GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

PariFi

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Kiki, ABA, EVDoc, VanGrim, Dimo

Client Firm PariFi Finance

Final Report Date November 15, 2023

Audit Summary

PariFi engaged Guardian to review the security of its decentralized synthetics perpetuals exchange. From the 20th of October to the 1st of November, a team of 7 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Arbitrum**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/PariFi-10-2023

Table of Contents

Project Overview

Project Summary

Project Name	PariFi Finance
Language	Solidity
Codebase	https://github.com/Parifi/parifi-contracts-internal
Commit(s)	Initial: b7c2307de2d762ac4f4296c697dd571fc246e922 Final: 04b1ea8d4895638bd4e5466c4b7a71625517077e

Audit Summary

Delivery Date	November 15, 2023
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	1	0	0	0	0	1
High	5	0	0	1	0	4
Medium	11	0	0	2	0	9
Low	18	0	0	6	0	12

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
FM	FeeManager.sol	5106dc7f4d404a025438a91f2c256f25c5c56b6f
VAULT	ParifiVault.sol	851a11684455e2d5c34b201d8e02f8f14301e84e
ADAP	Adapter.sol	75db5b1c89ffa7118fb567565718a8dfcc04e78f
DATA	DataFabric.sol	f0cfdb56f767f0f750fa4e67abc9692e59511990
FWD	ParifForwarder.sol	5e6637806cfa2ec22f9d60379dc4a9922452dc44
LIBE	LibError.sol	70aa0ffccb5b445818c1db735ed5bcd73b536d04
ODS	OrderDS.sol	c33a26b55e8eaf5de5cc227218b5ed07aa7a9e7f
ORDM	OrderManager.sol	b9519041c330a71111606b0c9783cdf696df0411
RB	RBAC.sol	a32c576552f12e7b9010684c50e289b3adea584b
FEED	PriceFeed.sol	db7ed6bd1d1b1099bc8110c4f08abcef8d586290
IPV	IParifiVault.sol	3c3e46bcbb2ae7951e56d561dc9777b9cff59470
IRB	IRBAC.sol	9a536452c28452644fce409910bfac18c5d134f4
IDF	IDataFabric.sol	0e10798d53cd6877d75675ffa71394028dd7e2d4
IPF	IPriceFeed.sol	459b17f2bc5c35f6946bb0eac12e8ccc81b53bea

Audit Scope & Methodology

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Methodology

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.
- Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
ORDM-1	Attacker Can Drain OrderManager	Logical Error	Critical	Resolved
ORDM-2	Position Created Without Reserves	Logical Error	High	Acknowledged
ORDM-3	Trapped collateralDelta With Increase Order	Logical Error	High	Resolved
VAULT-1	Withdrawal Cooldown Can Be Bypassed	Protocol Manipulation	High	Resolved
ORDM-4	Execution Fee May Be Circumvented	Logical Error	High	Resolved
ORDM-5	Impossible to Liquidate When Fee is Greater Than PnL	Logical Error	High	Resolved
DATA-1	OI Validation Leads To Skew	Logical Error	Medium	Resolved
GLOBAL-1	Risk-Free Trade During Equity Events	Price Feeds	Medium	Acknowledged
ORDM-6	Liquidations Fail On Price Drops	Logical Error	Medium	Resolved
ORDM-7	User Can Decrease Position Below Minimum Collateral	Logical Error	Medium	Resolved
DATA-2	Updating A Market After Pause Incurs Fees	Logical Error	Medium	Resolved
ORDM-8	Positions Can Be Liquidated In A Paused Market	Logical Error	Medium	Resolved
ORDM-9	Average Price of a Position is Miscalculated	Logical Error	Medium	Acknowledged
ORDM-10	Liquidations Revert With 0 netPnl	DoS	Medium	Pending

