GUARDIAN

Foil

Stable Gas Pricing Updates

Security Assessment

January 13th, 2025

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Nicholas Chew, Zdravko Hristov,
Osman Ozdemir, Mark Jonathas, Michael Lett

Client Firm Foil

Final Report Date January 13, 2025

Audit Summary

Foil engaged Guardian to review the security of its updates to the virtual gas marketplace. From the 21st of October to the 4th of November, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 6 High/Critical issues were uncovered and promptly remediated by the Foil team. Several issues impacted the fundamental behavior of the protocol, following their remediation Guardian believes the protocol to uphold the functionality described for Foil.

Security Recommendation Given the number of High and Critical issues detected, Guardian supports a secondary security review of the protocol at a finalized frozen commit.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Ethereum**

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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/foil-fuzzing

Table of Contents

Project Overview

Project Summary

Project Name	Foil
Language	Solidity
Codebase	https://github.com/foilxyz/foil
Commit(s)	Initial commit: 50373325e4ad7bb98382b5b4adce241a1ac1e770 Final commit: 5b3416a28dfaa24ba3844e10081e55425d0a286a

Audit Summary

Delivery Date	January 13, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	2	0	0	0	0	2
● High	4	0	0	0	0	4
● Medium	7	0	0	4	0	3
● Low	23	0	0	6	1	16

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	● Critical	● High	● Medium
Likelihood: *Medium*	● High	● Medium	● Low
Likelihood: *Low*	● Medium	● Low	● Low

Impact

High	Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
Medium	A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
Low	Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High	The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
Medium	An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
Low	Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

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.

Invariants Assessed

During Guardian's review of Foil, fuzz-testing with Echidna was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 5,000,000+ runs with a prepared Echidna fuzzing suite.

ID	Description	Passed	Remediation	Run Count
GLOBAL-01	The price of vGAS should always be in range of the configured min/max ticks.	✅	✅	5M+
GLOBAL-02	There should never be any liquidity outside of the [min, max] range of an epoch.	✅	✅	5M+
GLOBAL-03	The amount of vETH in the system, position manager & swap router should equal the max supply	✅	✅	5M+
GLOBAL-04	The amount of vGAS in the system, position manager & swap router should equal the max supply.	✅	✅	5M+
TRADE-01	The debt of a position should never be > the collateral of the position.	✅	✅	5M+
TRADE-02	Long positions have their debt in vETH and own vGAS	❌	❌	5M+
TRADE-03	Short positions have their debt in vGAS and own vETH.	❌	❌	5M+
TRADE-04	Trader should never have both borrowedVGas and borrowedVEth be non-zero.	✅	✅	5M+
TRADE-05	Trader's pending loss in ETH-worth should never exceed collateral put down (should never be in negative equity)	❌	❌	5M+

Invariants Assessed

ID	Description	Passed	Remediation	Run Count
TRADE-06	after creating/modifying trade position, the depositedCollateralAmount > debtValue - tokensValue	✅	✅	5M+
TRADE-07	After creating a trade position deposited collateral should be non-zero	✅	✅	5M+
TRADE-08	After user closes a trade position, no vGAS, vETH, borrowed vGAS, borrowed vETH	✅	✅	5M+
TRADE-09	After creating a trader position, positionSize is non-zero.	✅	✅	5M+
TRADE-10	createTradePosition should create a unique positionId	✅	✅	5M+
LIQUID-01	The debt of a position should not be > the collateral of the position.	✅	✅	5M+
LIQUID-02	A open LP position should not own any vETH or vGAS.	✅	✅	5M+
LIQUID-03	After all LP positions have been closed, for the remaining trader positions: net shorts == net longs.	✅	✅	5M+
LIQUID-04	Position.depositedCollateralAmount should be at least the required collateral for their position if their position turned into a Trade type.	✅	❌	5M+
LIQUID-05	QuoteLiquidityPositionTokens should match how many tokens are borrowed and how much liquidity is added after creating an LP position with createLiquidityPosition	✅	✅	5M+
LIQUID-06	After creating an LP position, liquidity in the Uni pool increases	✅	✅	5M+

