ...
    }
}
```

3S-OS-M01

Inability to Close Positions with Significant Positive PnL

Id	3S-OS-M01
Classification	Medium
Severity	High
Likelihood	Low
Category	Suggestion
Status	Acknowledged

Description

Presently, when there isn't adequate liquidity in the **`OstiumVault`** to cover profits for trades with substantial positive PnL, a revert occurs. While this behavior is intended, it's worth considering implementing a solution to partially receive the profit up to a maximum threshold or at least the collateral itself. Otherwise, traders won't be able to exit a position at all.

3S-OS-M02

Missing Pausability Check in `OstiumTrading::executeAutomationOrder()` for Opening Orders

Id	3S-OS-M02
Classification	Medium
Severity	Medium
Likelihood	High
Category	Bug
Status	Addressed in #fa95bb0.

Description

The execution of a limit order within the `OstiumTrading::executeAutomationOrder()` function lacks a check to determine if the contract is paused.

Recommendation

```
if (orderType == IOstiumTradingStorage.LimitOrder.OPEN) {
    if (!storageT.hasOpenLimitOrder(trader, pairIndex, index)) {
        return IOstiumTrading.AutomationOrderStatus.NO_LIMIT;
    }
+   isNotPaused();
    IOstiumTradingStorage.OpenLimitOrder memory l =
storageT.getOpenLimitOrder(trader, pairIndex, index);
    leveragedPos = l.collateral * l.leverage / 100;
}
```

3S-OS-M03

`OstiumTrading::topUpCollateral()` is missing `pairsStored.groupMaxCollateral(pairIndex)` check

Id	3S-OS-M03
Classification	Medium
Severity	Medium
Likelihood	High
Category	Bug
Status	Addressed in .

Description

When opening trades it is checked if the collateral is within limits for each pair in OstiumTradingCallbacks::withinExposureLimits(). The same should be done in OstiumTrading::topUpCollateral().

Recommendation

Add the check to `OstiumTrading::topUpCollateral()`.

3S-OS-M04

`OstiumTrading::closeTradeMarket()` and `OstiumTrading::topUpCollateral()` are missing pending trigger checks

Id	3S-OS-M04
Classification	Medium
Severity	Medium
Likelihood	High
Category	Bug
Status	Addressed in #7a80be7.

Description

OstiumTrading::closeTradeMarket() and OstiumTrading::topUpCollateral() can be used to frontrun liquidation calls whose trigger has already been set, making the liquidation fail in OstiumPriceUpKeep::performUpkeep(), draining fees and gaming the system.

Recommendation

Add OstiumTrading::checkNoPendingTrigger() to both functions.

3S-OS-M05

Oracle fees should be payed upfront to protect the protocol from failed `performeUpkeep()` calls

Id	3S-OS-M05
Classification	Medium
Severity	High
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

At the moment oracle fees are payed after the trades have been finalized, either in registerTrade(), openTradeMarketCallback() or closeTradeMarketCallback(). There are some edge cases in which the trade does not get finalized, but the automation/upkeep is still performed, so the protocol incurs these losses. For example, if there is not a trade in openTradeMarketCallback(), fees will not be charged (this may happen if the user cancels the pending market open order after the timeout, but chainlink triggers the upkeep after closing).

Recommendation

Make the user pay oracle fees when they open the trades.

3S-OS-M06

Error in `OstiumPairsStorage::groupMaxCollateral()` calculation

Id	3S-OS-M06
Classification	Medium
Severity	Medium
Likelihood	High
Category	Bug
Status	Addressed in #733206c.

Description

The calculation of the maximum deposited collateral per group, erroneously divides by **`100`**, assuming it refers to 100%. However, the **`maxCollateralP`** variable, specified here, has a precision of 2 decimals. This means that 100% is expressed as **`100_00`**. This discrepancy results in incorrect checks in **`OstiumTradingCallbacks::withinExposureLimits()`**, potentially causing the vault's collateral to be capped at a much larger amount than intended. Furthermore, this miscalculation may dilute the reported profits.

Recommendation

To avoid errors related to magic numbers, utilize constants for precision. Additionally, if using **`100_00`** to denote 100% consistently, it's preferable to adhere to Basis Point (BPS) denomination.

3S-OS-M07

`updatePair()` is missing the **`pairOk()`** modifier

Id	3S-OS-M07
Classification	Medium
Severity	High
Likelihood	Low
Category	Bug
Status	Addressed in #7fe937e.

Description

OstiumPairsStorage::updatePair() updates a pair's attribute but does not check the new values.

Recommendation

Add the pairOk() modifier to **`updatePair()`**.

3S-OS-M08

`OstiumPriceUpKeep::performUpKeep()` does not correctly handle possible reverts

Id	3S-OS-M08
Classification	Medium
Severity	Medium
Likelihood	Medium
Category	Bug
Status	Acknowledged

Description

`OstiumPriceUpKeep::performUpKeep()` should unregister pending market orders or disable triggers and pending limit orders if the execution fails. However, it is not dealing with all the revert scenarios, for example:

1. Not enough value to pay for the verifier.

2. Invalid prices.

Recommendation

If `performUpKeep()` reverts due to one of these reasons, the protocol should be protected and make users pay for it.

If it does not have enough value to pay for the verifier or the prices are invalid, the corresponding user should still pay the oracle fees. Additionally, before calling OstiumPriceUpKeep::getPrice(), a check should be in place to ensure the contract will have enough eth to pay for performUpkeep().

Instead of reverting, it should also unregister the orders and the triggers to allow faster replayability.

3S-OS-M09

Trader Can Set Wrong TP and SL

Id	3S-OS-M09
Classification	Medium
Severity	Medium
Likelihood	Medium
Category	Bug
Status	Acknowledged

Description

In **`OstiumTrading::updateSl()`**, a validation is present for the maximum stop loss (SL) distance. However, there is no validation for the opposite scenario, as seen in **`OstiumTrading::openTrade()`** at L202. This absence of validation permits the setting of a stop loss for an amount greater than the open price when initiating a long position and for setting a smaller amount when initiating a short position. Consequently, this issue introduces the potential for errors, whereby automation could mistakenly trigger the closure of a position upon reaching this price, behaving akin to a take-profit order, contrary to its intended function. The same issue arises in **`OstiumTrading::updateTp()`**, which lacks validation to ensure that the new take profit (TP) is greater than the open price for long positions and smaller for short positions.

Recommendation

Change the validation on L381 in **`OstiumTrading::updateSl()`** to:

```
if (newSl != 0 && (t.buy ? newSl < t.openPrice - maxSlDist || newSl >=
t.openPrice : newSl > t.openPrice + maxSlDist || newSl <= t.openPrice))
revert WrongSL();
```

Add this validation to **`OstiumTrading::updateTp()`**:

```
if (newTp != 0 && (t.buy ? newTp <= t.openPrice : newTp >= t.openPrice)) {
    revert WrongTP();
```

```
}
```

PoC

```
function test_wrong_updateSl() public {
    uint192 newSL = 31000e18; // bigger than open price which is 30000e18
    mockTradingStorage.storeTrade(
        t,
        IOstiumTradingStorage.TradeInfo(
            1, t.collateral * 1e12 * t.leverage / 100 * PRECISION_18 /
t.openPrice, t.leverage, 0, 0, 0, false
        )
    );
    vm.prank(DEFAULT_SENDER);
    trading.updateSl(0, 0, newSL);
}



function test_wrong_updateTp() public {
    uint256 tradeId = 1;
    mockTradingStorage.storeTrade(
        t,
        IOstiumTradingStorage.TradeInfo(
            tradeId, t.collateral * 1e12 * t.leverage / 100 * PRECISION_18 /
t.openPrice, t.leverage, 0, 0, 0, false
        )
    );
    vm.prank(DEFAULT_SENDER);
    vm.expectEmit(true, true, true, true, address(trading));
    emit TpUpdated(tradeId, DEFAULT_SENDER, 0, 0, 28000e18); // lower than
open price = 30000e18
    trading.updateTp(0, 0, 28000e18);
}
```

3S-OS-M10

USDC Blacklisting Prevents Trader Liquidation

Id	3S-OS-M10
Classification	Medium
Severity	High
Likelihood	Low
Category	Bug
Status	Addressed in #6afda46.

Description

USDC incorporates a blacklist functionality that reverts any attempt to transfer funds to a blacklisted address, even if the transfer amount is 0. Consequently, during a liquidation, the **`transfer`** on L495 within **`OstiumTradingCallbacks`** will revert, preventing the liquidation from proceeding and resulting in bad debt for the protocol.

Likewise, other functions such as **`OstiumTrading::cancelOpenLimitOrder()`**, **`OstiumTrading::openTradeMarketTimeout()`**, and **`OstiumTradingCallbacks::openTradeMarketCallback()`** will also revert upon transferring the collateral back to the trader.

A Proof of Concept is available here.

Recommendation

To address this issue, we suggest one or more of the following solutions:

- **Blacklist Check During Borrowing:** Implement a check for USDC blacklisting during the loan initiation process to prevent blacklisted users from opening a loan position until their status is resolved. However, this is not a complete solution, as a user's blacklist status may change between the initiation of a trade and its liquidation.

- **Alter Liquidation Flow:** Eliminate the transfer of remaining USDC tokens to the liquidated user during the liquidation process. Instead, adopt a pull-over-push method, enabling users to withdraw their remaining USDC tokens independently after the liquidation process is complete.

3S-OS-M11

Casting from **`int256`** to **`uint256`** won't revert if the number is negative, possibly leading to issues

Id	3S-OS-M11
Classification	Medium
Severity	High
Likelihood	Low
Category	Bug
Status	Addressed in #38b09ab.

Description

Some instances, namely in OstiumPairInfos::getPendingAccFundingFees(), cast **`int256`** to **`uint256`**.

Recommendation

Avoid casting directly and use a wrapper library instead such as **`SafeCastUpgradeable`**.

3S-OS-M12

Setting `maxFundingFeePerBlock` to a lower value than `abs(lastFundingRate)` will brick `getPendingAccFundingFees()`

Id	3S-OS-M12
Classification	Medium
Severity	Medium
Likelihood	Medium
Category	Bug
Status	Addressed in #6b6d191, #829ad27.

Description

getPendingAccFundingFees() computes the number of blocks to the limit by subtracting **`absLastFundingRate`** to **`maxFundingFeePerBlock`**.

setMaxFundingFeePerBlock() allows setting the max to any value below MAX_FUNDING_FEE.