Findings & Resolutions

ID	Title	Category	Severity	Status
ORDM-10	Liquidations Revert With 0 netPnl	DoS	Medium	Resolved
DATA-3	Changing Market Settings Applies Fees Retroactively	Logical Error	Medium	Resolved
DATA-4	User Can Add Collateral When Market Is Set To closeOnly mode	Logical Error	Medium	Resolved
FEED-1	Price May Be 0	Precision	Medium	Resolved
VAULT-2	Vault Is Not ERC4626 Compliant	Specification	Low	Resolved
ORDM-11	Position Can Be Liquidated on Creation	Logical Error	Low	Acknowledged
ORDM-12	Stale Market Fees Used	Logical Error	Low	Acknowledged
FM-1	Inaccurate Comment On Fee Distribution	Documentation	Low	Resolved
ORDM-13	Fees Are Charged When Orders are Cancelled	Documentation	Low	Acknowledged
ORDM-14	_validateMarket Redundantly Called	Optimization	Low	Resolved
ORDM-15	User Position Stuck When Blacklisted	Logical Error	Low	Acknowledged
DATA-5	Liquidation Threshold at Max Will Put Protocol at Loss	Logical Error	Low	Resolved
DATA-6	getExpectedUtilization Lacks OI Validation	Validation	Low	Resolved
RB-1	Initial Multisig Address With DEFAULT_ADMIN_ROLE	Logical Error	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
VAULT-3	ParifiVault.cooldown missing whenNotPaused modifier	Logical Error	Low	Resolved
ORDM-16	Execution Ordering Is Not Guaranteed During Settlement	Validation	Low	Acknowledged
RB-2	Not Following A 2 step ADMIN Role Transfer	Optimization	Low	Resolved
FM-2	distributionFee DoS	DoS	Low	Resolved
ORDM-17	Lacking Referrer Incentive	Incentives	Low	Acknowledged
ORDM-18	Inefficient Price Computation	Optimization	Low	Resolved
ORDM-19	safeTransferFrom Should Occur Before Updates	Reentrancy	Low	Resolved
ORDM-20	orderToPositionId Not Cleared On Settlement	Logical Error	Low	Resolved

ORDM-1 | Attacker Can Drain OrderManager

Category	Severity	Location	Status
Logical Error	● Critical	OrderManager.sol: 824	Resolved

Description PoC

When creating an order with the function createNewPosition(), a user can send an executionFee to the feeReceiver. The feeReciever is different from the orderManager, meaning the executionFee will not be in the orderManager.

When cancelling an order the executionFee is returned to the user:

```
if (userOrder.executionFee != 0) {
  userBalance = userBalance + userOrder.executionFee;
}
...
if (userBalance != 0) {
  IERC20(market.depositToken).safeTransfer(userAddress, userBalance);
}
```

An attacker can create a order with a large execution fee and then cancel that order. By doing so, they can drain the orderManager as the orderManager is the one paying the refund, while the original funds are with the feeReceiver.

_closePosition(), _decreasePosition(), _increasePosition(),_liquidatePosition() will no longer fully work as all of these functions will eventually transfer the collateral either back to the user or to the vault. However, due to this attack the orderManager will not have sufficient funds to cover these transactions as it will have zero collateral.

Recommendation

Either do not refund the executionFee to the users or leave the executionFee in orderManager until settlement occurs. At that point, the keeper can pull the executionFee.

Resolution

PariFi Team: The issue was resolved in commit d8dca2f.

ORDM-2 | Position Created Without Reserves

Category	Severity	Location	Status
Logical Error	High	OrderManager.sol: 644	Acknowledged

Description

A user's position is created without taking the funds in the Vault into account. As a result, a trader may open a position when the Vault has zero funds, or not enough funds to support the market's PnL.

Users will be unable to realize their profits and be stuck with their position. Furthermore, as soon as there are enough reserves in the Vault for withdrawal, the Vault's funds will be drained and LP's will lose their deposited funds.

Recommendation

Validate that the open interest does not exceed some percentage of the Vault's funds. This would create a buffer and help avoid a scenario where the pool does not have enough liquidity to support user profits.

Resolution

PariFi Team: Acknowledged

ORDM-3 | Trapped collateralDelta With Increase Order

Category	Severity	Location	Status
Logical Error	High	OrderManager.sol: 820	Resolved

Description

If a user creates an increase order and provides a `deltaCollateral` that is less than their `openingFee`, they cannot cancel this order and their funds will be stuck if the order cannot be executed.

Additionally, there is no opening fee charged when a decrease or close order is cancelled. This is in contradiction to the behavior of increase orders.

Recommendation

Refactor the way the opening fee is charged for increase, decrease and close orders. Consider requiring that the opening fee be provided up front in the `modifyPosition` function even for decrease or close orders.

Alternatively, consider only charging the opening fee when an order is executed and instead maintaining a fraction of the `executionFee`. It would then be prudent to ensure that the `executionFee` is above a 0 or trivial amount.