Invariants Assessed

ID	Description	Passed	Remediation	Run Count
LIQUID-07	After increasing an LP position, liquidity in the Uni pool increases	✅	✅	5M+
LIQUID-08	After decrease an LP position, liquidity in the Uni pool decreases	❌	✅	5M+
LIQUID-09	After partial decrease an LP Position, should not get InsufficientColateral revert (unexpected in this case)	❌	❌	5M+
LIQUID-10	createLiquidityPosition should create a unique positionId	✅	✅	5M+
SETTLE-01	It should always be possible to settle all positions after the epoch is settled.	❌	✅	5M+
SETTLE-02	After settlement with settlePosition, position should not have any borrowedvETH nor borrowedVGAS, and no vGAS nor vETH (cleared out position)	✅	✅	5M+
SETTLE-03	Settlement should not revert with ERC20InsufficientBalance.	❌	❌	5M+
SETTLE-04	Settlement should not panic underflow	❌	✅	5M+
EPOCH-01	Position with non zero loan amount for lp should always have non-zero collateral required.	✅	✅	5M+
VLT-01	Vault functions should never revert with ERC20InsufficientBalance error	-	✅	5M+
VLT-02	totalPendingDeposits should be sum of deposit requests - withdrawRequestDeposit(s)	-	❌	5M+

Invariants Assessed

ID	Description	Passed	Remediation	Run Count
VLT-03	totalPendingWithdrawals should be sum of requestRedeem(s) - withdrawRequestRedeem(s)	-	❌	5M+
VLT-04	pendingSharesToBurn should always be less than or equal to total supply of shares	-	✅	5M+
VLT-05	Pending transaction requested epoch should never be greater than current epoch	-	✅	5M+
VLT-06	Vault should not Panic	-	❌	5M+
VLT-07	mint/deposit should decrease balance of shares in the Vault contract, total supply should stay the same	-	✅	5M+
VLT-08	redeem/withdraw should decrease total supply	-	✅	5M+

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Collateral Removed On Position Adjustment	Logical Error	● Critical	Resolved
C-02	tradeRatio Can Be Manipulated To Wipe Debt	Logical Error	● Critical	Resolved
H-01	Last User Of Epoch Cannot Withdraw Collateral	Logical Error	● High	Resolved
H-02	Collateral Returned Despite Bad Debt	Logical Error	● High	Resolved
H-03	Insolvency Because Of tradeRatio Rounding	Rounding	● High	Resolved
H-04	Settlement Failure Due To Underflow	DOS	● High	Resolved
M-01	Fee Collector Can Horde Fees	Logical Error	● Medium	Acknowledged
M-02	_checkOnERC721Received Bool Is Not Checked	Validation	● Medium	Resolved
M-03	Incorrect deltaCollateral Check When Negative	Validation	● Medium	Resolved
M-04	Rightful Disputer Might Lose Bonds	Validation	● Medium	Resolved
M-05	Decreasing LP May Require Collateral	DOS	● Medium	Acknowledged
M-06	Using LP For More Efficient Trades	Logical Error	● Medium	Acknowledged
M-07	Dangerous Price Used For Resolution Callback	Logical Error	● Medium	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-01	Frontrunning Pool Creation	DOS	● Low	Partially Resolved
L-02	Overflow In DecimalPrice Library	Arithmetic Error	● Low	Resolved
L-03	Unused Function	Unused code	● Low	Resolved
L-04	Redundant Function Call	Informational	● Low	Resolved
L-05	FeeCollectorNft Is Transferable	Informational	● Low	Acknowledged
L-06	Missing unchecked In Uniswap Libraries	Arithmetic Error	● Low	Resolved
L-07	Disputer Never Updated	Informational	● Low	Resolved
L-08	Inaccurate Custom Error	Informational	● Low	Resolved
L-09	Authorized Addresses Can't Modify Positions	Informational	● Low	Resolved
L-10	QuoterV2 Should Not Be Called On-Chain	Informational	● Low	Acknowledged
L-11	Typo	Typo	● Low	Resolved
L-12	Insufficient startingSqrtPriceX96 Validation	Validation	● Low	Acknowledged
L-13	View Function Should Account For Loss	Logical Error	● Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-14	Unnecessary Casting	Informational	● Low	Resolved
L-15	Unexpected Revert With Small Amounts	DOS	● Low	Resolved
L-16	MarketNotInitialized Is Never Thrown	Informational	● Low	Resolved
L-17	Fee Collectors Can Block Initialization	Warning	● Low	Acknowledged
L-18	bondAmount Is Not Sufficiently Validated	Warning	● Low	Resolved
L-19	Uniswap tickSpacing May Be Changed	Warning	● Low	Acknowledged
L-20	Traders Unable To Close Profitable Position	Logical Error	● Low	Acknowledged
L-21	tokenById Revert Reason	Error string	● Low	Resolved
L-22	Comment Typo	Logical Error	● Low	Resolved
L-23	Epoch End Time Off-by-One	Typo	● Low	Resolved