If **`maxFundingFeePerBlock`** is set to a value smaller than **`absLastFundingRate`** it will underflow, bricking **`getPendingAccFundingFees()`** and it can only be fixed by calling setPairFundingFees().

Here is a POC.

Recommendation

Revert if **`setMaxFundingFeePerBlock()`** is called with **`maxFundingFeePerBlock`** smaller than **`absLastFundingRate`**.

3S-OS-L01

`OstiumTradesUpKeep` Triggers Automation for Pending Orders Expected to Fail

Id	3S-OS-L01
Classification	Low
Severity	Low
Likelihood	High
Category	Bug
Status	Acknowledged

Description

In OstiumTradesUpKeep::_getOpenOrdersToTrigger(), orders are grouped together for execution without some important checks. This means that some orders that should be triggered can fail due to checks in **`OstiumTrading::executeAutomationOrder()`** or in the execution functions of **`OstiumTradingCallbacks`**.

For example, when fetching trades for liquidation in **`OstiumTradesUpKeep::_getOpenOrdersToTrigger()`**, there's no check to see if the trade has a stop loss. This means these orders won't execute because of this check in the **`OstiumTrading::executeAutomationOrder()`**.

Note: this check should also be added to OstiumTradingCallbacks::executeAutomationCloseOrderCallback().

Also, checks for market closure, max allowed leverage per pair and the pause status of the **`OstiumTradingCallbacks`** contract, are missing in OstiumTradesUpKeep::_getLimitOrdersToTrigger(), which means that these orders will fail inside the **`OstiumTradingCallbacks::executeAutomationOpenOrderCallback()`**.

Recommendation

Review the checks in **`OstiumTrading::executeAutomationOrder()`** and the execution functions of **`OstiumTradingCallbacks`**, and make sure similar checks are added to the functions responsible for fetching orders in **`OstiumTradesUpKeep`**. This ensures that only orders likely to succeed are triggered, saving gas.

3S-OS-L02

Limitations for Users Employing Multi-sig or Account Abstraction Wallets in Setting Delegate Address

Id	3S-OS-L02
Classification	Low
Severity	Low
Likelihood	Medium
Category	Bug
Status	Acknowledged

Description

Within the **`Delegatable::setDelegate()`** function, there exists a check to determine if a contract is executing the function, restricting the action solely to Externally Owned Accounts (EOAs). However, this approach may pose issues for users utilizing multi-sig or account abstraction wallets (ERC-4337). Additionally, in accordance with EIP 3074, which introduces the **`AUTH`** and **`AUTHCALL`** EVM instructions, the first one sets a context variable authorized based on an ECDSA signature and the second one sends a call as the authorized account, allowing for the delegation of control from an EOA to a smart contract.

As a result, relying on **`tx.origin`** to ensure **`msg.sender`** is an EOA may not remain valid if EIP 3074 is implemented.

Recommendation

Consider entirely removing this check, as there is a significant likelihood of users employing the mentioned wallets or new EIPs, potentially disrupting the introduced logic.

3S-OS-L03

`executeAutomationOpenOrderCallback` Executes `STOP` Orders at a Worse Price

Id	3S-OS-L03
Classification	Low
Severity	Low
Likelihood	High
Category	Bug
Status	Acknowledged

Description

The condition executed when the `OpenOrderType` is of type `STOP`, as seen here, triggers when the current price is greater than or equal to the target price. This approach results in traders potentially receiving a worse price compared to using `price after impact >= target price`. This is because when buying, the price typically rises, meaning that the target price may be reached sooner than when using the current market price.

Recommendation

Use:

```
cancelReason = (
    o.orderType == IOstiumTradingStorage.OpenOrderType.LIMIT
        ? (o.buy ? priceAfterImpact > o.targetPrice : priceAfterImpact <
o.targetPrice)
-       : (o.buy ? uint192(a.price) < o.targetPrice : uint192(a.price) >
o.targetPrice)
+       : (o.buy ? priceAfterImpact < o.targetPrice : priceAfterImpact >
o.targetPrice)
)
```

3S-OS-L04

`utilizationThresholdP` of `10_000` will make `OstiumPairInfos::_getUtilizationOpeningFee()` divide by 0

Id	3S-OS-L04
Classification	Low
Severity	Medium
Likelihood	Low
Category	Bug
Status	Addressed in #4c774b3.

Description

utilizationThresholdP should never be set to the max `10_000` or it will lead to division by 0 when calling OstiumPairInfos::_getUtilizationOpeningFee().

Recommendation

Add an equality restriction to utilizationThresholdP:

```
if (
    ...
        || value.utilizationThresholdP >= MAX_USAGE_THRESHOLDP ... //@audit
note the =
) {
    revert WrongParams();
}
```

3S-OS-L05

`OstiumVault::lockDiscount()` may revert due to division by 0

Id	3S-OS-L05
Classification	Low
Severity	Medium
Likelihood	Low
Category	Bug
Status	Addressed in #32af17a.

Description

OstiumVault::lockDiscount() reverts if **`maxDiscountThresholdP == uint16(100) * PRECISION_2`**, but the **`constructor`** and **`updateMaxDiscountThresholdP()`** allow setting it to this value.

Recommendation

Also check for equality by using **`<=`** instead of **`<`**.

3S-OS-L06

`reqID_pendingAutomationOrder` stores the index of the trade, which could point to a different trade since the request was created

Id	3S-OS-L06
Classification	Low
Severity	Low
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

`reqID_pendingAutomationOrder` store the **`trader`**, **`pairIndex`** and **`index`**. **`index`** points to the trade in OstiumTradingStorage::openTrades, but the trade itself may have changed since the **`OstiumPriceUpKeep`** automation was requested. A possible scenario is, for example, triggering a take profit, then closing the trade immediately and creating a new trade on **`index`** 0 before the first callback was triggered.

Note: this is also true for open limit orders, it should also check if the **`a.orderId`** matches the corresponding limit order. The easiest way to fix this is adding a **`orderId`** field to **`OpenLimitOrder`** which is filled when **`OstiumTrading::executeAutomationOrder()`** requests a price.

Recommendation

To avoid this problem, OstiumTradingCallbacks::executeAutomationCloseOrderCallback() and OstiumTradingCallbacks::closeTradeMarketCallback() should validate **`a.orderId`** against OstiumTradingStorage::openTradesInfo.tradeId.

3S-OS-L07

`OstiumTradingCallbacks::executeAutomationOpenOrderCallback()` opens limit orders at market price instead of the limit price

Id	3S-OS-L07
Classification	Low
Severity	Low
Likelihood	High
Category	Bug
Status	Acknowledged

Description

Limit orders usually trade at the specified price; however, OstiumTradingCallbacks::executeAutomationOpenOrderCallback() registers the limit order **`t.openPrice`** at **`priceAfterImpact`**, which would be different than the specified.

For example, a limit order at price 5000 will be opened when the price after impact is below it, for example 4950, and the trade will have an open price of 4950, not 5000 as the limit price specified.

This is more profitable for the user as it's buying cheaper, but most likely not the intended behaviour.

Recommendation

If the **`o.orderType == IOstiumTradingStorage.OpenOrderType.LIMIT`**, the open price should be **`o.targetPrice`**. If **`o.orderType == IOstiumTradingStorage.OpenOrderType.STOP`**, the open price should be **`priceAfterImpact`**.

Note that this would change the negative pnl of the trade so that should also be adjusted.

3S-OS-L08

`OstiumPairsStorage::getAllPairsMaxLeverage()` reverts if enough pairs are created

Id	3S-OS-L08
Classification	Low
Severity	Low
Likelihood	Medium
Category	Bug
Status	Addressed in #0d2334b.

Description

OstiumPairsStorage::getAllPairsMaxLeverage() reverts if enough pairs are created due to OOG.

Recommendation

Add an additional function **`OstiumPairsStorage::getAllPairsMaxLeverage(uint256 startId, uint256 finalId)`** to fetch paginated information.

3S-OS-L9

`Ownable2Step` is recommended over `Ownable`

Id	3S-OS-L9
Classification	Low
Severity	Low
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

`Ownable` uses a dangerous pattern of setting the address without proper checks. This means that if a mistake is made and ownership is transferred to the wrong address, the `owner` functionalities would be forever lost.

Recommendation

Ownable2Step eliminates this risk by using a 2 step pattern by setting a pending governor and having it accept ownership.

3S-OS-L10

`OstiumRegistry` should disable renouncing ownership if it is never intended

Id	3S-OS-L10
Classification	Low
Severity	Low
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

[

OstiumRegistry](https://github.com/0xOstium/smart-contracts-threeSigma/blob/audit-feedback/src/OstiumRegistry.sol#L9) allows renouncing ownership by inheriting **`Ownable`**, which could be catastrophic if triggered by mistake.

Recommendation

Consider overriding **`renounceOwnership()`** and revert if called to disable this functionality.

3S-OS-L11

`OstiumLinkUpKeep:topUp()` is missing a length check for `registryAddresses`

Id	3S-OS-L11
Classification	Low
Severity	Low
Likelihood	Low
Category	Bug
Status	Addressed in #ae32282.

Description

OstiumLinkUpKeep:topUp() checks `if (upkeepIDs.length != topUpAmounts.length) revert WrongParams();`, but does not include the length of `registryAddresses`, which is expected to be equal, as seen in OstiumLinkUpKeep:getUnderfundedUpkeeps().

Recommendation

Either do:

```
if (upkeepIDs.length != topUpAmounts.length && topUpAmounts.length != registryAddresses.length) revert WrongParams();
```

or use an array of `struct`s containing the 3 parameters.

3S-OS-L12

`.values()` may revert when calling `getWatchList()` due to OOG

Id	3S-OS-L12
Classification	Low
Severity	Low
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

If the number of registries in the s_registries set grows too large, it will revert when calling getWatchList() due to OOG.

Recommendation

Add a paginated function to get the watch list.

3S-OS-L13

`OstiumTradingCallbacks::executeAutomationOpenOrderCallback()` reverts if it can not find the `openLimitOrder`

Id	3S-OS-L13
Classification	Low
Severity	Medium
Likelihood	Low
Category	Bug
Status	Addressed in #295430f, #4688377.

Description

OstiumTradingCallbacks::executeAutomationOpenOrderCallback() fetches the `openLimitOrder` before checking for its existence, which will end up reverting if it does not exist, not finishing execution.

Recommendation

Check the existence of the limit order before fetching it.

3S-OS-L14

`OstiumTrading::executeAutomationOrder()` and `_getLimitOrdersToTrigger()` should check `isPaused` for open limit orders

Id	3S-OS-L14
Classification	Low
Severity	Medium
Likelihood	Low
Category	Bug
Status	Acknowledged

Description

`OstiumTrading::openTrade()` correctly checks `isPaused` to not allow opening positions when the protocol is paused. However, executeAutomationOrder() still allows execution if it is an open limit order, which it should not, as it will fail when OstiumPriceUpKeep()::performUpKeep() executes. Moreover, it will not discount oracle fees if it is paused or the market has been closed.

Recommendation

Check if `OstiumTradingCallbacks` is paused in `OstiumTradingUpKeep()` and in `OstiumTrading::executeAutomationOrder()` for open limit orders.

3S-OS-L15

In **`OstiumLinkUpKeep`**, empty **`watchlist`** or **`keeperIds`** arguments are not handled

Id	3S-OS-L15
Classification	Low
Severity	Low
Likelihood	Low
Category	Bug
Status	Addressed in #c9ebe98.

Description

In **`OstiumLinkUpKeep`**, functions setWatchList(), addToWatchList() and removeFromWatchList() don't handle 0 length array inputs, which could lead to incorrect state.

Recommendation

Explicitly check for empty arrays.

3S-OS-L16

Using `transfer()` instead of `call()` may revert

Id	3S-OS-L16
Classification	Low
Severity	Low
Likelihood	Low
Category	Bug
Status	Addressed in #173093.

Description

When withdrawing ETH in `OstiumPriceUpKeep::withdrawEth()` using the deprecated `transfer()` function will make the transaction revert when:

1. The claimer smart contract does not implement a payable function.

2. The claimer smart contract does implement a payable fallback which uses more than 2300 gas units.

3. The claimer smart contract implements a payable fallback function that needs less than 2300 gas units but is called through proxy, raising the call's gas usage above 2300.

Additionally, using higher than 2300 gas might be mandatory for some multisig wallets.

Recommendation

Use sendValue() or:

```
payable(msg.sender).call{value: amount}("");
```

3S-OS-L17

Missing **`disableInitializers()`** call in the constructor

Id	3S-OS-L17
Classification	Low
Severity	Low
Likelihood	Medium
Category	Bug
Status	Addressed in #1a1c899.

Description

When using **`Initializable.sol`**, it's a good practice calling **`disableInitializers()`** in the constructor, such that the implementation itself can't be initialized.

Recommendation

Use:

```
constructor() {
     _disableInitializers();
}
```

3S-OS-N01

Implement Storage Gap in **`Delegatable`** Contract

Id	3S-OS-N01
Classification	None
Category	Good Code Practices
Status	Acknowledged

Description

If additional variables are added to the **`Delegatable`** contract, conflicts may arise with the storage slots previously written on the proxy.

Recommendation

To mitigate this issue, insert a storage gap in the contract:

```
abstract contract Delegatable is IDelegatable {
    mapping(address => address) public delegations;
    address private senderOverride;
@>  uint256[49] __gap;
    // *** code ***
```

3S-OS-N02

Unused Trade Size Variable in `OstiumTrading::executeAutomationOrder()`

Id	3S-OS-N02
Classification	None
Category	Optimization
Status	Addressed in #27e6f2c.

Description

The **`leveragedPos`** variable in **`OstiumTrading::executeAutomationOrder()`** is calculated on L443 and L465, but its value isn't utilized anywhere in the function.

3S-OS-N03

Implement Custom Errors Instead of **`require`** Statements

Id	3S-OS-N03
Classification	None
Category	Optimization
Status	Addressed in #7565043.

Description

The usage of **`require`** statements instead of custom errors is evident in both **`OstiumLinkUpKeep`** and particularly in **`OstiumVault`**.

Recommendation

It is advisable to opt for custom errors as they offer a more gas-efficient method to explain to users why an operation failed. Additionally, utilizing custom errors is more cost-effective during deployment.

3S-OS-N04

Inconsistent Precision of Percentage Variables

Id	3S-OS-N04
Classification	None
Category	Good Code Practices
Status	Acknowledged

Description

Throughout the codebase, variables representing percentages maintain a precision of 2 decimal places. However, there are instances where this precision is not consistently applied:

- The **`MAX_GAIN_P`** variable on L31 in **`OstiumTradingCallbacks`**.

- The **`maxSl_P`** variable on L47 in **`OstiumTradingCallbacks`**.

- The **`liqThresholdP`** variable in **`OstiumPairInfos`**.

- The **`liqFeeP`** variable of the **`Fee`** struct in **`OstiumPairsStorage`**.

Recommendation

It is advisable to review the contracts and ensure that variables representing percentages consistently adhere to a precision of two decimal places.

3S-OS-N05

Unnecessary Typecasting of `msg.sender`

Id	3S-OS-N05
Classification	None
Category	Optimization
Status	Addressed in #0b6bca6.

Description

Unnecessary typecasting of **`msg.sender`** to **`address`** is seen on L34 in **`OstiumLinkUpKeep::_onlyGov()`** and on L32 in **`OstiumRegistry::_onlyGov()`**. This redundancy is unnecessary as **`msg.sender`** already returns an address.

Recommendation

To improve efficiency and clarity, remove the typecasting operation to save gas and eliminate confusion.

3S-OS-N06

Use of Magic Numbers

Id	3S-OS-N06
Classification	None
Category	Good Code Practices
Status	Acknowledged

Description

Several instances of magic numbers are present throughout the contracts:

- The validation for **`slippageP`** in **`OstiumTrading::openTrade()`** on L169.

- L117 in **`OstiumOpenPnl::getOpenPnl()`**.

- L401 in **`OstiumTrading::topUpCollateral()`**.

- L149 in **`OstiumTradingCallbacks::openTradeMarketCallback()`**.

- L446 in **`OstiumTradingCallbacks::registerTrade()`**.

Recommendation

It is recommended to review the contracts and replace these magic numbers with constants for improved clarity. Not all instances of them are mentioned in the description.

3S-OS-N07

Unused and Wrong Imports

Id	3S-OS-N07
Classification	None
Category	Good Code Practices
Status	Addressed in #840e77e.

Description

Unused Imports

OstiumVault

```
import './lib/ChainUtils.sol';
import '@openzeppelin/contracts/access/Ownable.sol';
import '@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol';
```

OstiumWhitelist

```
import '@openzeppelin/contracts/utils/Address.sol';
```

OstiumLockedDepositNft

```
import '@openzeppelin/contracts/interfaces/IERC20Metadata.sol';
import '@openzeppelin/contracts/interfaces/IERC4626.sol';
```

OstiumPriceUpKeep

```
import './lib/ChainUtils.sol';
import './interfaces/IOstiumTradingStorage.sol';
```

OstiumTradesUpKeep

```
import 'src/interfaces/external/IChainlinkVerifierProxy.sol';
import 'src/interfaces/external/IChainlinkRewardManager.sol';
import 'src/interfaces/external/IChainlinkFeeManager.sol';
import
'@chainlink/contracts/src/v0.8/automation/interfaces/ILogAutomation.sol'; //
duplicate
```

OstiumTradingStorage

```
import './interfaces/IOstiumOpenPnl.sol';
```

Wrong imports

OstiumTradesUpKeep

```
import 'src/OstiumTrading.sol';
```

should be changed to

```
import 'src/interfaces/IOstiumPairInfos.sol';
```

Because it's only using the**`IOstiumPairInfos`** interface.

OstiumPairsStorage

```
import './interfaces/IOstiumTradingStorage.sol';
```

should be changed to

```
import './interfaces/IOstiumPairsStorage.sol';
```

Because it's only using the**`IOstiumPairsStorage`** interface.

3S-OS-N08

Redundant Calculations in **`OstiumTrading::openTrade()`**

Id	3S-OS-N08
Classification	None
Category	Optimization
Status	Addressed in #3d116cd.

Description

When opening a **`LIMIT`** or **`STOP`** order, the first empty index is calculated using **`OstiumTradingStorage::firstEmptyOpenLimitIndex()`** on L209. However, the same function is executed again in **`OstiumTradingStorage::storeOpenLimitOrder()`** on L233, which results in the **`index`** property being overridden with the same value. Similarly, the current block number is retrieved on L211 in **`OstiumTrading`**, and then on L231 in **`OstiumTradingStorage`**.

Recommendation

Remove the redundant overriding of these variables from **`OstiumTradingStorage::storeOpenLimitOrder()`**.

3S-OS-N09

`OstiumTrading::closeTradeMarket()` Makes Redundant Call to `OstiumTradingStorage::getOpenTradeInfo()`

Id	3S-OS-N09
Classification	None
Category	Optimization
Status	Addressed in #155ab21.

Description

At L284 in **`OstiumTrading::closeTradeMarket()`**, the trade info can be reused from the previous call to **`OstiumTradingStorage::getOpenTradeInfo()`** at L265 to conserve gas by avoiding redundant function calls.

The same issue is seen in **`OstiumTrading::closeTradeMarketTimeout()`** on L561 and L568

Recommendation

```
emit MarketCloseOrderInitiated(
-    orderId, storageT.getOpenTradeInfo(sender, pairIndex, index).tradeId,
sender, pairIndex
+    orderId, i.tradeId, sender, pairIndex
);
```

3S-OS-N10

`OstiumLinkUpKeep` Config Not Set In `initialize()`

Id	3S-OS-N10
Classification	None
Category	Optimization
Status	Acknowledged

Description

The variable s_config is not set in the initializer function, which is preferable.

Recommendation

Although **`setConfig`** is called right after the deployment of the contract in the deployer script, it will be more efficient to pass the config to the initializer and call **`setConfig`** there.

3S-OS-N11

`OstiumPriceUpKeep` Verifies Reports Using Native Token Instead of `LINK`

Id	3S-OS-N11
Classification	None
Category	Optimization
Status	Acknowledged

Description

The Chainlink documentation states that verifying data streams via native blockchain gas tokens and their ERC20-wrapped versions incurs a surcharge when compared to LINK payments. However, the payment for verification on L126 is conducted using the native token, suggesting that it would be more cost-effective for the protocol to use `LINK` for payments.

Recommendation

It is advised to consult the specific section within the documentation that details the implementation of payments using `LINK` and follow those guidelines.

3S-OS-N12

`OstiumTrading::updateOpenLimitOrder()` Makes Unnecessary Field Updates

Id	3S-OS-N12
Classification	None
Category	Optimization
Status	Addressed in #d825181.

Description

In **`OstiumTrading::updateOpenLimitOrder()`**, unnecessary fields of the **`OpenLimitOrder`** struct are updated. Specifically, the **`createdAt`** field is set, despite its intended immutability after creation, and the **`lastUpdated`** field is also updated. Subsequently, this object is passed to **`OstiumTradingStorage::updateOpenLimitOrder()`**, where **`lastUpdated`** is reassigned to the current block. However, the **`createdAt`** field remains unaltered as intended.

Recommendation

Do the following changes in **`OstiumTrading::updateOpenLimitOrder()`**:

```
- uint32 b = ChainUtils.getBlockNumber().toUint32();
  o.targetPrice = price;
  o.tp = tp;
  o.sl = sl;
- o.createdAt = b;
- o.lastUpdated = b;
```

3S-OS-N13

`OstiumTrading::canExecute()` should also be checked in `OstiumTradesUpKeep::checkCallback()`

Id	3S-OS-N13
Classification	None
Category	Bug, Optimization
Status	Addressed in #20d8cee.

Description

Orders in OstiumTrading::executeAutomationOrder() will not execute if they are in the timeout period. This same check should exist in OstiumTradesUpKeep::checkCallback() to ensure that non executable orders are added to **`tradesToTrigger`**.

Recommendation

Add the timeout check to **`OstiumTradesUpKeep::checkCallback()`**.

3S-OS-N14

Unused `UPDATE_SL` `OstiumPriceUpKeep::OrderType`

Id	3S-OS-N14
Classification	None
Category	Good Code Practices
Status	Addressed in #ab4e1ae.

Description

Enum **`OrderType`** in **`IOstiumPriceUpKeep`** has a **`UPDATE_SL`** property which is never used.

Recommendation

Remove this property if it is not used.

3S-OS-N15

In `OstiumTradesUpKeep`, `_getLimitOrdersToTrigger()` and `_getOpenOrdersToTrigger()` return tradesToTrigger with 1 extra length

Id	3S-OS-N15
Classification	None
Category	Bug
Status	Addressed in #8ff89f3.

Description

In OstiumTradesUpKeep::_getLimitOrdersToTrigger() and OstiumTradesUpKeep::_getOpenOrdersToTrigger(), **`tradesToTrigger`** are returned with 1 extra length due to incorrect **`++tradesToTrigger`** operation. **`tradesToTriggerIndex`** starts at 0 and increases whenever a trade is found, so it is always ahead of the biggest index by 1 already, representing the length.

Recommendation

Remove **`++`** from the conditions **`if (++tradesToTriggerIndex < tradesToTrigger.length)`**.

Additionally, check if the length of the array here is **`0`** instead of the first trader.

3S-OS-N16

Wrong decoding of **`verifierResponse`** in **`OstiumPriceUpKeep::performUpkeep()`**

Id	3S-OS-N16
Classification	None
Category	Bug
Status	Addressed in #3717915.

Description

OstiumPriceUpKeep::performUpkeep() decodes the **`verifierResponse`** setting an **`expiresAt`** as uint192, when it is in fact a **`uint32`** in both Basic and Premium reports. This has no impact as the variables are encoded as **`uint256`** with **`abi.encode()`**, but should be addressed.

Note: here too.

Recommendation

Instead of decoding each element separately, decode to a **`BasicReport`** or **`PremiumReport`** struct.

3S-OS-N17

`updateGroupCollateral()` should revert if the `_pairIndex` does not exist

Id	3S-OS-N17
Classification	None
Category	Bug
Status	Addressed in #ee024b2.

Description

OstiumPairsStorage::updateGroupCollateral() does not check for the pair's index existence, which means that it would update group with index `0` incorrectly.

Recommendation

Add `if (!isPairIndexListed[_pairIndex]) revert PairNotListed(_pairIndex);` to the function.

3S-OS-N18

Error parameters should be more descriptive

Id	3S-OS-N18
Classification	None
Category	Good Code Practices
Status	Acknowledged

Occurences

`OstiumLinkUpkeep`, WrongParams.

3S-OS-N19

Faulty revert reason decoding in **`Delegatable`**

Id	3S-OS-N19
Classification	None
Category	Bug
Status	Addressed in #83ee4a7.

Description

`Delegatable` decodes the revert reason of a revert in a way that may fail. It adds 4 bytes to the **`result`** pointer, which may mess up the length of the **`result`** variable, see more details here.

Recommendation

Don't decode the reason.

3S-OS-N20

`OstiumOpenPnl::average()` can be simplified by adding a state variable that tracks the cumulative sum of **`nextEpochValues`**

Id	3S-OS-N20
Classification	None
Category	Optimization
Status	Acknowledged

Description

OstiumOpenPnl::average() calculates the average of all the **`nextEpochValues`**, which is an expensive operation (depending on **`requestsCount`** and could be replaced by a state variable tracking the cummulative sum.

3S-OS-N21

`OstiumOpenPnl::nextEpochValuesRequestCount` stores the same information as **`nextEpochValues.length`**

Id	3S-OS-N21
Classification	None
Category	Optimization
Status	Acknowledged

Description

OstiumOpenPnl::nextEpochValuesRequestCount can be removed as it is the same as checking nextEpochValues.length.

3S-OS-N22

Redundant `abi.decode()` in `OstiumTradesUpKeep::checkCallback()`

Id	3S-OS-N22
Classification	None
Category	Optimization
Status	Addressed in .

Description

The second decoding on L126 of the `data` parameter is identical to the first and does not assign its result to any variable. This appears to be redundant and serves no discernible purpose.

Recommendation

Remove the duplicate `abi.decode()` call.

3S-OS-N23

Consider using `forceApprove` instead of `safeApprove` in `OstiumTraidingCallbacks`

Id	3S-OS-N23
Classification	None
Category	Good Code Practices
Status	Addressed in #55fde4e.

Description

`safeApprove` should only be called when setting an initial allowance, because it reverts when a non-zero approval is changed to a non-zero approval. Even though it's only called with either `type(uint256).max` or `0`, using forceApprove is safer.

3S-OS-N24

`OstiumLinkUpKeep:removeFromWatchList()` complexity could be reduced

Id	3S-OS-N24
Classification	None
Category	Optimization
Status	Acknowledged

Description

OstiumLinkUpKeep:removeFromWatchList() currently iterates over the full **`s_registryWatchLists`** to find the **`keeperIds`** and delete them if equal. This has O(n^2) complexity, which could cost a significant amount of gas.

Recommendation

One way to tackle this problem is tweaking the **`isInWatchList`** mapping to store the index of the array instead of a **`bool`**, adding 1 to each index so that 0 is not a valid index and can be used to check for abcense. This way, when removing, the index can be fetchd from the corresponding mapping.

3S-OS-N25

Typos in **`OstiumLinkUpKeep`**

Id	3S-OS-N25
Classification	None
Category	Good Code Practices
Status	Addressed in #119db4a.

Description

OstiumLinkUpKeep has typos in errors **`WhatchlistEmpty`** , **`HasWhatchlist`** and **`NoWhatchlist`**.