Resolution

PariFi Team: The issue was resolved in commit d8dca2f.

VAULT-1 | Withdrawal Cooldown Can Be Bypassed

Category	Severity	Location	Status
Protocol Manipulation	High	ParifiVault.sol	Resolved

Description PoC

In ParifiVault, the cooldown function only checks that the balance of a sender is not 0. It does not check how many tokens the user has or the amount that could be withdrawn after the cooldown period. A user may:

- Prepare X addresses for which it sends 1 WEI of share tokens
- Call the ParifiVault.cooldown function from each address
- Rotate this system in order to also bypass the expiry window constraint
- Whenever a user wishes to withdraw any amount of LPs, they send all token shares to one of the pre-warmed addresses and withdraw reserves

Because the cooldown can be avoided, a depositor can view a profitable position but withdraw their liquidity without waiting. This is extremely detrimental to traders as they will be unable to withdraw their profits due to the lack of reserves.

Due to the bypassed cooldown, profit from the vault may also be extracted with the following steps:

- Flash-loan a large amount of tokens
- Deposit the tokens into the vault
- Create a new position that triggers fee distribution and increases the value per share
- Withdraw tokens using a pre-warmed address and profit

Recommendation

Note the balance of users that call the cooldown function and allow a maximum of that amount to be withdrawn in redeem and withdraw functions. Alternatively, reset the cooldown on shares transfer and deposit.

Resolution

PariFi Team: The issue was resolved in commit 6c07f1a.

ORDM-4 | Execution Fee May Be Circumvented

Category	Severity	Location	Status
Logical Error	High	OrderManager.sol: 691	Resolved

Description

The current execution fee process only charges the user if the user-supplied order specifies a non-zero execution fee.

```
if (_order.executionFee != 0) {
  _chargeExecutionFee(orderId, market.depositToken, _order.executionFee,_order.userAddress);
}
```

Because there is no requirement that the execution fee must be non-zero, a user has no incentive to pass a non-zero execution fee and pay extra for their order.

As a result, keepers will not be properly remunerated for settling orders and liquidations. This can lead to griefing as the protocol must pay a fee each time the oracle price is updated, in addition to the gas needed for execution.

Recommendation

Create a state variable for the execution fee and validate that it matches the execution fee passed on the order.

Resolution

PariFi Team: The issue was resolved in commit d8dca2f.

ORDM-5 | Impossible to Liquidate When Fee is Greater Than PnL

Category	Severity	Location	Status
Logical Error	High	OrderManager.sol 600	Resolved

Description PoC

In the _liquidatePosition function the PnlRealized event is emitted which includes the pnlInCollateral. This is calculated by taking the netPnl and subtracting feesInCollateral. However, the feesInCollateral may be greater than netPnl, causing an underflow revert.

```
emit PnlRealized(_positionId, false, netPnl - feesInCollateral, feesInCollateral,
executionPrice);
```

In the _getNetProfitOrLossIncludingFees function which is called in _liquidatePosition, if the position is in profit excluding fees it will enter the inner if-statement the isNetProfit will be set to false and feesInCollateral will be greater than netPnl.

At the end of the _liquidatePosition function when the PnlRealized event is emitted the transaction will revert, making any liquidation impossible until the position is completely underwater, leading to a loss of yield for the protocol and LP's.

Recommendation

Since netPnl and fees will be a loss for the position and go to the feeManager anyway, do not emit an event where netPnl - feesInCollateral is being calculated. Instead, have a separate event for liquidations where only the netPnl is being emitted.

Resolution

PariFi Team: Resolved in commit 3b5f6b.

DATA-1 | OI Validation Leads To Skew

Category	Severity	Location	Status
Logical Error	Medium	DataFabric.sol: 435	Resolved

Description

The validation to ensure the maximum open interest is not exceeded compares both trading sides in aggregate:

```
if (config.totalLongs + config.totalShorts > (2 * config.maximumOi)) revert LibError.MaxOI();
```

With the current open interest (OI) validation, longs are able to dictate how much in shorts can be opened and vice versa. For example, if traders establish 1800 ETH in long OI, only 200 ETH in short OI can be opened when the `maximumOi` is set to 1000 ETH. This inherently leads the market to be imbalanced.

Recommendation

Validate open interest per side rather than in aggregate.

Resolution

PariFi Team: Resolved in commit [e186c87](e186c87).