C-01 | Collateral Removed On Position Adjustment

Category	Severity	Location	Status
Logical Error	● Critical	Position.sol	Resolved

Description PoC

Fee collectors can create under-collateralized positions and collateralize them using depositCollateral.

However, when calling increaseLiquidityPosition or decreaseLiquidityPosition, the zero collateral requirement for fee collectors causes updateCollateral to mistakenly remove and transfer all collateral back to the fee collector.

This allows fee collectors to withdraw collateral after depositing, potentially avoiding any loss at the end of the epoch. This is against protocol spec that the fee collector should never be able to back out of provided collateral, even if adjusting positions.

Recommendation

updateCollateral should not be triggered for fee collectors when modifying a position or position modification should be restricted during the epoch.

Resolution

Foil Team: The issue was resolved in PR#155.

C-02 | tradeRatio Can Be Manipulated To Wipe Debt

Category	Severity	Location	Status
Logical Error	● Critical	TradeModule.sol	Resolved

Description PoC

When modifying a trade position, the output of a swap is used to calculate tradeRatio, which serves as a proxy price to determine the value of vGas. This ratio is essential for calculating PnL and setting the borrowed amounts for the new position.

However, if a small (dust) amount of vGas is swapped, amountIn or amountOut for vETH may round to zero due to Uniswap's rounding behavior, causing tradeRatio to also be zero. This allows for potential exploitation: in a long position, borrowedVEth becomes zero, effectively wiping the position's debt and creating bad debt in the system.

Attack Scenario:
1. Alice opens a long position.
2. Alice decreases the position by 1 wei, setting tradeRatio to zero, which is below minPrice, creating bad debt by wiping all borrowedVEth.
3. Alice closes the position, recovering all previously deposited collateral plus additional funds, effectively stealing from the system.

Recommendation

If the trade price is below or above the min or max price for a pool, then revert.

Resolution

Foil Team: The issue was resolved in PR#161.

H-01 | Last User Of Epoch Cannot Withdraw Collateral

Category	Severity	Location	Status
Logical Error	● High	LiquidityModule.sol	Resolved

Description PoC

In LiquidityModule._closeLiquidityPosition, collected amounts are rounded up by adding 1 wei to offset Uniswap's rounding when opening a position.

However, borrowed amounts may be zero (e.g., when adding liquidity outside the current price tick), and collected amounts can also be zero, depending on the price tick.

By adding 1 wei, users may withdraw more collateral than they initially deposited. Over time, this leads to the last user in an epoch being unable to withdraw due to insufficient collateral.

This behavior can also be exploited by malicious users with the following steps:
• Provide liquidity above the current price tick, so only vGas is borrowed and no vETH.
• Immediately decrease liquidity, collecting all borrowed vGas plus 1 wei of vETH.
• The 1 wei of vETH is added to the user's deposited collateral and then withdrawn.

Recommendation

1. If collected amount is zero, do not add the 1 wei adjustment.
2. Modify settlePosition to allow payouts of the contract's remaining balance when the exact collateral amount is insufficient, preventing the last withdrawal from reverting if the balance is short by a few wei.

Resolution

Foil Team: The issue was resolved in PR#150.

H-02 | Collateral Returned Despite Bad Debt

Category	Severity	Location	Status
Logical Error	● High	TradeModule.sol	Resolved

Description PoC

When a trader is closing out a position, if the loss exceeds collateral deposited then this case is entered. As bad debt has been incurred, the collateral should be reduced to zero but currently depositedCollateralAmount remains unchanged.

The extraCollateralRequired would cover the losses, but however it is only taken into account if the trader is re-opening a new position.

So, If the trader was closing the position (i.e. size = 0), then all deposited collateral is returned to the trader implying losses are borne by the protocol/other LPs and traders.