GLOBAL-1 | Risk-Fee Trade During Equity Events

Category	Severity	Location	Status
Price Feeds	Medium	Global	Acknowledged

Description

Pyth provides price feeds for numerous US equities with significant dividends. A position on a share that pays a dividend will see its price adjusted to reflect the dividend payment. For example, with a dividend of $1 per share, a stock that was trading at $100 will drop to $99 on the ex-dividend date. Furthermore, stocks may go through splits and reverse stock splits, drastically changing the price of a share.

Consider the following scenario:

- A trader anticipates a stock split so they sell 1 share for $100
- A 2:1 stock split occurs and the new price is $50
- The trader closes their short, making a risk-free profit.

Recommendation

Exercise caution with which markets are supported for trading and carefully monitor for equity events as they are announced in advanced. In anticipation of an event, put the market in close-only mode and pause the market afterwards to prevent further trading. Ensure the market starts from a clean slate post-event.

Resolution

PariFi Team: Acknowledged.

ORDM-6 | Liquidations Fail On Price Drops

Category	Severity	Location	Status
Logical Error	Medium	OrderManager.sol: 188-191	Resolved

Description

In the case of a steep price move (UST for example), the protocol needs to be able to perform liquidations to ensure the system remains solvent. During this volatility, the percentage difference between the lagging EMA and the current price may exceed the `market.maxPriceDeviation` and revert, causing liquidations to fail.

Recommendation

Consider simply fetching and utilizing the `primaryPrice` for liquidation. Because the price feed is updated prior to liquidation, the call `priceFeed.getMarketPricePrimary()` should not revert.

It would also be worth adding a confidence interval when fetching only the `primaryPrice` to mitigate any potential price manipulation.

Resolution

PariFi Team: Resolved in commit 352ff02.

ORDM-7 | User Can Decrease Position Below Minimum Collateral

Category	Severity	Location	Status
Logical Error	Medium	OrderManager.sol: 698	Resolved

Description

In `createNewPosition` there is a check that prevents an order from being created if the collateral is below a set minimum. This is in part to ensure that liquidations are profitable. However, a user can decrease the position so that the collateral is below the minimum, making liquidations not profitable.

In addition, normal users can unintentionally decrease the position to such a low level that they don't bother to close it. Because the position is not being closed and may end up being unprofitable to liquidate, these small positions will reduce the available OI until the keepers opt to liquidate the position which at that point they will be liquidating at a loss.

Recommendation

Add a minimum collateral check in `modifyPosition` to ensure the collateral remains above a desired minimum.

Resolution

PariFi Team: Resolved in commit 352ff02.

DATA-2 | Updating A Market After Pause Incurs Fees

Category	Severity	Location	Status
Logical Error	Medium	DataFabric.sol: 546-547	Resolved

Description

Updating an existing market, by calling the `updateExistingMarket` function from the `DataFabric` contract, incorrectly also calculates market fees up to that point. This is because it also includes a call to the `_updateCumulativeFees` function, which is responsible for updating fees up to that point.

The `updateExistingMarket` function cannot be called if the market is not paused, but pausing the market does not fast forward `feeLastUpdatedTimestamp`, only unpausing does. For the time since the market was paused and until it was updated by the admin, the market will incorrectly deduct fees from participants.

Recommendation

Delete the call to the `_updateCumulativeFees` function since a call to it is already done in the `pause` function.

Resolution

PariFi Team: Resolved in commit 62c8818.

ORDM-8 | Positions Can Be Liquidated In A Paused Market

Category	Severity	Location	Status
Logical Error	Medium	OrderManager.sol: 545-549	Resolved

Description

Liquidating a position, by calling the function `liquidatePosition` from the `OrderManager` contract, does not check if the market in which the order was placed is currently paused or not. Any already existing positions can be liquidated but users cannot cancel or add to them during pause by design.

Another issue that appears when a liquidation is done on an order in a paused market is triggering market fee payment. This takes place as the function `updateCumulativeFees` from the `DataFabric` will get called.

Recommendation

Add a call to the `_validateMarket` function at the beginning of the `_liquidatePosition` function in the `OrderManager` contract.

Resolution

PariFi Team: Resolved in commit e186c8.