Recommendation

Change the logic to:

```
if (collateralLoss > params.oldPosition.depositedCollateralAmount)
      output.position.depositedCollateralAmount = 0;
      extraCollateralRequired = collateralLoss - params.oldPosition.depositedCollateralAmount;
```

Resolution

Foil Team: The issue was resolved in PR#164.

H-03 | Insolvency Because Of tradeRatio Rounding

Category	Severity	Location	Status
Rounding	● High	TradeModule.sol	Resolved

Description PoC

When _quoteOrTrade is called and PnL is calculated, the tradeRatio experiences precision loss because of rounding down when performing divDecimal. While this is fine for longs, it's not for shorts.

That's because the tradeRatio is a fill price and if the fill price is lower, shorts will have made a profit. In result, when the PnL for shorts is calculated the trader will experience a smaller loss, leaving the system with fewer funds available than it should have in order to operate.

This can be most visible if a position has only borrowedVGas (short) and makes a trade to close the position. Because the entirety of the debt is being paid off, it would be expected that the vEthToZero would at least match the runtime.tradedVEth.

However, the vEthToZero would be slightly less due to the tradeRatioD18 rounding, and less collateral being held in the Foil contract. Later, when LPs try to close or settle their position, they will not be able to do so.

The contract will try to send them the amount they have earned, but this amount is not fully backed by the losses of the traders and the transaction will revert with ERC20: transfer amount exceeds balance.

Recommendation

In case of a short position, round the tradeRatio up to provide a worse fill price when going towards the long direction.

Resolution

Foil Team: The issue was resolved in PR#168.

H-04 | Settlement Failure Due To Underflow

Category	Severity	Location	Status
DOS	● High	Position.sol: 286	Resolved

Description PoC

When settling a liquidity position, getCurrentPositionTokenAmounts is called to retrieve the corresponding vGas and vETH token amounts of the position, which are then later rebalanced during position.settle.

The rebalancing process converts everything to ETH, adding all value to depositedCollateral and subtracting all debt from depositedCollateral.

However, the calculation during getCurrentPositionTokenAmounts rounds down, which can cause the total value of the position (including the collateral) to be less than the total debt in some cases.

This results in the settlement reverting due to an underflow in the following line: self.depositedCollateralAmount = self.borrowedVEth.

Recommendation

Rounding should be accounted for when calculating the required collateral. Consider adjusting loanAmount0 and loanAmount1 up by 1 wei during calculation.

Resolution

Foil Team: The issue was resolved in PR#174.

M-01 | Fee Collector Can Horde Fees

Category	Severity	Location	Status
Logical Error	● Medium	LiquidityModule.sol	Acknowledged

Description

Fee collectors can create under-collateralized positions and collateralize them using `depositCollateral`.

Currently, there are no restrictions preventing a fee collector from creating an oversized liquidity position, which can monopolize all available liquidity and hoard fees, preventing other fee collectors from benefiting.

Recommendation

Impose limits on the size of liquidity positions that fee collectors can create to ensure fair distribution of fees.

Resolution

Foil Team: Acknowledged.

M-02 | _checkOnERC721Received Bool Is Not Checked

Category	Severity	Location	Status
Validation	● Medium	LiquidityModule.sol: 37, TradeModule.sol: 52	Resolved

Description

While creating a position in the liquidity or trading modules, `_checkOnERC721Received` function is called and then the position NFT is minted.

However, `_checkOnERC721Received` does not revert on failure but only returns false. Return value is not checked and positions can be minted to contracts that can't hold NFTs

Recommendation

Check the return value of the function before continuing.

Resolution

Foil Team: The issue was resolved in PR#149.

M-03 | Incorrect deltaCollateral Check When Negative

Category	Severity	Location	Status
Validation	● Medium	TradeModule.sol: 348	Resolved

Description

Users provide `deltaCollateralLimit` when modifying their trade positions. While a positive `deltaCollateralLimit` indicates the maximum amount a user wants to provide to the protocol, a negative `deltaCollateralLimit` represents the minimum collateral amount a user wishes to receive from the protocol when decreasing or closing a position.

However, the negative case in `_checkDeltaCollateralLimit` is incorrect and behaves oppositely. It reverts when `deltaCollateralLimit < 0 && deltaCollateral < deltaCollateralLimit`. The user-provided value functions as a maximum limit instead of a minimum limit, resulting in the user receiving less than intended all the time.

Recommendation

Change `deltaCollateral < deltaCollateralLimit` to `deltaCollateral > deltaCollateralLimit`.

Resolution

Foil Team: The issue was resolved in PR#159.

M-04 | Rightful Disputer Might Lose Bonds

Category	Severity	Location	Status
Validation	● Medium	UMASettlementModule.sol	Resolved

Description

Currently, there is no mechanism that checks whether there is already an ongoing dispute or not while submitting a price. Asserter can submit a new price after an initial incorrect submission without waiting a dispute to resolve in 48-96 hours.

This would cause disputer to lose their bonds since the settlement will fail at this line as the assertionIds won't match.

Recommendation

Do not allow submitting new price if there is already an ongoing dispute.

Resolution

Foil Team: The issue was resolved in PR#169.

M-05 | Decreasing LP May Require Collateral

Category	Severity	Location	Status
DOS	● Medium	Epoch.sol	Acknowledged

Description

Whenever a position is modified in position.updateValidLp, the required collateral for that position is calculated and compared against the current available collateral.

The collateral is calculated by using two values - debitEth and creditEth (debit is taken from the user and credit is given to them).

When removing a small amount of liquidity, it's possible that the decrease in creditEth is larger than the decrease in debitEth, which would lead to increased collateral requirements.

Since additionalCollateral is 0 when decreasing a position, the transaction will revert with InsufficientCollateral().

Recommendation

Allow the user to supply additional collateral when decreasing their position.

Resolution

Foil Team: Acknowledged.

M-06 | Using LP For More Efficient Trades

Category	Severity	Location	Status
Logical Error	● Medium	LiquidityModule.sol	Acknowledged

Description PoC

Instead of opening a long position in the TradeModule to gain exposure to vGas, traders can use the LiquidityModule for a more efficient strategy.

By adding liquidity below the current price with a lower tick set to their desired entry price, traders can effectively create a limit order.

When the price reaches this minimum tick, the LP position converts fully to vGas, which can then be closed and transitioned into a Trade position.

Since LP positions have reduced collateral requirements (no swap fees nor price impact on entry), this approach allows for the same vGas position with less collateral.

Recommendation

Consider if this behavior should be prevented from a protocol perspective. One possible solution would be to fully close an LP's position in _closeLiquidityPosition instead of the transition to a Trade position, although low liquidity environments would have to be taken into consideration, and slippage protection would have to be appropriately handled.

Resolution

Foil Team: Acknowledged.

M-07 | Dangerous Price Used For Resolution Callback

Category	Severity	Location	Status
Logical Error	● Medium	UMASettlementModule.sol	Acknowledged

Description

If settlement.settlementPriceD18 in assertionResolvedCallback() is outside the acceptable price range for the given epoch, the new price for the epoch will be capped to either min or max with function setSettlementPriceInRange.

However, resolutionCallback() is still called with the original settlement.settlementPriceD18 and not the newly set price of the epoch. Whenever the settlement price is outside the acceptable range, the callback will receive an incorrect price.

The protocol team plans to create new epochs with that price which will lead to an epoch starting with prices outside the valid range.

Recommendation

Pass epoch.settlementPriceD18 instead of settlement.settlementPriceD18 to assertionResolvedCallback().

Resolution

Foil Team: In Vault we use the resolution settlementPrice (not capped) to create the next epoch and compute new bounds.

L-01 | Frontrunning Pool Creation

Category	Severity	Location	Status
DOS	● Low	Epoch.sol	Partially Resolved

Description

When a new epoch is created by calling Epoch.createValid() two virtual tokens are deployed and used to create a new UniswapV3Pool.

The virtual tokens are deployed by the Epoch contract via the CREATE2 opcode. The owner of the Foil market will pass a salt parameter which will determine the address of the newly deployed tokens.

A malicious entity can frontrun the epoch creation transaction and use the salt passed in order to calculate the addresses of the two virtual tokens. These addresses can then be used to call UniswapV3Factory.createPool().

The pool for the two tokens will be created and when the Foil owner's transaction calls createPool(), it will revert because the pool already exists and the epoch won't be created. This frontrunning can be executed to stop any epoch creation.

Recommendation

Be sure to use a private network RPC when submitting the create transaction.

Resolution

Foil Team: Partially Resolved.