ORDM-9 | Average Price of a Position is Miscalculated

Category	Severity	Location	Status
Logical Error	Medium	OrderManager.sol: 429, 464	Acknowledged

Description

In `_increasePosition` and `_decreasePosition` the user's position is essentially recreated with a modified `positionSize` and/or `positionCollateral`. When the position is recreated, the `userPosition.avgPrice` is set to the `updatedAvgPrice`, which is based on the `deltaSize` and calculated as follows.

```
uint256 updatedAvgPrice = _verifyAndUpdatePrice(
  userPosition.marketId, userPosition.isLong, false, OrderDS.OrderType.OPEN_NEW_POSITION,
userOrder.deltaSize
);
```

When the `updatedAvgPrice` is calculated, it will include the negative impact from the increase or decrease delta change. Therefore, the remaining position immediately has negative PnL as the `avgPrice` assigned is automatically worse than market price.

Recommendation

Calculate the average price so that the updated size is valued at the current price, as this was the price the PnL was settled at.

```
avgPrice = marketPrice * (sizeRemaining/totalNewSize) + updatedPrice *
(sizeDelta/totalNewSize)
```

Resolution

PariFi Team: Acknowledged as intended behavior for the pricing curve.

ORDM-10 | Liquidations Revert With 0 netPnl

Category	Severity	Location	Status
DoS	Medium	OrderManager.sol: 574	Resolved

Description

In the `_liquidatePosition` function, the `netPnl` is capped to the `userPosition.positionCollateral` amount before the `liquidationThreshold` validation.

Therefore if the user's position has 0 collateral, the `netPnl` will be assigned to 0 and subsequently fail the validation on line 574 as the `liquidationThreshold` is also 0.

There is no straightforward path to getting a position with 0 collateral, however the liquidation logic should be refactored as certainly any position with 0 collateral must be liquidated.

Additionally, in the case that the `netPnl` is capped to 0, the `safeTransfer` and fee distribution on lines 593 and 594 should not occur as certain tokens may revert on 0 transfers.

Recommendation

Cap the `netPnl` to the `userPosition.positionCollateral` after the `liquidationThreshold` is validated. Additionally, do not execute the fee distribution logic if the `netPnl` is 0.

Resolution

PariFi Team: Resolved in commit 3b5f6b.

DATA-3 | Changing Market Settings Applies Fees Retroactively

Category	Severity	Location	Status
Logical Error	Medium	DataFabric.sol: 482-488, 521-533	Resolved

Description

Changing market settings, by calling the `setMaximumOi` or `setBorrowingCurveConfig` functions from the `DataFabric` contract, creates an issue regarding fee updates. Fees are updated and deduced whenever any protocol operation is settled with the current value applied for the entire time since the last fee update. Neither of the two functions call the `_updateCumulativeFees` function to update fees up to that point before influencing the value.

Consider a situation when there are no operations for 3 hours, in which, after 2 hours the maximum OI was decreased. The next operation will commit all fees during those 3 hours with the new OI taken into consideration for fee calculation. This results in a higher fee being paid by users since only 1 hrs of those 3 hrs was spent in the market with the new, higher fees.

Recommendation

- Consider allowing a grace period when configuring these market values so that users have a time limit to modify/cancel their current positions
- Make the `setMaximumOi` and `setBorrowingCurveConfig` functions callable only in a paused market. If the protocol cannot do this, call `_updateCumulativeFees` before setting new configuration value as to not impact fees since the last checkpoint. `_updateCumulativeFees` must not be called in a paused market as to not accumulate fees for users if paused.

Resolution

PariFi Team: Resolved in commit 62c8818.

DATA-4 | User Can Add Collateral When Market Is Set To closeOnly mode

Category	Severity	Location	Status
Logical Error	Medium	DataFabric.sol: 422	Resolved

Description

When a market is in `closeOnly` mode, users are able to add collateral to an existing position. When adding collateral to an existing position, the `_increasePosition` function is called, which, in turns, calls `DataFabric::updateMarketData`.

In the case of adding collateral, `userOrder.deltaSize` equals 0, so the `updateMarketData` function will return in the first check and avoid the `LibError.CloseOnlyMode()` revert.

This can result in a position being kept longer in a market than intended, by continuously adding collateral when needed rather than closing it out.

Recommendation

Add a check to make sure the market is not in `closeOnly` mode:

`if (size == 0 && !closeOnlyMode[marketId]) return;`

Otherwise, if this functionality is indeed to be supported, clearly document this behavior.

Resolution

PariFi Team: Resolved in commit 6b92dc.