L-02 | Overflow In DecimalPrice Library

Category	Severity	Location	Status
Arithmetic Error	● Low	DecimalPrice.sol	Resolved

Description

The function sqrtRatioX96ToPrice is used to obtain price in several parts of the codebase. The issue lies with performing a square of two uint160 numbers which could overflow uint256. Overflow occurs when sqrtRatioX96 exceeds 2^128 - 1.

Recommendation

Perform >> 96 shift operation on sqrtRatioX96 first before doing square operation. Alternatively, use Uniswap's FullMath.mulDiv which handles the intermediate overflow case.

Resolution

Foil Team: The issue was resolved in PR#168.

L-03 | Unused Function

Category	Severity	Location	Status
Unused code	● Low	Position.sol	Resolved

Description

The `Position.getRequiredCollateral()` function is not used anywhere

Recommendation

Consider removing it if unnecessary

Resolution

Foil Team: The issue was resolved in PR#168.

L-04 | Redundant Function Call

Category	Severity	Location	Status
Informational	● Low	TradeModule.sol	Resolved

Description

In `quoteModifyTraderPosition`, `validateNotSettled` is called redundantly twice. Additionally, the `validateSettlementSanity` function in `Epoch.sol` is unused and can be removed.

Recommendation

Remove the redundant `validateNotSettled` call and delete the unused `validateSettlementSanity` function.

Resolution

Foil Team: The issue was resolved in PR#168.

L-05 | FeeCollectorNft Is Transferable

Category	Severity	Location	Status
Informational	● Low	FeeCollectorNft.sol	Acknowledged

Description

The FeeCollectorNft is used to assert a user is a fee collector. Fee collectors are given special privileges that allow them to take under collateralized loans.

The FeeCollectorNft is transferable, and a malicious fee collector could take advantage of this to sell their FeeCollectorNft to users so they can take under collateralized loans, and jeopardize the health of the protocol.

Recommendation

Do not allow fee collectors to transfer FeeCollectorNfts.

Resolution

Foil Team: Acknowledged.

L-06 | Missing unchecked In Uniswap Libraries

Category	Severity	Location	Status
Arithmetic Error	● Low	FullMath.sol, TickMath.sol	Resolved

Description

The `FullMath` and `TickMath` libraries were adapted from Uniswap, which relies on overflow wrapping behavior available only in Solidity versions <0.8. Foil's implementation targets Solidity versions >0.8.2, where unchecked arithmetic is not default.

Without wrapping these functions with `unchecked`, phantom overflows may occur, causing unexpected reverts when intermediate values exceed 256 bits.

For example, `mulDiv(type(uint).max`, `type(uint).max`, `type(uint).max)` would revert in Solidity >0.8 but return `type(uint).max` in older versions.

Recommendation

Wrap all relevant function bodies in unchecked to prevent phantom overflows. See the Uniswap v0.8 library implementation for reference:

https://github.com/Uniswap/v3-core/blob/0.8/contracts/libraries/FullMath.sol

Resolution

Foil Team: The issue was resolved in PR#168.

L-07 | Disputer Never Updated

Category	Severity	Location	Status
Informational	● Low	UMASettlementModule.sol: 113-125	Resolved

Description

When a settlement price is submitted, `disputer` is set as `address(0)`. However, the `disputer` is never updated in the even if a dispute happens and will remain as `address(0)`.

Recommendation

Consider removing `disputer` as it is never used in the codebase or update it by getting the address from the oracle contract.

Resolution

Foil Team: The issue was resolved in PR#169.

L-08 | Inaccurate Custom Error

Category	Severity	Location	Status
Informational	● Low	Epoch.sol	Resolved

Description

`Epoch.validateNotSettled()` will revert with `EpochNotSettled` if the epoch has expired and is not settled. This is slightly inaccurate because the main reason the revert happens is because the epoch has expired.

Recommendation

Consider changing the error to `EpochExpired`

Resolution

Foil Team: The issue was resolved in PR#168.

L-09 | Authorized Addresses Can't Modify Positions

Category	Severity	Location	Status
Informational	● Low	Global	Resolved

Description

Currently, only the position owners can modify liquidity or trade positions. However, since positions are NFTs, users can assign operators or approve other addresses to manage their NFTs. An operator, even if authorized, cannot modify users' positions.

Additionally, the error message during the ownership check is `NotAccountOwnerOrAuthorized`, which implies that authorized addresses should be able to modify positions.