FEED-1 | Price May Be 0

Category	Severity	Location	Status
Precision	Medium	PriceFeed.sol: 101	Resolved

Description

The price returned by Pyth is checked to be non-zero, however the `priceUsd` may become 0 after decimal adjustment:
`priceUsd = SafeCast.toUint256(pythPrice.price) / (10 ** adjustedExpo);`

For example, if the `pythPrice.price = 1` and the `pythPrice.expo = -9`, then `priceUsd = 1 / 10 = 0`

Key protocol actions such as liquidations will fail because `_verifyAndUpdatePrice` calculates the price deviation between primary and secondary price with `uint256 diffBps = (_getDiff(primaryPrice, secondaryPrice) * PRECISION_MULTIPLIER) / secondaryPrice;` and there will be division by zero.

Recommendation

Carefully select which assets are supported for trading, as assets with a low price and a large, negative exponent are susceptible to this issue. Furthermore, validate the price is non-zero after conversion to `FEED_DECIMALS`.

Resolution

PariFi Team: Resolved in commit 62c8818.

VAULT-2 | Vault Is Not ERC4626 Compliant

Category	Severity	Location	Status
Specification	Low	ParifiVault.sol	Resolved

Description

The vault does not conform to the ERC4626 standard which may break external integrations. Some examples of non-compliance include:

- `maxWithdraw` does not take into account whether the user is in their cooldown period and cannot withdraw. According to specification, `maxWithdraw` "MUST factor in both global and user-specific limits, like if withdrawals are entirely disabled (even temporarily) it MUST return 0."
- `previewRedeem` "MUST be inclusive of withdrawal fees. Integrators should be aware of the existence of withdrawal fees."
- `previewWithdraw` "MUST be inclusive of withdrawal fees. Integrators should be aware of the existence of withdrawal fees."

Recommendation

Consider adjusting the non-compliant functions to be in-line with ERC4626 standards.

Resolution

PariFi Team: Resolved in commit 6c07f1a.

ORDM-11 | Position Can Be Liquidated on Creation

Category	Severity	Location	Status
Logical Error	Low	OrderManager.sol: 351	Acknowledged

Description

In the `_getNetProfitOrLossIncludingFees` function, a position's `isProfit` status is determined by both the fees incurred and the position's `avgPrice` compared to the current price of the asset. For liquidations these fees are `liquidationFee` and `closingFee`.

It is possible for a user to create a position that is immediately under water because fees are not considered when creating a position.

For example, if a user were to open a 100x position by supplying 100 USDC as collateral the user would pass the check in the `validateLeverage` function. However, with a `liquidationFee` of 1% and a `closingFee` of 0.1% the user would be immediately under water.

1% of 10000 = 100
0.1% of 10000 = 10
Total Fees: 110
Collateral: 100
The fees alone outweigh the users collateral leading to a complete liquidation.

Recommendation

Upon creating a position, calculate the net PnL with the `liquidationFee` and `closingFee` and ensure the liquidation threshold is not passed.

Resolution

PariFi Team: Acknowledged.

ORDM-12 | Stale Market Fees Used

Category	Severity	Location	Status
Logical Error	Low	OrderManager.sol: 387-389, 417-419, 450-452, 554-556, 815-818	Acknowledged

Description

During a paused market period, the protocol may choose to change opening, closing or liquidation fees. If this happens, any pending order, when settled, will use the new fee instead of the one that was at the time the order was created.

This affects:

- Canceling or settling a modify increase order
- Decreasing or closing an existing position
- Liquidating a position

Creating a pending order is the only operation that extracts fees exactly when it is executed.

Depending on the increase or decrease in fees, several unwanted scenarios may appear. Example, during a pause the opening fees were reduced and closing fees were increased. Afterwards:

- Any cancelled increase order will pay less fees then it was expecting and had agreed to by taking the initial trade, resulting in protocol funds losses
- Any settling of closing or decreasing orders, or liquidating a user will result more fees then user initial took into consideration when making his trade. Possibly the user would have not taken the trade with the new fee system

Recommendation

When creating pending orders also save a snapshot of the current market fees and use those when settling them.

Resolution

PariFi Team: Acknowledged.

FM-1 | Inaccurate Comment On Fee Distribution

Category	Severity	Location	Status
Documentation	Low	FeeManager.sol: 86	Resolved

Description

Function `distributeFees` implements a delay between fee distributions, where the delay is arbitrarily set by an admin.

However, the comment states that the function aims to `// Distribute fees at regular intervals of every 1 hour`.

Recommendation

Modify the comment to "Distribute fees at regular intervals of every delay period".

Resolution

PariFi Team: Resolved in commit 62c8818.

ORDM-13 | Fees Are Charged Even When Orders Are Cancelled

Category	Severity	Location	Status
Documentation	Low	OrderManager.sol: 800	Acknowledged

Description

In the `cancelPendingOrder` function, a user can cancel their order and will get the collateral they deposited back. However, the fees are already transferred to the `feeManager` so the user will not get those funds returned to them. This could be an issue when users are not aware of how fees are charged.

Recommendation

Clearly document that fees are always charged regardless if the order is settled or canceled.

Resolution

PariFi Team: Acknowledged.

ORDM-14 | _validateMarket Redundantly Called

Category	Severity	Location	Status
Optimization	Low	OrderManager.sol: 359,413	Resolved

Description

Both `_createNewPosition` and `_increasePosition` functions from the `OrderManager` contract validate markets by calling the `_validateMarket` function. This is redundant, since the two functions are only reached via `_settleOrder` which already validates the market in the same manner.

Recommendation

Remove the redundant call to the `_validateMarket` function from within the `_increasePosition` and `_createNewPosition` functions.

Resolution

PariFi Team: Resolved with commit 352ff02.

ORDM-15 | User Position Stuck When Blacklisted

Category	Severity	Location	Status
Logical Error	Low	OrderManager.sol: 400, 597, 683, 761, 833	Acknowledged

Description

If a user gets blacklisted by the `depositToken`, such as USDT or USDC, they will be unable to modify, close, or cancel their pending order, as the `safeTransfer/safeTransferFrom` method will revert. Also, the user position cannot be liquidated if `remainingCollateral !=0`.

Recommendation

Consider letting the user change their `userAddress` for a certain position.

Resolution

PariFi Team: Acknowledged.

DATA-5 | Liquidation Threshold at Max Will Put Protocol at Loss

Category	Severity	Location	Status
Logical Error	Low	DataFabric.sol: 580	Resolved

Description

In the DataFabric contract, the liquidationThreshold has a maximum of 100%, meaning that a position cannot be liquidated until they are at a loss of greater than 100%. Consequently, the only time that liquidations will occur is when the protocol is at a loss, leading to a loss of yield for the protocol and the LP's.

Recommendation

Ideally, check that the liquidationThreshold for any given market and ensure that it is less than PRECISION_MULTIPLIER. This can be done by changing:

```
if (_newMarket.liquidationThreshold < 5_000 || _newMarket.liquidationThreshold >
PRECISION_MULTIPLIER) {
```

to

```
if (_newMarket.liquidationThreshold < 5_000 || _newMarket.liquidationThreshold >=
PRECISION_MULTIPLIER) {
```

Resolution

PariFi Team: Resolved in commit 62c8818.

DATA-6 | getExpectedUtilization Lacks OI Validation

Category	Severity	Location	Status
Validation	Low	DataFabric.sol: 245-270	Resolved

Description

In the `getExpectedUtilization` function in the `DataFabric` contract there is no validation that the size increase would remain under the maximum allowed OI.

The maximum OI validation occurs later on in the order execution, in the `updateMarketData` function, however adding the check in the `getExpectedUtilization` function would terminate execution earlier and save gas expenditure.

Additionally, any integrating system relying on the `getExpectedUtilization` function would not receive an invalid response when the OI exceeds the allowed maximum.

Recommendation

Consider implementing validation such that the maximum OI is validated in the `getExpectedUtilization` function.

Resolution

PariFi Team: Resolved in commit e186c8.

RB-1 | Initial Multisig Address With DEFAULT_ADMIN_ROLE

Category	Severity	Location	Status
Logical Error	● Low	RBAC.sol: 45-46	Resolved

Description

The `DEFAULT_ADMIN_ROLE` role can change any other roles by default and is a security risk the team explicitly stated they do not want in their roles. This role is however granted to the initial multisig and a TODO mentioning it to be removed was forgotten in the code.

Recommendation

Remove lines 45-46 from the `RBAC.sol` file.

Resolution

PariFi Team: Resolved in commit 62c8818.