Recommendation

Consider allowing authorized addresses to modify positions.

Resolution

Foil Team: The issue was resolved in PR#169.

L-10 | QuoterV2 Should Not Be Called On-Chain

Category	Severity	Location	Status
Informational	● Low	Trade.sol: 57 & 126	Acknowledged

Description

quoteCreateTraderPosition() and quoteModifyTraderPosition() are both functions that are meant to be used to quote prices, however neither of them are marked as view functions. They cannot be marked as view functions because they use IQuoterV2.

Uniswaps documentation on IQuoterV2 states, "These functions are not marked view because they rely on calling non-view functions and reverting to compute the result. They are also not gas efficient and should not be called on-chain."

This will lead to users having to pay gas costs if these functions are called.

Recommendation

Document to users that they should only call quoteCreateTraderPosition() and quoteModifyTraderPosition() off-chain.

Resolution

Foil Team: Acknowledged.

L-11 | Typo

Category	Severity	Location	Status
Typo	● Low	Epoch.sol: 231	Resolved

Description

The Natspec comment above the `Epoch.getCollateralRequirementsForTrade` function has a typo: "Gets the reuired collateral amount…"

Recommendation

Update the comment.

Resolution

Foil Team: The issue was resolved in PR#168.

L-12 | Insufficient startingSqrtPriceX96 Validation

Category	Severity	Location	Status
Validation	● Low	Epoch.sol	Acknowledged

Description

An Epoch is meant to have it's price bounded between its minPriceD18 and maxPriceD18. Upon creation, the owner passes a startingSqrtPriceX96 parameter to initialize the epoch's pool with. This price is not validated to be in the allowed range which allows a pool creation with invalid price.

Recommendation

Validate the startingSqrtPriceX96 variable.

Resolution

Foil Team: Later will be done by the vault, so will be secure.

L-13 | View Function Should Account For Loss

Category	Severity	Location	Status
Logical Error	● Low	ViewsModule.sol: 188	Resolved

Description

The function getPositionCollateralValue should return the current value of a position. However, in the current implementation it only accounts for gains and not losses, therefore returning an inaccurate value if it is a losing position.

Recommendation

Do not cap totalNetValue to a minimum of zero, and subtract any losses from depositedCollateral.

Resolution

Foil Team: The issue was resolved in PR#170.

L-14 | Unnecessary Casting

Category	Severity	Location	Status
Informational	● Low	Epoch.sol	Resolved

Description

When an epoch is created, epoch.pool is assigned the IUniswapV3Pool value of the newly deployed pool. After that, epoch.pool is again casted to IUniswapV3Pool, which is redundant since it's already a variable of that type.

Recommendation

You can use epoch.pool without casting.

Resolution

Foil Team: The issue was resolved in PR#168.

L-15 | Unexpected Revert With Small Amounts

Category	Severity	Location	Status
DOS	● Low	Trade.sol: 73	Resolved

Description

When modifying positions in the `TradeModule`, the `Trade.swapOrQuoteTokensExactIn` function is called, which subsequently calls the Uniswap swap router.

The swap router performs the swap within the Uniswap pool, and the pool then invokes the `uniswapV3SwapCallback` function of the router.

The `uniswapV3SwapCallback` function expects at least one of the delta amounts to be greater than zero. However, if the trade amounts are very small, the swap steps in the Uniswap pool can result in both delta amounts being zero, which causes `uniswapV3SwapCallback` to revert.

Therefore, it is possible for a user to create a small trade e.g. long 1 wei vGas, and then be unable to close it due to the swap amounts rounding down in Uniswap when calculating swap steps.

This will ultimately cause a revert within the `uniswapV3SwapCallback`: `require(amount0Delta > 0 || amount1Delta > 0);`. Consequently, a user has a position they are unable to close.

Recommendation

Consider implementing minimum trade sizes and/or documenting this behavior.

Resolution

Foil Team: The issue was resolved in PR#154.

L-16 | MarketNotInitialized Is Never Thrown

Category	Severity	Location	Status
Informational	● Low	ConfigurationModule.sol	Resolved

Description

The onlyOwner modifier in ConfigurationModule should revert with the MarketNotInitialized() error if the owner of the market is not set.

However, the onlyOwner modifier first checks if the msg.sender is the current market owner and if they are not, the transaction will revert with OnlyOwner() error.

In the case where the market is not initialized, i.e owner == address(0), the revert reason will always be OnlyOwner(), since nobody can send a call from address(0). In result, the MarketNotInitialized() error will never be used.

Recommendation

Switch the order of the two ifs.

Resolution

Foil Team: The issue was resolved in PR#168.

L-17 | Fee Collectors Can Block Initialization

Category	Severity	Location	Status
Warning	● Low	ConfigurationModule.sol	Acknowledged

Description

ConfigurationModule.initializeMarket() is used to create the market. It also mints the FeeCollector NFT to the fee collectors. If any of them don't support receiving NFTs or revert intentionally, the market won't be created.

Recommendation

Be aware of this situation. If this happens, you can call initializeMarket again, but this time without the specific fee collector.

Resolution

Foil Team: Acknowledged.

L-18 | bondAmount Is Not Sufficiently Validated

Category	Severity	Location	Status
Warning	● Low	Market.sol	Resolved

Description

`Epoch.validateEpochParams()` validates that the `bondAmount` should be a positive number. However, it should be at least as big as the result of the `getMinimumBond()` function of the UMA oracle. Otherwise, the assertions will not be accepted.

Recommendation

Make sure to pass a valid `bondAmount` when creating the market.

Resolution

Foil Team: The issue was resolved in PR#172.

L-19 | Uniswap tickSpacing May Be Changed

Category	Severity	Location	Status
Warning	● Low	Market.sol	Acknowledged

Description

`Market.getTickSpacingForFee()` returns the Uniswap tick spacing associated with the given fee tier. The tick spacings are hardcoded, but the Uniswap Factory has a function enableFeeAmount() which allows the owner to change the fee tiers. If this happens, the Foil contracts may use stale data.

Recommendation

Be aware of the risk.

Resolution

Foil Team: Acknowledged.

L-20 | Traders Unable To Close Profitable Position

Category	Severity	Location	Status
Logical Error	● Low	Global	Acknowledged

Description

Fee Collectors opened LP positions at the beginning of an epoch and deposit collateral after they've earned fees. This collateral could be streamed in periodically or provided in bulk at settlement.

Due to the under-collateralized LP positions, traders may find themselves unable to exit profitable positions until Fee Collectors deposit collateral. As Fee Collectors are expected to hold large LP positions, this may affect a large group of traders.

This leads to temporarily locked funds and potential loss of yield for traders who are unable to close a profitable position promptly.

Recommendation

Consider implementing a minimum deposit amount for fee collectors. Or else, document this risk for users.

Resolution

Foil Team: Acknowledged.

L-21 | tokenById Revert Reason

Category	Severity	Location	Status
Error string	● Low	ERC721EnumerableStorage.sol	Resolved

Description

ERC721EnumerableStorage.tokenByIndex() reverts with a custom error if a non existent token was passed. It does so when index > totalSupply(). Because the index of allTokens starts from 0, this statement will not catch all possible cases.

For example, when totalSupply is 1, there is only 1 token in the allTokens array, at index 0 and index 1 is empty. If we call tokenByIndex(1) it won't enter the if statement and will revert with index out of bounds error instead of the custom error.

Recommendation

Change the condition to index >= totalSupply()

Resolution

Foil Team: The issue was resolved in PR#169.

L-22 | Comment Typo

Category	Severity	Location	Status
Logical Error	● Low	TradeModule.sol	Resolved

Description

The comment `// net vEth from oritinal positon minus the vEth to zero` misspells “original”.

Recommendation

Correct the typo.

Resolution

Foil Team: The issue was resolved in PR#168.

L-23 | Epoch End Time Off-by-One

Category	Severity	Location	Status
Typo	● Low	TradeModule.sol	Resolved

Description

Epoch trade and liquidity activity is prevented once the `block.timestamp >= self.endTime` as can be seen in function `validateNotSettled`.

However, price submissions are restricted with `block.timestamp > epoch.endTime` within function `validateSubmission`.

When the `block.timestamp == epoch.endTime`, prices can be submitted since market activity is disallowed at that point, but submissions are restricted with current validation.

Recommendation

Adjust the validations appropriately within the `UMASettlementModule` or clearly document this behavior as this is extremely edgecase behavior.

Resolution

Foil Team: The issue was resolved in PR#169.

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