VAULT-3 | ParifiVault.cooldown missing whenNotPaused modifier

Category	Severity	Location	Status
Logical Error	Low	ParifiVault.sol: 224	Resolved

Description

The `cooldown` function from the `ParifiVault` contract is a user facing function. In case of a vault pause users may still call this function and, when unpause happens, have a direct withdraw executed.

Recommendation

Add the `whenNotPaused` modifier to the `cooldown` function.

Resolution

PariFi Team: Resolved in commit e186c8.

ORDM-16 | Execution Ordering Is Not Guaranteed During Settlement

Category	Severity	Location	Status
Validation	Low	OrderManager.sol: 481-483	Acknowledged

Description

There is no validation that pending orders settled by keepers on behalf of users are executed in the correct order, meaning in the order they were created by the user.

Consider the following scenario:

- User has a position that is close to being liquidated so he creates an order to add collateral
- Then user realizes he added too much collateral and send a new order to slightly reduce the collateral
- A keeper may, by mistake, execute the second order before the first, that would make the position liquidatable

Recommendation

Settling an order should have a mechanism to ensure that initial user order creation is respected. If the keeper role will be decentralized in the future, this is an issue that must be fixed before that point.

Resolution

PariFi Team: Acknowledged.

RB-2 | Not Following A 2 step ADMIN Role Transfer

Category	Severity	Location	Status
Optimization	Low	RBAC.sol: 90	Resolved

Description

In a standard 2-step role transfer, the current holder initiates the pending transfer and the new holder must accept it. As it is implemented in the `RBAC` contract, the old `ADMIN` role holder initiates the transfer by calling the `proposeNewMultisig` function and again the old holder then commits the change by calling the `updateMultisig` function.

Recommendation

Change so that the new `ADMIN` role holder must call the `updateMultisig` function.

Resolution

PariFi Team: Resolved in commit 6b92dc9.

FM-2 | distributionFee DoS

Category	Severity	Location	Status
DoS	Low	FeeManager.sol: 99	Resolved

Description

Tokens that revert on 0 transfers could cause a DoS in the `distributeFees` function if the `lpAmount` rounds to 0 or if the `protocolFeeAmount` is ever 0.

Recommendation

Consider only making the transfers if the amount to transfer is nonzero so that these will not fail.

Resolution

PariFi Team: Resolved in commit 6b92dc9.

ORDM-17 | Lacking Referrer Incentive

Category	Severity	Location	Status
Incentives	Low	OrderManager.sol: 668	Acknowledged

Description

Usually, there is an incentive to have a partner/referral address. However, the person getting referred here has no incentive, as the `feeInCollateral` experienced by the `msg.sender` is not reduced.

Recommendation

Consider implementing an incentive for the user to use a partner.

Resolution

PariFi Team: Acknowledged.

ORDM-18 | Inefficient Price Computation

Category	Severity	Location	Status
Optimization	Low	OrderManager.sol: 144	Resolved

Description

In the `_getPriceWithDeviation` function, the `increasedPrice` and `reducedPrice` are always computed, however only one of these is ever used depending on `isLong`.

Recommendation

Compute the `increasePrice` if `isLong == true` and the `reducedPrice` if `isLong == false` to save on gas.

Resolution

PariFi Team: Resolved in commit 352ff02.

ORDM-19 | safeTransferFrom Should Occur Before Updates

Category	Severity	Location	Status
Reentrancy	Low	OrderManager.sol: 683	Resolved

Description

In the `createNewPosition` function, `safeTransferFrom` occurs at the end of the function, however, this is potentially handing over tx execution to an untrusted address when the system is in an invalid state.

The invalid state is the order having been saved to storage but the collateral not actually collected into the contract. The risk is low as this requires tokens with callbacks that execute before the transfer is made, however should be addressed if tokens with callbacks are to be supported.

Recommendation

Collect tokens at the beginning of the `createNewPosition` function.

Resolution

PariFi Team: Resolved in commit 45d7f88.

ORDM-20 | orderToPositionId Not Cleared On Settlement

Category	Severity	Location	Status
Logical Error	Low	OrderManager.sol: 537	Resolved

Description

After an order is settled, it still exists in the `orderToPositionId` mapping although it has been deleted from the `pendingOrders` mapping. This contrasts with the functionality in function `cancelPendingOrder()` where the order is deleted from both mappings.

Recommendation

Perform `delete orderToPositionId[_orderId];` at the end of settlement.

Resolution

PariFi Team: Resolved in commit 352ff02.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits