Report

v. 2.0

Customer

Term Structure Labs

Smart Contract Audit

TermMax. Phase III

1st October 2025

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	29.07.25	A. Zveryanskaya	Initial Draft
0.2	29.07.25	A. Zveryanskaya	Minor revision
1.0	29.07.25	A. Zveryanskaya	Release
1.1	01.10.25	A. Zveryanskaya	Reclassified: CVF-1, 2, 3, 4, 5
2.0	01.10.25	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

TermMax is a DeFi protocol that simplifies DeFi borrowing, lending, and leveraging with one-click token trading, enabling predictable borrowing costs and stable returns.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	ITermMaxMarketV2.sol	TermMaxMarketV2.sol	TermMaxOrderV2.sol
access/			
	AccessManagerV2.sol		
errors/			
	TermMaxTokenErrors.sol		
events/			
	GearingTokenEventsV2.sol	TermMaxTokenEvents.sol	
extensions/aave/			
	IAaveV3Minimal.sol		
factory/			
	TermMaxFactoryV2.sol	TermMaxPriceFeedFactoryV2.sol	
lib/			
	MarketConstantsV2.sol		
oracle/priceFeeds/			
	ITermMaxPriceFeed.sol	TermMaxERC4626PriceFeed.sol	TermMaxPriceFeedConverter.sol
	TermMaxPTPriceFeed.sol		
oracle/			
	IOracleV2.sol	OracleAggregatorV2.sol	

router/swapAdapters/			
	TermMax SwapAdapter.sol	TermMax TokenAdapter.sol	TermMax TokenAdapter.sol
router/			
	TermMaxRouterV2.sol		
tokenomics/			
	PreTMX.sol		
tokens/			
	AbstractGearing TokenV2.sol	GearingToken WithERC20V2.sol	IGearingTokenV2.sol
	IMintableERC20V2.sol	ITermMaxToken.sol	MintableERC20V2.sol
	StakingBuffer.sol	TermMaxToken.sol	

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

We found 3 major, and a few less important issues. All identified Major issues have been fixed.

Fixed 3 out of 3 issues

6 Major Issues

CVF-7 FIXED

- **Category** Unclear behavior
- **Source** TermMaxRouterV2.sol

Description The returned values are ignored.

Recommendation Explicitly require true to be returned or use the “safeTransferFrom” function.

```
192 ft.transferFrom(msg.sender, address(this), ftInAmt);
    xt.transferFrom(msg.sender, address(this), xtInAmt);
```

CVF-8 FIXED

- **Category** Unclear behavior
- **Source** TermMaxRouterV2.sol

Description There are no msg.sender checks in these functions to ensure they are called as callbacks by the contracts the function are supposed to be called by.

Recommendation Add appropriate msg.sender checks.

```
599 /// @dev Market flash leverage flashloan callback
600 function executeOperation(address, IERC20, uint256 amount, bytes
      ↪ memory data)
```

```
632 /// @dev Gt flash repay flashloan callback
    function executeOperation(
```

CVF-10 FIXED

- **Category** Flaw
- **Source** OracleAggregatorV2.sol

Description A price returned by an oracle could be silently amended here, which could be very dangerous.

Recommendation Revert in case an oracle price is outside the allowed range.

```
227 if (oracle.maxPrice != 0 && price > oracle.maxPrice) {
        return oracle.maxPrice;
    }
230 if (oracle.minPrice != 0 && price < oracle.minPrice) {
        return oracle.minPrice;
    }
```

7 Moderate Issues

CVF-1 FIXED

- **Category** Unclear behavior
- **Source** AbstractGearingTokenV2.sol

Description It is unclear why loan is updated and collateral is removed only for a partial repayment, but not for a complete one.

Recommendation Update the the loan and transfer collateral on any repayment, or clearly explain, why full repayment doesn't need updating the loan and transferring collateral.

Client Comment *Thanks for your suggestion,I add the comments to clear the collateral is removed in _repay function if repay full debts.*

```
301 // Check ltv after partial repayment
    if (!repayAll) {
```

```
307     loanMapping[id] = loan;
        // Transfer collateral to the owner
        _transferCollateral(msg.sender, removedCollateral);
```

CVF-2 FIXED

- **Category** Suboptimal
- **Source** GearingTokenWithERC20V2.sol

Description This logic is overcomplicated and inefficient. It could be significantly simplified, taking into account that all denominators are powers of 10.

187
```
uint256 ddPriceToCdPrice = (
    valueAndPrice.debtPrice * cPriceDenominator * cPriceDenominator
       ↪ * 10 + collateralPrice - 1
) / collateralPrice;
```

193
```
uint256 cEqualRepayAmt = (repayAmt * ddPriceToCdPrice *
   ↪ cTokenDenominator)
    / (valueAndPrice.debtDenominator * cPriceDenominator *
       ↪ valueAndPrice.priceDenominator * 10);
```

Recommendation Simplify the logic like this:

```
uint256 repayValue = repayAmt * valueAndPrice.debtPrice;
uint256 scaleUp = cTokenDenominator * cPriceDenominator;
uint256 scaleDown = valueAndPrice.debtDenominator * valueAndPrice.
   ↪ priceDenominator;
uint256 cEqualRepayAmt;
if (scaleUp >= scaleDown)
  cEqualRepayAmt = repayValue * (scaleUp / scaleDown) /
     ↪ collateralPrice;
else
  cEqualRepayAmt = repayValue / (collateralPrice * (scaleDown /
     ↪ scaleUp))
```

CVF-3 FIXED

- **Category** Readability
- **Source** GearingTokenWithERC20V2.sol

Description Using different denominators for different prices and token amounts makes logic overcomplicated, inefficient, and prone to phantom overflows.

Recommendation Treat all token amounts as integers. Use the same constant denominator for all prices.

```
187 uint256 ddPriceToCdPrice = (
        valueAndPrice.debtPrice * cPriceDenominator * cPriceDenominator
            ↪ * 10 + collateralPrice - 1
    ) / collateralPrice;
```

```
193 uint256 cEqualRepayAmt = (repayAmt * ddPriceToCdPrice *
        ↪ cTokenDenominator)
        / (valueAndPrice.debtDenominator * cPriceDenominator *
            ↪ valueAndPrice.priceDenominator * 10);
```

CVF-4 FIXED

- **Category** Unclear behavior
- **Source** GearingTokenWithERC20V2.sol

Description The “remainningC” value is set to zero, but later overwritten with another value. This looks very suspicious.

Recommendation Refactor the code to assign each returned value exactly once.

```
212     remainningC = _encodeAmount(0);
```

```
218     remainningC = _encodeAmount(0);
```

```
226 remainningC = _encodeAmount(collateralAmt - removedCollateralAmt);
```

CVF-5 FIXED

- **Category** Unclear behavior
- **Source** PreTMX.sol

Description Burning tokens is equivalent to sending them to a dead address, so burning shouldn't be allowed in case transfer from the tokens owner are restricted.

Recommendation Don't allow burning tokens from addresses, which transfers are restricted from.

Client Comment *We limit only the owner can burn tokens.*

```
57 function burn(uint256 amount) external {
```

CVF-11 FIXED

- **Category** Unclear behavior
- **Source** AbstractGearingTokenV2.sol

Description The configuration is not validated.

Recommendation Implement proper configuration checks.

```
93 _config = config_;
```

CVF-12 INFO

- **Category** Unclear behavior
- **Source** AbstractGearingTokenV2.sol

Description There is no range check for the "debtAmt" value.

Recommendation Implement a check to ensure "debtAmt" is not zero.

Client Comment *Thanks for your suggestion, we allow establish zero debt.*

```
156 function _mintInternal(address to, uint128 debtAmt, bytes memory
    ↪ collateralData, GtConfig memory config)
```

CVF-13 INFO

- **Category** Suboptimal
- **Source** AbstractGearingTokenV2.sol

Description These two function have very similar code.

Recommendation Extract common logic into a function.

Client Comment *Thanks for your suggestion, there were some different between them so I don't consider to change them.*

```
156 function _mintInternal(address to, uint128 debtAmt, bytes memory
    ↪ collateralData, GtConfig memory config)
```

```
174 function augmentDebt(address caller, uint256 id, uint256 ftAmt)
    ↪ external virtual override nonReentrant onlyOwner {
```

CVF-14 FIXED

- **Category** Unclear behavior
- **Source** AbstractGearingTokenV2.sol

Description There is no check to ensure the "ids" array is not empty.

Recommendation Implement such a check.

```
220 function merge(uint256[] memory ids) external virtual nonReentrant
    ↪ returns (uint256 newId) {
```

CVF-15 FIXED

- **Category** Bad naming
- **Source** AbstractGearingTokenV2.sol

Description The error name doesn't match the error condition. What actually checked is that all the loans being merged belong to msg.sender, not just to the same owner.

Recommendation Rename the error or change the check.

```
227 if (msg.sender != owner) {
        revert CanNotMergeLoanWithDiffOwner(id, owner);
```

CVF-16 INFO

- **Category** Suboptimal
- **Source** AbstractGearingTokenV2.sol

Description Here a new bytes array is allocated on every iteration.

Recommendation Refactor the code to merge collateral in place.

Client Comment *Thanks for your reminder, I don't consider modifying it because the impact is limited.*

```
231 mergedCollateralData =
        i == 0 ? loan.collateralData : _mergeCollateral(
            ↪ mergedCollateralData, loan.collateralData);
```

CVF-17 FIXED

- **Category** Suboptimal
- **Source** AbstractGearingTokenV2.sol

Recommendation It would be more efficient to merge all the remaining loans into the first loan, instead of creating a new loan.

```
235 newId = _mintInternal(msg.sender, totalDebtAmt, mergedCollateralData
        ↪ , _config);
```

CVF-18 FIXED

- **Category** Suboptimal
- **Source** AbstractGearingTokenV2.sol

Description In case the "ids" array is empty, the "mergedCollateralData" passed here will be uninitialized.

Recommendation Forbid merging zero loans.

```
235 newId = _mintInternal(msg.sender, totalDebtAmt, mergedCollateralData
        ↪ , _config);
```

CVF-19 FIXED

- **Category** Unclear behavior
- **Source** AbstractGearingTokenV2.sol

Description It is unclear why collateral is transferred only on full repayment, while the loan is updated only on partial repayment. Such inconsistent behavior makes using this function very error-prone.

Recommendation Implement a consistent behavior.

328
```
if (loan.debtAmt == 0) {
```

331
```
    _burnInternal(id);
    _transferCollateral(gtOwner, loan.collateralData);
```

334
```
} else {
    loanMapping[id].debtAmt = loan.debtAmt;
```

CVF-20 INFO

- **Category** Procedural
- **Source** AbstractGearingTokenV2.sol

Recommendation The corresponding “loanMapping” entry should also be deleted here.

Client Comment *It’s deleted in the internal function. function _burnInternal(uint256 id) internal { _burn(id); delete loanMapping[id]; }*

330
```
// Burn this nft
_burnInternal(id);
```

CVF-21 FIXED

- **Category** Overflow/Underflow
- **Source** AbstractGearingTokenV2.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while certain intermediary calculation overflows.

Recommendation Use the “mulDiv” function.

```
429         maxRepayAmt = (valueAndPrice.collateralValue * Constants.
                ↪ DECIMAL_BASE) / valueAndPrice.priceDenominator
```

```
482             (loan.debtAmt * valueAndPrice.debtPrice) / valueAndPrice
                    ↪ .debtDenominator;
```

```
565 valueAndPrice.debtValueWithDecimals = (loan.debtAmt * valueAndPrice.
      ↪ debtPrice) / valueAndPrice.debtDenominator;
```

```
577     (valueAndPrice.debtValueWithDecimals * Constants.DECIMAL_BASE_SQ
            ↪ )
          / (valueAndPrice.collateralValue * valueAndPrice.
              ↪ priceDenominator)
```

CVF-22 FIXED

- **Category** Suboptimal
- **Source** AbstractGearingTokenV2.sol

Description Calculating denominators on each invocation is inefficient, taking into account the oracle decimals usually don’t change over time.

Recommendation Refactor the code, to either calculate the denominators once, or to obtain denominator rather than decimals from the oracle.

Client Comment *Price decimals may change, so we need to calculate the price denominator.*

```
561 valueAndPrice.priceDenominator = 10 ** priceDecimals;
```

```
563 valueAndPrice.debtDenominator = 10 ** debtDecimals;
```

CVF-23 FIXED

- **Category** Overflow/Underflow
- **Source** AbstractGearingTokenV2.sol

Description Having decimal multipliers in both, the numerator and the denominator increases chances of phantom overflow.

Recommendation Compare the decimals multipliers, then divide the bigger by the smaller, and use the division result.

```
577 (valueAndPrice.debtValueWithDecimals * Constants.DECIMAL_BASE_SQ)
        / (valueAndPrice.collateralValue * valueAndPrice.
            ↪ priceDenominator)
```

CVF-24 FIXED

- **Category** Overflow/Underflow
- **Source** GearingTokenWithERC20V2.sol

Recommendation Phantom overflow is possible here, i.e. a situation when the final calcualtion result would fit into the destination type, while certain intermediary calcualtion overflows.

```
54 uint256 amount = (collateralReserve * proportion) / Constants.
     ↪ DECIMAL_BASE_SQ;
```

CVF-25 FIXED

- **Category** Overflow/Underflow
- **Source** GearingTokenWithERC20V2.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while certain intermediary calculation overflows.

Recommendation Use the “mulDiv” function.

```
100 return (collateralAmt * price * Constants.DECIMAL_BASE) / (
      ↪ priceDenominator * collateralDemonimator);
```

CVF-26 FIXED

- **Category** Procedural
- **Source** GearingTokenWithERC20V2.sol

Recommendation This commented out code should be removed or uncommented.

173
```
// maxRomvedCollateral = min(
// (repayAmt * (1 + REWARD_TO_LIQUIDATOR + REWARD_TO_PROTOCOL)) *
   ↪ debtTokenPrice / collateralTokenPrice ,
// collateralAmt *(repayAmt / debtAmt)
// )
```

CVF-27 INFO

- **Category** Unclear behavior
- **Source** GearingTokenWithERC20V2.sol

Description This expression in this comment looks incorrect both, syntactically and semantically, which makes it much harder to read.

Recommendation Fix the expression.

Client Comment *We did not find any error in the calculation formula. Please point out the error. Thank you.*

184
```
*     ddPriceToCdPrice = roundUp((DP/DPD) / (CP/CPD) = (DP*CPD*SD) /
   ↪  (CP*DPD))
```

CVF-28 FIXED

- **Category** Overflow/Underflow
- **Source** GearingTokenWithERC20V2.sol

Description Phantom overflow is possible here, i.e. a situation where the final calculation result would fit into the destination type, while certain intermediary calculation overflows.

Recommendation Use the “mulDiv” function.

187
```
uint256 ddPriceToCdPrice = (
    valueAndPrice.debtPrice * cPriceDenominator * cPriceDenominator
       ↪ * 10 + collateralPrice - 1
) / collateralPrice;
```

193
```
uint256 cEqualRepayAmt = (repayAmt * ddPriceToCdPrice *
   ↪ cTokenDenominator)
    / (valueAndPrice.debtDenominator * cPriceDenominator *
       ↪ valueAndPrice.priceDenominator * 10);
```

CVF-29 FIXED

- **Category** Procedural
- **Source** GearingTokenWithERC20V2.sol

Description A rounded up value “ddPriceToCdPrice” is used in the numerator of a rounded down fraction, which makes the rounding of the final result unpredictable.

Recommendation Use consistent rounding.

193
```
uint256 cEqualRepayAmt = (repayAmt * ddPriceToCdPrice *
   ↪ cTokenDenominator)
```

CVF-30 FIXED

- **Category** Suboptimal
- **Source** GearingTokenWithERC20V2.sol

Description This logic is overcomplicated and could be simplified.

```
208 // Case 1: removed collateral can not cover repayAmt +
    ↪ rewardToLiquidator
    if (removedCollateralAmt <= cEqualRepayAmt + rewardToLiquidator) {
210     cToLiquidator = _encodeAmount(removedCollateralAmt);
        cToTreasurer = _encodeAmount(0);
        remainningC = _encodeAmount(0);
    }
    // Case 2: removed collateral can cover repayAmt +
    ↪ rewardToLiquidator but not rewardToProtocol
    else if (removedCollateralAmt < cEqualRepayAmt + rewardToLiquidator
    ↪ + rewardToProtocol) {
        cToLiquidator = _encodeAmount(cEqualRepayAmt +
        ↪ rewardToLiquidator);
        cToTreasurer = _encodeAmount(removedCollateralAmt -
        ↪ cEqualRepayAmt - rewardToLiquidator);
        remainningC = _encodeAmount(0);
    }
220 // Case 3: removed collateral equal repayAmt + rewardToLiquidator +
    ↪ rewardToProtocol
    else {
        cToLiquidator = _encodeAmount(cEqualRepayAmt +
        ↪ rewardToLiquidator);
        cToTreasurer = _encodeAmount(rewardToProtocol);
    }
```

Recommendation Simplify the logic like this:

```
uint256 cToLiquidatorAmount = removedCollateralAmt.min (
   ↪ cEqualRepayAmt + rewardToLiquidator);
removedCollateralAmt -= cToLiquidatorAmount;
uint256 cToTreasurerAmount = removedCollateralAmt.min (
   ↪ rewardToProtocol);
removedCollateralAmt -= cToTreasurerAmount;
uint256 remainningCAmount = removedCollateralAmt;
cToLiquidator = \_encodeAmount(cToLiquidatorAmount);
cToTreasurer = \_encodeAmount(cToTreasurerAmount);
remainningC = \_encodeAmount(remainningCAmount);
```

CVF-31 FIXED

- **Category** Documentation
- **Source** IMintableERC20V2.sol

Description The logic of this function is unclear, as sender is supposed to be “msg.sender”.

Recommendation Explain the use cases for this function.

```
16 function burn(address owner, address spender, uint256 amount)
      ↪ external;
```

CVF-32 INFO

- **Category** Unclear behavior
- **Source** StakingBuffer.sol

Description There is not check to ensure that after this adjustment “amountFromPool” is still sufficient to fulfill tie withdrawal request.

Recommendation Implement such a check.

Client Comment *This contract is out of scope, it is still under development.*

```
38 if (amountFromPool > aTokenBalance) {
       amountFromPool = aTokenBalance;
40 }
```

CVF-33 INFO

- **Category** Suboptimal
- **Source** TermMaxToken.sol

Recommendation Unchained initializers should be used here.

Client Comment *This contract is out of scope, it is still under development.*

```
58 __ERC20_init(name, symbol);
   __Ownable_init(admin);
60 __ReentrancyGuard_init();
```

CVF-34 INFO

- **Category** Procedural
- **Source** PreTMX.sol

Description Minting tokens is similar to transferring tokens to the destination address, so minting to an address, which transfers are restricted to should be forbidden.

Recommendation Forbid minting to addresses which transfers are restricted to.

Client Comment *We want to let the owner have the permission that mint/burn tokens.*

```
53 function mint(address to, uint256 amount) external onlyOwner {
```

CVF-35 INFO

- **Category** Suboptimal
- **Source** PreTMX.sol

Description The “transferRestricted” flag is checked twice.

Recommendation Refactor the code to check the flag only once.

Client Comment *The logic of this process is transferring from or to whitelist address is allowed.*

```
62 if (transferRestricted && !isTransferredFromWhitelisted[from]) {
```

```
65 if (transferRestricted && !isTransferredToWhitelisted[to]) {
```

CVF-36 INFO

- **Category** Suboptimal
- **Source** TermMaxSwapAdapter.sol

Description Approving on every loop iteration is waste of gas.

Recommendation Approve once before the loop.

Client Comment *This contract is out of scope, it is still under development.*

```
35 tokenIn.forceApprove(order, data.netTokenAmt);
```

```
44 tokenIn.forceApprove(order, data.netTokenAmt);
```

CVF-37 FIXED

- **Category** Procedural
- **Source** ITermMaxRouterV2.sol

Recommendation These functions are a part of the administrative API and should be removed from the public router interface.

```
31 function pause() external;
```

```
37 function unpause() external;
```

```
53 function setAdapterWhitelist(address adapter, bool isWhitelist)
   ↪ external;
```

CVF-38 FIXED

- **Category** Suboptimal
- **Source** TermMaxRouterV2.sol

Recommendation Unchained initializers should be used here.

```
69 __UUPSUpgradeable_init();
70 __Pausable_init();
__Ownable_init(admin);
```

CVF-39 FIXED

- **Category** Suboptimal
- **Source** TermMaxRouterV2.sol

Description The allowance here is increased by the maximum input amount, and the actual input amount could be smaller than this value. However, extra allowance is never revoked.

Recommendation Revoke unused allowance after performing the swap.

```
167 tokenIn.safeIncreaseAllowance(address(order), maxTokenIn);
```

CVF-40 FIXED

- **Category** Procedural
- **Source** TermMaxRouterV2.sol

Recommendation This low-level function should be moved into a utility library.

```
174 function sum(uint128[] memory values) internal pure returns (uint256
    ↪  total) {
```

CVF-41 FIXED

- **Category** Overflow/Underflow
- **Source** TermMaxPriceFeedConverter.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation resuol would fit into the destination type while certain intermediary calculation overflows.

Recommendation Use the mulDiv function.

```
60 answer = answer * answer2 * int256((10 ** decimals())) /
    ↪ priceDemonitor;
```

CVF-42 FIXED

- **Category** Suboptimal
- **Source** TermMaxPriceFeedConverter.sol

Description The value “10 ** decimals()” is actually constant. Calculating it on every invocation is waste of gas.

Recommendation Refactor the code to avoid recalculating constant values.

```
60 answer = answer * answer2 * int256((10 ** decimals())) /
    ↪ priceDemonitor;
```

CVF-43 FIXED

- **Category** Suboptimal
- **Source** TermMaxPriceFeedConverter.sol

Description As long as `10 ** decimals()` is constant, `priceDenominator` is immutable, and `priceDenominator` is a power of 10, this expression could be optimized either as `answer * answer2 * scaleUp" or "answer * answer2 / scaleDown` where `scaleUp` and `scaleDown` are immutable powers of 10.

Recommendation Do the suggested optimization.

```
60 answer = answer * answer2 * int256((10 ** decimals())) /
       ↪ priceDemonitor;
```

CVF-44 INFO

- **Category** Unclear behavior
- **Source** OracleAggregatorV2.sol

Description There is no range check for the "timeLock" argument.

Recommendation Implement an appropriate check.

Client Comment *Thanks for your suggestion, this check is optional because it is an immutable value.*

```
116 constructor(address _owner, uint256 timeLock) Ownable(_owner) {
```

CVF-45 FIXED

- **Category** Unclear behavior
- **Source** OracleAggregatorV2.sol

Description There is no explicit check to ensure an oracle for the given asset even exists.

Recommendation Implement such an explicit check.

```
202 Oracle memory oracle = oracles[asset];
```

CVF-46 FIXED

- **Category** Procedural
- **Source** AccessManagerV2.sol

Recommendation This conditional operator should be outside the loop for efficiency.

24
```
if (state) {
```

CVF-47 INFO

- **Category** Unclear behavior
- **Source** TermMaxOrderV2.sol

Description There are no range checks for the arguments.

Recommendation Add appropriate range checks.

Client Comment *They are private functions.*

60
```
function setInitialFtReserve(uint256 ftReserve) private {
```

66
```
function setInitialXtReserve(uint256 xtReserve) private {
```

CVF-48 FIXED

- **Category** Suboptimal
- **Source** TermMaxOrderV2.sol

Recommendation Unchained initializers should be used here.

147
```
__Ownable_init(maker_);
__ReentrancyGuard_init();
__Pausable_init();
```

CVF-49 INFO

- **Category** Overflow/Underflow
- **Source** TermMaxOrderV2.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while certain intermediary calculation overflows.

Recommendation Use the “mulDiv function”.

Client Comment *Thanks for your suggestion, I prefer a more intuitive display logic here.*

196
```
        ((lendVFtReserve * Constants.DECIMAL_BASE * Constants.
           ↪ DAYS_IN_YEAR) / (lendVXtReserve * daysToMaturity));
```

207
```
        (borrowVFtReserve * Constants.DECIMAL_BASE * Constants.
           ↪ DAYS_IN_YEAR)
            / (borrowVXtReserve * daysToMaturity)
```

268
```
                != (
                    newCurveCuts.lendCurveCuts[i - 1].liqSquare
270                     * (
                            (
                                (
                                    newCurveCuts.lendCurveCuts[i].
                                       ↪ xtReserve.plusInt256(
                                        newCurveCuts.lendCurveCuts[i
                                           ↪ ].offset
                                    )
                                ) ** 2 * Constants.DECIMAL_BASE
                            )
                                / (
                                    newCurveCuts.lendCurveCuts[i].
                                       ↪ xtReserve.plusInt256(
280                                     newCurveCuts.lendCurveCuts[i
                                           ↪  - 1].offset
                                    ) ** 2
                                )
                        )
                ) / Constants.DECIMAL_BASE
```

```
301                 != (
                        newCurveCuts.borrowCurveCuts[i - 1].liqSquare
                            * (
                                (
                                    (
                                        newCurveCuts.borrowCurveCuts[i].
                                           ↪ xtReserve.plusInt256(
                                            newCurveCuts.borrowCurveCuts
                                               ↪ [i].offset
                                        )
                                    ) ** 2 * Constants.DECIMAL_BASE
310                                 )
                                    / (
                                        newCurveCuts.borrowCurveCuts[i].
                                           ↪ xtReserve.plusInt256(
                                            newCurveCuts.borrowCurveCuts
                                               ↪ [i - 1].offset
                                        ) ** 2
                                    )
                            )
                    ) / Constants.DECIMAL_BASE
```

```
473 feeAmt = (tokenAmtOut * (Constants.DECIMAL_BASE + uint256(feeConfig.
       ↪ borrowMakerFeeRatio))) / nif - tokenAmtOut;
```

```
487 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
524 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
537 feeAmt = (debtTokenAmtOut * (Constants.DECIMAL_BASE + uint256(
       ↪ feeConfig.borrowMakerFeeRatio))) / nif
```

```
660 feeAmt = (negDeltaFt * (Constants.DECIMAL_BASE + uint256(feeConfig.
       ↪ borrowMakerFeeRatio))) / nif - negDeltaFt;
```

```
674 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
742 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
759 feeAmt = (negDeltaFt * (Constants.DECIMAL_BASE + uint256(feeConfig.
       ↪ borrowMakerFeeRatio))) / nif - negDeltaFt;
```

(... 770)

CVF-50 INFO

- **Category** Suboptimal
- **Source** TermMaxOrderV2.sol

Description These checks makes the “libSquare” fields for all cuts except for the first one redundant.

Recommendation Pass “libSquare” only for the first cut.

Client Comment *We need to check that the curve is continuous.*

```
267 newCurveCuts.lendCurveCuts[i].liqSquare
        != (
```

```
300 newCurveCuts.borrowCurveCuts[i].liqSquare
        != (
```

CVF-51 FIXED

- **Category** Overflow/Underflow
- **Source** TermMaxMarketV2.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type while certain intermediary calculation overflows.

Recommendation Use the “mulDiv” function.

```
80  return (daysToMaturity * uint256(_config.feeConfig.mintGtFeeRatio) *
       ↪  uint256(_config.feeConfig.mintGtFeeRef))
        / (Constants.DAYS_IN_YEAR * Constants.DECIMAL_BASE + uint256(
           ↪ _config.feeConfig.mintGtFeeRef) * daysToMaturity);
```

```
273 uint128 debt = ((xtAmt * Constants.DECIMAL_BASE) / (Constants.
       ↪ DECIMAL_BASE - mintGtFeeRatio())).toUint128();
```

```
308 uint128 issueFee = ((debt * mintGtFeeRatio()) / Constants.
       ↪ DECIMAL_BASE).toUint128();
```

```
338 uint128 issueFee = ((debt * mintGtFeeRatio()) / Constants.
       ↪ DECIMAL_BASE).toUint128();
```

```
367 uint256 proportion = (ftAmount * Constants.DECIMAL_BASE_SQ) / (ft.
       ↪ totalSupply() - ft.balanceOf(address(this)));
```

```
419 uint256 proportion = (ftAmount * Constants.DECIMAL_BASE_SQ) / ft.
       ↪ totalSupply();
```

CVF-52 FIXED

- **Category** Suboptimal
- **Source** TermMaxMarketV2.sol

Recommendation Unchained initializers should be used here.

```
88 __Ownable_init(params.admin);
   __ReentrancyGuard_init();
```

CVF-53 FIXED

- **Category** Procedural
- **Source** TermMaxMarketV2.sol

Recommendation This low-level function should be moved into a library.

```
137 function _contactString(string memory a, string memory b) internal
      ↪ pure returns (string memory) {
```

8 Minor Issues

CVF-54 FIXED

- **Category** Procedural
- **Source** AbstractGearingTokenV2.sol

Description Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

Recommendation Also relevant for: GearingTokenWithERC20V2.sol, IGearingTokenV2.sol, IMintableERC20V2.sol, ITermMaxToken.sol, MintableERC20V2.sol, StakingBuffer.sol, TermMaxToken.sol, PreTMX.sol, TermMaxSwapAdapter.sol, TermMaxTokenAdapter.sol, ITermMaxRouterV2.sol, TermMaxRouterV2.sol, ITermMaxPriceFeed.sol, TermMaxERC4626PriceFeed.sol, TermMaxPriceFeedConverter.sol, TermMaxPTPriceFeed.sol, IOracleV2.sol, OracleAggregatorV2.sol, TermMaxFactoryV2.sol, TermMaxPriceFeedFactoryV2.sol, AccessManagerV2.sol, TermMaxOrderV2.sol, TermMaxMarketV2.sol.

Client Comment *Thanks for the suggestion, this is now optional.*

```
2 pragma solidity ^0.8.27;
```

CVF-55 INFO

- **Category** Procedural
- **Source** AbstractGearingTokenV2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
9  import {Constants} from "../../v1/lib/Constants.sol";
10 import {GearingTokenConstants} from "../../v1/lib/
      ↪ GearingTokenConstants.sol";
   import {TransferUtils} from "../../v1/lib/TransferUtils.sol";
   import {IFlashRepayer} from "../../v1/tokens/IFlashRepayer.sol";
   import {IGearingToken, IERC20Metadata, IERC20} from "../../v1/tokens
      ↪ /IGearingToken.sol";
   import {GearingTokenErrors} from "../../v1/errors/GearingTokenErrors
      ↪ .sol";
   import {GearingTokenEvents} from "../../v1/events/GearingTokenEvents
      ↪ .sol";
   import {GtConfig, IOracle} from "../../v1/storage/TermMaxStorage.sol
      ↪ ";
```

CVF-56 FIXED

- **Category** Documentation
- **Source** AbstractGearingTokenV2.sol

Description It is unclear what this “with price and token decimals” means.

Recommendation Elaborate more.

```
49 /// @notice USD value of debt with price and token decimals
50 uint256 debtValueWithDecimals;
```

CVF-57 FIXED

- **Category** Documentation
- **Source** AbstractGearingTokenV2.sol

Description It is unclear what price the denominator belong to.

Recommendation Use a more specific name and elaborate more in the comment.

```
53 /// @notice Denominator of USD price
   uint256 priceDenominator;
```

CVF-58 FIXED

- **Category** Documentation
- **Source** AbstractGearingTokenV2.sol

Description It is unclear what “denominator of a token” means.

Recommendation Elaborate more.

```
55 /// @notice Denominator of debt token
   uint256 debtDenominator;
```

CVF-59 FIXED

- **Category** Documentation
- **Source** AbstractGearingTokenV2.sol

Recommendation The format of this data is unclear.

```
57 /// @notice Encoded USD price of collateral token
   bytes collateralPriceData;
```

CVF-60 FIXED

- **Category** Procedural
- **Source** AbstractGearingTokenV2.sol

Description These variables don't have access level specified, so internal access will be used by default.

Recommendation Explicitly specify an access level.

```
62 GtConfig _config;
```

```
64 uint256 total;
```

```
66 mapping(uint256 => LoanInfo) loanMapping;
```

```
68 uint8 debtDecimals;
```

CVF-61 FIXED

- **Category** Bad naming
- **Source** AbstractGearingTokenV2.sol

Description The name is too generic.

Recommendation Use a more specific name.

```
64 uint256 total;
```

CVF-62 FIXED

- **Category** Suboptimal
- **Source** AbstractGearingTokenV2.sol

Recommendation Unchained initializers should be used here.

```
91 __ERC721_init(name, symbol);
   __Ownable_init(msg.sender);
```

CVF-63 INFO

- **Category** Unclear behavior
- **Source** AbstractGearingTokenV2.sol

Description This function should emit some event.

Client Comment *Events are emitted in the market, and they are linked.*

```
102 function setTreasurer(address treasurer) external virtual onlyOwner
      ↪ {
```

CVF-64 INFO

- **Category** Unclear behavior
- **Source** AbstractGearingTokenV2.sol

Description There is no range check for the “ftAmt” argument.

Recommendation Implement a check to ensure “ftAmt” is not zero.

Client Comment *Thanks for the suggestion, I thought it was moot that this check was optional.*

```
174 function augmentDebt(address caller, uint256 id, uint256 ftAmt)
      ↪ external virtual override nonReentrant onlyOwner {
```

CVF-65 FIXED

- **Category** Suboptimal
- **Source** AbstractGearingTokenV2.sol

Description The error condition actually doesn't depend on "id" so including "id" into the error looks redundant.

Recommendation Reconsider error parameters.

```
295 revert GtIsExpired(id);
```

```
349 revert GtIsExpired(id);
```

```
378 revert GtIsExpired(id);
```

CVF-66 FIXED

- **Category** Procedural
- **Source** AbstractGearingTokenV2.sol

Recommendation Brackets are redundant.

```
429 maxRepayAmt = (valueAndPrice.collateralValue * Constants.
    ↪ DECIMAL_BASE) / valueAndPrice.priceDenominator
```

```
482     (loan.debtAmt * valueAndPrice.debtPrice) / valueAndPrice.
        ↪ debtDenominator;
```

CVF-67 INFO

- **Category** Procedural
- **Source** GearingTokenWithERC20V2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
4 import {MathLib} from "../../v1/lib/MathLib.sol";
```

CVF-68 INFO

- **Category** Suboptimal
- **Source** GearingTokenWithERC20V2.sol

Recommendation This error could be made more useful by adding certain parameters into it.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
19 error CollateralCapacityExceeded();
```

CVF-69 FIXED

- **Category** Procedural
- **Source** GearingTokenWithERC20V2.sol

Description There is no access level specified for this variable, so internal access will be used by default.

Recommendation Explicitly specify an access level.

```
30 uint8 collateralDecimals;
```

CVF-70 FIXED

- **Category** Procedural
- **Source** GearingTokenWithERC20V2.sol

Recommendation Brackets are redundant.

```
54 uint256 amount = (collateralReserve * proportion) / Constants.
   ↪ DECIMAL_BASE_SQ;
```

CVF-71 FIXED

- **Category** Suboptimal
- **Source** GearingTokenWithERC20V2.sol

Description Using dynamic denominator makes code less efficient. Also, it increases the chances of phantom overflow.

Recommendation Use constant denominators or precomputed denominators.

```
98  (uint256 price, uint256 priceDenominator, uint256
       ↪ collateralDemonimator) =
        abi.decode(priceData, (uint256, uint256, uint256));
100 return (collateralAmt * price * Constants.DECIMAL_BASE) / (
       ↪ priceDenominator * collateralDemonimator);
```

CVF-72 FIXED

- **Category** Suboptimal
- **Source** GearingTokenWithERC20V2.sol

Description Calculating denominators from decimals on every invocation is inefficient.

Recommendation Use constant or precomputed denominators.

```
114 uint256 priceDenominator = 10 ** decimals;
```

```
116 uint256 cTokenDenominator = 10 ** collateralDecimals;
```

CVF-73 FIXED

- **Category** Procedural
- **Source** GearingTokenWithERC20V2.sol

Recommendation Brackets are redundant.

```
197     (cEqualRepayAmt * GearingTokenConstants.REWARD_TO_LIQUIDATOR) /
           ↪ Constants.DECIMAL_BASE;
    uint256 rewardToProtocol = (cEqualRepayAmt * GearingTokenConstants.
       ↪ REWARD_TO_PROTOCOL) / Constants.DECIMAL_BASE;
```

CVF-74 FIXED

- **Category** Procedural
- **Source** GearingTokenWithERC20V2.sol

Recommendation Brackets around multiplication are redundant.

```
205 removedCollateralAmt = removedCollateralAmt.min((collateralAmt *
    ↪ repayAmt) / loan.debtAmt);
```

CVF-75 FIXED

- **Category** Documentation
- **Source** IGearingTokenV2.sol

Description This phrase sounds odd.

Recommendation Fix or rephrase.

```
10 ///         the collateral will send by flashloan first.
```

CVF-76 FIXED

- **Category** Procedural
- **Source** IMintableERC20V2.sol

Description We didn't review this file.

```
4 import "../../v1/tokens/IMintableERC20.sol";
```

CVF-77 FIXED

- **Category** Procedural
- **Source** IMintableERC20V2.sol

Description This import is not used.

Recommendation Remove it.

```
4 import "../../v1/tokens/IMintableERC20.sol";
```

CVF-78 INFO

- **Category** Bad datatype
- **Source** ITermMaxToken.sol

Recommendation The return type should be more specific.

Client Comment *This contract is out of scope, it is still under development.*

```
29 function aToken() external view returns (address);
```

CVF-79 INFO

- **Category** Bad datatype
- **Source** ITermMaxToken.sol

Recommendation The return type should be “IERC20”.

Client Comment *This contract is out of scope, it is still under development.*

```
38 function asset() external view returns (address);
```

CVF-80 INFO

- **Category** Bad naming
- **Source** ITermMaxToken.sol

Description The name “to” sounds inappropriate here.

Recommendation Rename the “to” argument to “from”.

Client Comment *This contract is out of scope, it is still under development.*

```
76 function burn(address to, uint256 amount) external;
```

CVF-81 INFO

- **Category** Procedural
- **Source** MintableERC20V2.sol

Description We didn’t review this file.

Client Comment *They were audit in other competition.*

```
5 import {MintableERC20} from "../../v1/tokens/MintableERC20.sol";
```

CVF-82 INFO

- **Category** Procedural
- **Source** StakingBuffer.sol

Description We didn't review this file.

Client Comment *They were audit in other competition.*

```
4 import {TransferUtils, IERC20} from "../../v1/lib/TransferUtils.sol"
    ↪ ;
```

CVF-83 INFO

- **Category** Bad datatype
- **Source** StakingBuffer.sol

Recommendation The type for the "assetAddr" argument should be "IERC20".

Client Comment *This contract is out of scope, it is still under development.*

```
17 function _depositWithBuffer(address assetAddr) internal {
```

```
25 function _withdrawWithBuffer(address assetAddr, address to, uint256
    ↪ amount) internal {
```

```
49 function _bufferConfig(address assetAddr) internal view virtual
    ↪ returns (BufferConfig memory);
```

```
51 function _depositToPool(address assetAddr, uint256 amount) internal
    ↪ virtual;
```

```
53 function _withdrawFromPool(address assetAddr, address to, uint256
    ↪ amount) internal virtual;
```

```
55 function _aTokenBalance(address assetAddr) internal view virtual
    ↪ returns (uint256 amount);
```

CVF-84 INFO

- **Category** Readability
- **Source** StakingBuffer.sol

Description The code below looks like it is always executed, while actually it is executed only when the buffer is insufficient.

Recommendation Put the rest of the function into an explicit “else” branch.

Client Comment *This contract is out of scope, it is still under development.*

```
29 if (assetBalance >= amount && assetBalance - amount >= bufferConfig.
      ↪ minimumBuffer) {
```

```
32     return;
   }
```

CVF-85 INFO

- **Category** Procedural
- **Source** TermMaxToken.sol

Description We didn’t review these files.

Client Comment *They were audit in other competition.*

```
13 import {TransferUtils} from "../../v1/lib/TransferUtils.sol";
```

```
17 import {PendingLib, PendingAddress} from "../../v1/lib/PendingLib.
      ↪ sol";
```

CVF-86 INFO

- **Category** Bad datatype
- **Source** TermMaxToken.sol

Recommendation The type for the “aavePool_” argument should be “IAaveV3Minimal”.

Client Comment *This contract is out of scope, it is still under development.*

```
47 constructor(address aavePool_, uint16 referralCode_) {
```

CVF-87 INFO

- **Category** Bad datatype
- **Source** TermMaxToken.sol

Recommendation The type for the “underlying_” argument should be “IERC20”.

Client Comment *This contract is out of scope, it is still under development.*

```
53 function initialize(address admin, address underlying_, BufferConfig
   ↪  memory bufferConfig_) public initializer {
```

CVF-88 INFO

- **Category** Bad datatype
- **Source** TermMaxToken.sol

Recommendation The type for the “assetAddr” argument should be “IERC20”.

Client Comment *This contract is out of scope, it is still under development.*

```
129 function _depositToPool(address assetAddr, uint256 amount) internal
   ↪ virtual override {
```

```
134 function _withdrawFromPool(address assetAddr, address to, uint256
   ↪ amount) internal virtual override {
```

CVF-89 INFO

- **Category** Suboptimal
- **Source** TermMaxToken.sol

Description This check is redundant, as a dead address can be passed anyway.

Recommendation Remove the check.

Client Comment *This contract is out of scope, it is still under development.*

```
147 if (newImplementation == address(0)) revert InvalidImplementation();
```

CVF-90 FIXED

- **Category** Suboptimal
- **Source** PreTMX.sol

Recommendation It would be more efficient to merge these two mapping into a single mapping whose keys are addresses and values are bit masks encapsulating values of the original mappings.

```
10 mapping(address => bool) public isTransferredFromWhitelisted;
   mapping(address => bool) public isTransferredToWhitelisted;
```

CVF-91 FIXED

- **Category** Procedural
- **Source** PreTMX.sol

Recommendation The address parameters should be indexed.

```
17 event TransferFromWhitelisted(address from, bool isWhitelisted);
   event TransferToWhitelisted(address to, bool isWhitelisted);
```

CVF-92 INFO

- **Category** Bad datatype
- **Source** PreTMX.sol

Recommendation The initial supply should be a named constant.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
21 _mint(admin, 1e9 ether);
```

CVF-93 INFO

- **Category** Procedural
- **Source** TermMaxSwapAdapter.sol

Description We didn't review this file.

Client Comment *They were audit in other competition.*

```
5 import {ITermMaxOrder} from "contracts/interfaces/ITermMaxOrder.sol"
    ↪ ;
```

CVF-94 INFO

- **Category** Suboptimal
- **Source** TermMaxSwapAdapter.sol

Description Declaring a top-level type in a file named after a contract makes it harder navigating through code.

Recommendation Move the type declaration into the contract or move it into a separate file.

Client Comment *This contract is out of scope, it is still under development.*

```
8 struct TermMaxSwapData {
```

CVF-95 INFO

- **Category** Bad datatype
- **Source** TermMaxSwapAdapter.sol

Recommendation The type for these fields should be “IERC20”.

Client Comment *This contract is out of scope, it is still under development.*

```
10 address tokenIn;
   address tokenOut;
```

CVF-96 INFO

- **Category** Bad datatype
- **Source** TermMaxSwapAdapter.sol

Recommendation The type for this field should be “ITermMaxOrder[]”.

Client Comment *This contract is out of scope, it is still under development.*

```
12 address[] orders;
```

CVF-97 INFO

- **Category** Suboptimal
- **Source** TermMaxSwapAdapter.sol

Recommendation It would be more efficient to use a single array of structs with two fields, instead of two parallel arrays. This would also make the length check unnecessary.

Client Comment *This contract is out of scope, it is still under development.*

```
12 address[] orders;
   uint128[] tradingAmts;
```

CVF-98 INFO

- **Category** Suboptimal
- **Source** TermMaxSwapAdapter.sol

Recommendation This error could be made more useful by adding certain parameters into it.

Client Comment *This contract is out of scope, it is still under development.*

```
21 error OrdersAndAmtsLengthNotMatch();
```

CVF-99 INFO

- **Category** Procedural
- **Source** ITermMaxRouterV2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
5  import {ITermMaxMarket, IGearingToken} from "../../v1/ITermMaxMarket
   ↪ .sol";
   import {ITermMaxOrder} from "../../v1/ITermMaxOrder.sol";
   import {SwapUnit} from "../../v1/router/ISwapAdapter.sol";
   import {ISwapCallback} from "../../v1/ISwapCallback.sol";
   import {OrderConfig} from "../../v1/storage/TermMaxStorage.sol";
```

CVF-100 INFO

- **Category** Bad datatype
- **Source** ITermMaxRouterV2.sol

Recommendation The type for these tokens should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
19 address tokenIn;
20 address tokenOut;
```

CVF-101 INFO

- **Category** Bad datatype
- **Source** ITermMaxRouterV2.sol

Recommendation The type for the “adapter” arguments should be “IERC20SwapAdapter”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
45 function adapterWhitelist(address adapter) external view returns (
   ↪ bool);
```

```
53 function setAdapterWhitelist(address adapter, bool isWhitelist)
   ↪ external;
```

CVF-102 INFO

- **Category** Suboptimal
- **Source** ITermMaxRouterV2.sol

Description It looks weird to return fixes-sized arrays here, even if for the current implementation this does make sense.

Recommendation Return dynamic arrays for “tokens” and “balances”.

Client Comment *Thanks for the suggestion. We use fixed array because the market contains 4 types of ERC20 token.*

```
68 returns (IERC20[4] memory tokens, uint256[4] memory balances,
   ↪ address gt, uint256[] memory gtIds);
```

CVF-103 INFO

- **Category** Suboptimal
- **Source** ITermMaxRouterV2.sol

Recommendation Consider returning a single array of structs with two fields, instead of two parallel arrays.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
68 returns (IERC20[4] memory tokens, uint256[4] memory balances,
      ↪ address gt, uint256[] memory gtIds);
```

CVF-104 INFO

- **Category** Bad datatype
- **Source** ITermMaxRouterV2.sol

Recommendation The type for the “gt” returned value should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
68 returns (IERC20[4] memory tokens, uint256[4] memory balances,
      ↪ address gt, uint256[] memory gtIds);
```

CVF-105 FIXED

- **Category** Suboptimal
- **Source** ITermMaxRouterV2.sol

Description It is unclear what swap path is referred here, as there is no corresponding argument.

Recommendation Elaborate more.

```
76 * @param orders Array of orders to use for the swap path
```

```
98 * @param orders Array of orders to use for the swap path
```

CVF-106 INFO

- **Category** Suboptimal
- **Source** ITermMaxRouterV2.sol

Recommendation It would be more efficient to return a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

Client Comment *Thanks for the suggestion, the changes will include in the next version.*

```
86  ITermMaxOrder[] memory orders,
    uint128[] memory tradingAmts,
```

```
108 ITermMaxOrder[] memory orders,
    uint128[] memory tradingAmts,
```

```
132 ITermMaxOrder[] memory orders,
    uint128[] memory amtsToSellTokens,
```

```
156 ITermMaxOrder[] memory orders,
    uint128[] memory amtsToBuyXt,
```

```
210 ITermMaxOrder[] memory orders,
    uint128[] memory tokenAmtsWantBuy,
```

```
286 ITermMaxOrder[] memory orders,
    uint128[] memory ftAmtsWantBuy,
```

CVF-107 FIXED

- **Category** Documentation
- **Source** ITermMaxRouterV2.sol

Description Unlike other functions in the same interface, this function is not documented.

Recommendation Document it.

```
185 function leverageFromXtAndCollateral(
```

```
259 function flashRepayFromCollV2(
```

CVF-108 INFO

- **Category** Procedural
- **Source** TermMaxRouterV2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
14 import {ITermMaxMarket} from "../../v1/ITermMaxMarket.sol";

16 import {ITermMaxOrder} from "../../v1/ITermMaxOrder.sol";
   import {SwapUnit} from "../../v1/router/ISwapAdapter.sol";
   import {RouterErrors} from "../../v1/errors/RouterErrors.sol";
   import {RouterEvents} from "../../v1/events/RouterEvents.sol";
20 import {TransferUtils} from "../../v1/lib/TransferUtils.sol";
   import {IFlashLoanReceiver} from "../../v1/IFlashLoanReceiver.sol";
   import {IFlashRepayer} from "../../v1/tokens/IFlashRepayer.sol";

24 import {IGearingToken} from "../../v1/tokens/IGearingToken.sol";

26 import {CurveCuts, OrderConfig} from "../../v1/storage/
      ↪ TermMaxStorage.sol";
   import {Constants} from "../../v1/lib/Constants.sol";
   import {MathLib} from "../../v1/lib/MathLib.sol";
```

CVF-109 FIXED

- **Category** Procedural
- **Source** TermMaxRouterV2.sol

Recommendation This file should be imported as "../ITermMaxMarketV2.sol".

```
15 import {ITermMaxMarketV2} from "../../v2/ITermMaxMarketV2.sol";
```

CVF-110 INFO

- **Category** Procedural
- **Source** TermMaxRouterV2.sol

Recommendation The key type for this mapping should be “IERC20SwapAdapter”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
62 mapping(address => bool) public adapterWhitelist;
```

CVF-111 INFO

- **Category** Bad datatype
- **Source** TermMaxRouterV2.sol

Recommendation The type for the “adapter” argument should be “IERC20SwapAdapter”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
77 function setAdapterWhitelist(address adapter, bool isWhitelist)
      ↪ external onlyOwner {
```

CVF-112 INFO

- **Category** Bad datatype
- **Source** TermMaxRouterV2.sol

Recommendation The type for the “gtAddr” returned value should be ”IERC721”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
89 returns (IERC20[4] memory tokens, uint256[4] memory balances,
      ↪ address gtAddr, uint256[] memory gtIds)
```

CVF-113 INFO

- **Category** Suboptimal
- **Source** TermMaxRouterV2.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

Client Comment *Thanks for the suggestion, the changes will include in the next version.*

```
111 ITermMaxOrder[] memory orders,
    uint128[] memory tradingAmts,
```

```
126 ITermMaxOrder[] memory orders,
    uint128[] memory tradingAmts,
```

```
144 ITermMaxOrder[] memory orders,
    uint128[] memory tradingAmts,
```

```
159 ITermMaxOrder[] memory orders,
160 uint128[] memory tradingAmts,
```

```
185 ITermMaxOrder[] memory orders,
    uint128[] memory amtsToSellTokens,
```

```
205 ITermMaxOrder[] memory orders,
    uint128[] memory amtsToBuyXt,
```

```
288 ITermMaxOrder[] memory orders,
    uint128[] memory tokenAmtsWantBuy,
```

```
408 ITermMaxOrder[] memory orders,
    uint128[] memory ftAmtsWantBuy,
```

CVF-114 INFO

- **Category** Procedural
- **Source** TermMaxRouterV2.sol

Recommendation Brackets around multiplication are redundant.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
322 uint128 debtAmt = ((borrowAmt * Constants.DECIMAL_BASE) / (Constants
    ↪ .DECIMAL_BASE - mintGtFeeRatio)).toUint128();
```

```
346 uint128 debtAmt = ((borrowAmt * Constants.DECIMAL_BASE) / (Constants
    ↪ .DECIMAL_BASE - mintGtFeeRatio)).toUint128();
```

```
691         (swapData.netTokenAmt * Constants.DECIMAL_BASE) / (Constants
              ↪ .DECIMAL_BASE - mintGtFeeRatio)
```

CVF-115 FIXED

- **Category** Documentation
- **Source** TermMaxRouterV2.sol

Description The semantics of the last argument is unclear.

Recommendation Put the argument name in comment next to the argument value.

```
425 gt.repay(gtId, repayAmt, false);
```

```
711         gt.repay(gtId, repaidFtAmt.toUint128(), false);
```

```
718         gt.repay(gtId, repaidDebtAmt.toUint128(), true);
```

CVF-116 INFO

- **Category** Suboptimal
- **Source** TermMaxRouterV2.sol

Description These functions are very similar.

Recommendation Extract common logic into a utility function.

Client Comment *Thanks for the suggestion, the changes will include in the next version.*

```
521 function rolloverGt(
```

```
559 function rolloverGtV2(
```

CVF-117 INFO

- **Category** Documentation
- **Source** TermMaxRouterV2.sol

Description The semantics of the first argument is unclear.

Recommendation Give a descriptive name to the first argument.

Client Comment *The unnamed arguments are unused.*

```
600 function executeOperation(address, IERC20, uint256 amount, bytes
    ↪ memory data)
```

CVF-118 INFO

- **Category** Procedural
- **Source** ITermMaxPriceFeed.sol

Description This version requirement is inconsistent with other files in the same code base.

Recommendation Use consistent version requirements.

```
2 pragma solidity ^0.8.20;
```

CVF-119 INFO

- **Category** Bad datatype
- **Source** ITermMaxPriceFeed.sol

Recommendation The return type should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
13 function asset() external view returns (address);
```

CVF-120 INFO

- **Category** Procedural
- **Source** TermMaxERC4626PriceFeed.sol

Description We didn’t review this file.

Client Comment *They were audit in other competition.*

```
7 import {MathLib} from "contracts/v1/lib/MathLib.sol";
```

CVF-121 INFO

- **Category** Bad datatype
- **Source** TermMaxERC4626PriceFeed.sol

Recommendation The type for this variable should be “IERC4626”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
15 address public immutable asset;
```

CVF-122 INFO

- **Category** Procedural
- **Source** TermMaxERC4626PriceFeed.sol

Recommendation Should probably by “priceDenominator” and “vaultDenominator”.

```
16 int256 immutable priceDemonitor;
   uint256 immutable vaultDemonitor;
```

CVF-123 INFO

- **Category** Bad datatype
- **Source** TermMaxERC4626PriceFeed.sol

Recommendation The type for the “_assetPriceFeed” argument should be “AggregatorV3Interface”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
19 constructor(address _assetPriceFeed, address _asset) {
```

CVF-124 INFO

- **Category** Bad datatype
- **Source** TermMaxERC4626PriceFeed.sol

Recommendation The type for the “asset” argument should be IERC4626”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
19 constructor(address _assetPriceFeed, address _asset) {
```

CVF-125 FIXED

- **Category** Suboptimal
- **Source** TermMaxERC4626PriceFeed.sol

Description In ERC20 the "decimals" property is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.

Recommendation Treat all token amounts are integers.

```
23 vaultDemonitor = 10 ** IERC4626(asset).decimals();
```

CVF-126 INFO

- **Category** Suboptimal
- **Source** TermMaxERC4626PriceFeed.sol

Description Calling both, "previewRedeem" and "convertToAssets" seems redundant and inefficient.

Recommendation Use either of the functions, but not both.

Client Comment *Thank you for your suggestion. We use the smaller value of the two because some vaults have additional fees when redeeming, which cannot present their true value.*

```
59 IERC4626(asset).previewRedeem(vaultDemonitor).min(IERC4626(asset).
   ↪ convertToAssets(vaultDemonitor));
```

CVF-127 FIXED

- **Category** Suboptimal
- **Source** TermMaxERC4626PriceFeed.sol

Description The value "10 ** decimals()" is actually constant. Calculating it on every invocation is waste of gas.

Recommendation Refactor the code to not recalculate constant values.

```
60 answer = answer * int256(vaultAnswer) * int256((10 ** decimals())) /
   ↪  priceDemonitor;
```

CVF-128 INFO

- **Category** Procedural
- **Source** TermMaxPriceFeedConverter.sol

Description We didn't review this file.

Client Comment *They were audit in other competition.*

```
6 import {MathLib} from "contracts/v1/lib/MathLib.sol";
```

CVF-129 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedConverter.sol

Recommendation The type for this variable should be "IERC20".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
17 address public immutable asset;
```

CVF-130 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedConverter.sol

Recommendation The type for the price feed arguments should be "AggregatorV3Interface".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
19 constructor(address _aTokenToBTokenPriceFeed, address
   ↪ _bTokenToCTokenPriceFeed, address _asset) {
```

CVF-131 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedConverter.sol

Recommendation The type for the "_asset" argument should be "IERC20".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
19 constructor(address _aTokenToBTokenPriceFeed, address
      ↪ _bTokenToCTokenPriceFeed, address _asset) {
```

CVF-132 FIXED

- **Category** Suboptimal
- **Source** TermMaxPriceFeedConverter.sol

Recommendation This could be optimized as: 10 ** (aTokenToBTokenPriceFeed.decimals() + bTokenToCTokenPriceFeed.decimals())

```
24 int256(10 ** aTokenToBTokenPriceFeed.decimals()) * int256(10 **
      ↪ bTokenToCTokenPriceFeed.decimals());
```

CVF-133 INFO

- **Category** Procedural
- **Source** TermMaxPTPriceFeed.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
4 import {PendlePYLpOracle} from "@pendle/core-v2/contracts/oracles/
     ↪ PtYtLpOracle/PendlePYLpOracle.sol";
  import {PendlePYOracleLib} from "@pendle/core-v2/contracts/oracles/
     ↪ PtYtLpOracle/PendlePYOracleLib.sol";
  import {PMath} from "@pendle/core-v2/contracts/core/libraries/math/
     ↪ PMath.sol";
  import {IPMarket, IPPrincipalToken, IStandardizedYield} from "
     ↪ @pendle/core-v2/contracts/interfaces/IPMarket.sol";
```

CVF-134 INFO

- **Category** Procedural
- **Source** TermMaxPTPriceFeed.sol

Recommendation UPPER_CASE names are conventionally used for constants, not variables.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
24 PendlePYLpOracle public immutable PY_LP_ORACLE;
```

```
26 IPMarket public immutable MARKET;
```

```
28 uint32 public immutable DURATION;
```

```
30 AggregatorV3Interface public immutable PRICE_FEED;
```

```
33 uint256 private immutable PT_TO_SY_RATE_BASE;
```

CVF-135 INFO

- **Category** Bad datatype
- **Source** TermMaxPTPriceFeed.sol

Recommendation The type for this variable should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
31 address public immutable asset;
```

CVF-136 INFO

- **Category** Bad datatype
- **Source** TermMaxPTPriceFeed.sol

Recommendation The type for the “pendlePYLpOracle” argument should be “PendlePYLpOracle”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
49 constructor(address pendlePYLpOracle, address market, uint32
   ↪ duration, address priceFeed) {
```

CVF-137 INFO

- **Category** Bad datatype
- **Source** TermMaxPTPriceFeed.sol

Recommendation The type for the “market” argument should be “IPMarket”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
49 constructor(address pendlePYLpOracle, address market, uint32
   ↪ duration, address priceFeed) {
```

CVF-138 INFO

- **Category** Bad datatype
- **Source** TermMaxPTPriceFeed.sol

Recommendation The type for the “priceFeed” argument should be “AggregatorV3Interface”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
49 constructor(address pendlePYLpOracle, address market, uint32
   ↪ duration, address priceFeed) {
```

CVF-139 INFO

- **Category** Unclear behavior
- **Source** TermMaxPTPriceFeed.sol

Description There is no range check for the “duration” argument.

Recommendation Implement an appropriate check.

Client Comment *Thanks for your suggestion, this check is optional because it is an immutable value.*

```
49 constructor(address pendlePYLpOracle, address market, uint32
   ↪ duration, address priceFeed) {
```

CVF-140 INFO

- **Category** Suboptimal
- **Source** TermMaxPTPriceFeed.sol

Description In ERC20 the “decimals” property is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.

Recommendation Treat token amounts as integers.

Client Comment *Thanks for the suggestion, I have to check the decimal of pendle tokens for the logic.*

```
58 uint8 syDecimals = _SY.decimals();
   uint8 ptDecimals = _PT.decimals();
```

CVF-141 INFO

- **Category** Suboptimal
- **Source** TermMaxPTPriceFeed.sol

Description In the “mulDiv” function most of gas is consumed to invert the denominator. In case denominator is constant, its reciprocal could be precomputed.

Recommendation Consider implementing a special version of the “mulDiv” function that accepts already inverted denominator.

Client Comment *Thanks for the suggestion, we may change it in the future.*

```
104 answer = ptRateInSy.mulDiv(answer.toUint256(), PT_TO_SY_RATE_BASE).
      ↪ toInt256();
```

CVF-142 FIXED

- **Category** Documentation
- **Source** IOracleV2.sol

Description This argument is not documented.

Recommendation Document it.

```
27 int256 minPrice;
```

CVF-143 INFO

- **Category** Bad datatype
- **Source** IOracleV2.sol

Recommendation The parameter should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
37 error OracleIsNotWorking(address asset);
```

CVF-144 INFO

- **Category** Bad datatype
- **Source** IOracleV2.sol

Recommendation The type for the “asset” argument type should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
48 function getPrice(address asset) external view returns (uint256
   ↪ price, uint8 decimals);
```

```
59 function submitPendingOracle(address asset, Oracle memory oracle)
   ↪ external;
```

```
69 function acceptPendingOracle(address asset) external;
```

```
79 function revokePendingOracle(address asset) external;
```

CVF-145 INFO

- **Category** Procedural
- **Source** IOracleV2.sol

Description Returning price along with its decimals makes proper price handling expensive, as the caller would need to calculate the price scaling factor for each returned price.

Recommendation Use the same decimals for all assets or separate decimals value for each asset, but not a separate decimals value for each returned price.

Client Comment *Thanks for your suggestion, we return it because price decimals may be changed.*

```
48 function getPrice(address asset) external view returns (uint256
   ↪ price, uint8 decimals);
```

CVF-146 FIXED

- **Category** Documentation
- **Source** IOracleV2.sol

Description It is unclear what currency the returned price is denominated in.

Recommendation Clarify in the documentation comment.

```
48 function getPrice(address asset) external view returns (uint256
   ↪ price, uint8 decimals);
```

CVF-147 FIXED

- **Category** Suboptimal
- **Source** IOracleV2.sol

Description These functions looks like a part of oracle administration API and shouldn't be part of the public interface. However, events, potentially emitted by these function, should probably be included into the public interface.

Recommendation Remove these functions from the interface, but add corresponding events instead.

```
59 function submitPendingOracle(address asset, Oracle memory oracle)
     ↪ external;
```

```
69 function acceptPendingOracle(address asset) external;
```

```
79 function revokePendingOracle(address asset) external;
```

CVF-148 FIXED

- **Category** Suboptimal
- **Source** OracleAggregatorV2.sol

Description These two items looks very similar.

Recommendation Merge them or make them more different.

```
14 * - Independent heartbeat configuration for backup oracles
```

```
17 * - Separate backup heartbeat for more flexible oracle management
```

CVF-149 INFO

- **Category** Suboptimal
- **Source** OracleAggregatorV2.sol

Recommendation These errors could be made more useful by adding certain parameters into them.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
38 error InvalidAssetOrOracle();
```

```
43 error AlreadySet();
```

```
48 error AlreadyPending();
```

```
53 error NoPendingValue();
```

```
58 error TimelockNotElapsed();
```

CVF-150 INFO

- **Category** Bad datatype
- **Source** OracleAggregatorV2.sol

Recommendation The type for the “asset” parameters should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
71     address indexed asset,
```

```
92     address indexed asset,
```

```
106 event RevokePendingOracle(address indexed asset);
```

CVF-151 INFO

- **Category** Bad datatype
- **Source** OracleAggregatorV2.sol

Recommendation The key type for these mappings should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
110 mapping(address => Oracle) public oracles;
```

```
114 mapping(address => PendingOracle) public pendingOracles;
```

CVF-152 INFO

- **Category** Suboptimal
- **Source** OracleAggregatorV2.sol

Recommendation It would be more efficient to merge these two mappings into a single mapping whose keys are assets and values are structs encapsulating values of the original mappings.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
110 mapping(address => Oracle) public oracles;
```

```
114 mapping(address => PendingOracle) public pendingOracles;
```

CVF-153 INFO

- **Category** Bad datatype
- **Source** OracleAggregatorV2.sol

Recommendation The type for the “asset” arguments should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
123 function submitPendingOracle(address asset, Oracle memory oracle)
    ↪ external onlyOwner {
```

```
163 function acceptPendingOracle(address asset) external {
```

```
190 function revokePendingOracle(address asset) external onlyOwner {
```

```
201 function getPrice(address asset) external view override returns (
    ↪ uint256, uint8) {
```

CVF-154 FIXED

- **Category** Suboptimal
- **Source** OracleAggregatorV2.sol

Recommendation It would be more efficient to assign the whole structure at once.

```
144 pendingOracles[asset].oracle = oracle;
```

```
146 pendingOracles[asset].validAt = validAt;
```

CVF-155 INFO

- **Category** Documentation
- **Source** OracleAggregatorV2.sol

Description The semantics of the returned values is unclear.

Recommendation Give descriptive names to the returned values and/or explain in the documentation comment.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
201 function getPrice(address asset) external view override returns (
    ↪ uint256, uint8) {
```

CVF-156 FIXED

- **Category** Suboptimal
- **Source** OracleAggregatorV2.sol

Description These two code blocks are very similar.

Recommendation Refactor the code to reduce code duplication.

```
206 (, int256 answer,, uint256 updatedAt,) = oracle.aggregator.
       ↪ latestRoundData();
    // Check if primary oracle is fresh and has positive price
    if ((oracle.heartbeat == 0 || oracle.heartbeat + updatedAt >= block.
       ↪ timestamp) && answer > 0) {
        return (uint256(_processPriceRange(answer, oracle)), oracle.
           ↪ aggregator.decimals());
210 }
```

```
215 (, int256 answer,, uint256 updatedAt,) = oracle.backupAggregator.
       ↪ latestRoundData();
    // Check if backup oracle is fresh and has positive price
    if ((oracle.backupHeartbeat == 0 || oracle.backupHeartbeat +
       ↪ updatedAt >= block.timestamp) && answer > 0) {
        return (uint256(_processPriceRange(answer, oracle)), oracle.
           ↪ backupAggregator.decimals());
    }
```

CVF-157 INFO

- **Category** Procedural
- **Source** TermMaxFactoryV2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
 7 import {MarketInitialParams} from "../../v1/storage/TermMaxStorage.
      ↪ sol";
   import {FactoryErrors} from "../../v1/errors/FactoryErrors.sol";
   import {FactoryEvents} from "../../v1/events/FactoryEvents.sol";
10 import {ITermMaxMarket} from "../../v1/ITermMaxMarket.sol";
   import {ITermMaxFactory} from "../../v1/factory/ITermMaxFactory.sol"
      ↪ ;
```

CVF-158 INFO

- **Category** Bad datatype
- **Source** TermMaxFactoryV2.sol

Recommendation The type for this variable should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
27 address public immutable TERMMAX_MARKET_IMPLEMENTATION;
```

CVF-159 INFO

- **Category** Bad datatype
- **Source** TermMaxFactoryV2.sol

Recommendation The value type for this mapping should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
33 mapping(bytes32 => address) public gtImplements;
```

CVF-160 INFO

- **Category** Bad datatype
- **Source** TermMaxFactoryV2.sol

Recommendation The type for the “TERMMAX_MARKET_IMPLEMENTATION_” argument should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
42 constructor(address admin, address TERMMAX_MARKET_IMPLEMENTATION_)
   ↪ Ownable(admin) {
```

CVF-161 INFO

- **Category** Bad datatype
- **Source** TermMaxFactoryV2.sol

Recommendation The type for the “gtImplement” argument should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
60 function setGtImplement(string memory gtImplementName, address
   ↪ gtImplement) external onlyOwner {
```

CVF-162 INFO

- **Category** Bad datatype
- **Source** TermMaxFactoryV2.sol

Recommendation The type for this argument should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
80 address debtToken,
```

CVF-163 INFO

- **Category** Bad datatype
- **Source** TermMaxFactoryV2.sol

Recommendation The return type should be ”ITermMaxMarket’.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
83 ) external view returns (address market) {
```

```
103     returns (address market)
```

CVF-164 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The type for the "_assetPriceFeed" interface should be "AggregatorV3Interface".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
25 function createPriceFeedWithERC4626(address _assetPriceFeed, address
   ↪  _vault) external returns (address) {
```

CVF-165 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The type for the "_value" argument should be "IERC4626".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
25 function createPriceFeedWithERC4626(address _assetPriceFeed, address
   ↪  _vault) external returns (address) {
```

CVF-166 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The return type should be "TermMaxERC4626PriceFeed".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
25 function createPriceFeedWithERC4626(address _assetPriceFeed, address
   ↪  _vault) external returns (address) {
```

CVF-167 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The type for these arguments should be “AggregatorV3Interface”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
42 address _aTokenToBTokenPriceFeed,
   address _bTokenToCTokenPriceFeed,
```

CVF-168 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The type for this argument should be “IERC20”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
44 address _asset
```

CVF-169 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The return type should be “TermMaxPriceFeedConverter”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
45 ) external returns (address) {
```

CVF-170 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The type for the "_pendlePYLpOracle" argument should be "PendlePYLpOracle".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
64 function createPTWithPriceFeed(address _pendlePYLpOracle, address
   ↪ _market, uint32 _duration, address _priceFeed)
```

CVF-171 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The type for the "_market" argument should be "IPMarket".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
64 function createPTWithPriceFeed(address _pendlePYLpOracle, address
   ↪ _market, uint32 _duration, address _priceFeed)
```

CVF-172 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The type for the "_priceFeed" argument should be "AggregatorV3Interface".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
64 function createPTWithPriceFeed(address _pendlePYLpOracle, address
   ↪ _market, uint32 _duration, address _priceFeed)
```

CVF-173 INFO

- **Category** Bad datatype
- **Source** TermMaxPriceFeedFactoryV2.sol

Recommendation The return type should be “TermMaxPTPriceFeed”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
66 returns (address)
```

CVF-174 INFO

- **Category** Bad naming
- **Source** IAaveV3Minimal.sol

Description This interface looks seems to be a reduced version of the IPool interface form Aave V3, however, this is unclear from the interface name.

Recommendation Rename to “IAaveV3PoolMinimal” and/or clearly explain in a documentation comment what particular Aave interface this interface is intended to minimize.

Client Comment *This contract is out of scope, it is still under development.*

```
4 interface IAaveV3Minimal {
```

CVF-175 INFO

- **Category** Bad datatype
- **Source** IAaveV3Minimal.sol

Recommendation The type for the asset arguments should be more specific.

Client Comment *This contract is out of scope, it is still under development.*

```
67 function getReserveData(address asset) external view returns (
   ↪ ReserveData memory);
```

```
80 function supply(address asset, uint256 amount, address onBehalfOf,
   ↪ uint16 referralCode) external;
```

```
93 function withdraw(address asset, uint256 amount, address to)
   ↪ external returns (uint256);
```

CVF-176 INFO

- **Category** Bad naming
- **Source** GearingTokenEventsV2.sol

Recommendation Events are usually named via nouns, such as “GearingToken”.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
18 event GearingTokenInitialized(address indexed market, string name,
   ↪ string symbol, bytes initialData);
```

CVF-177 INFO

- **Category** Bad datatype
- **Source** GearingTokenEventsV2.sol

Recommendation The type for the “market” parameter should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
18 event GearingTokenInitialized(address indexed market, string name,
   ↪ string symbol, bytes initialData);
```

CVF-178 INFO

- **Category** Bad naming
- **Source** TermMaxTokenEvents.sol

Recommendation Events are usually named via nouns, such as “TermMaxToken”.

Client Comment *This contract is out of scope, it is still under development.*

```
17 event TermMaxTokenInitialized(address indexed admin, address indexed
   ↪  underlying);
```

CVF-179 INFO

- **Category** Suboptimal
- **Source** TermMaxTokenErrors.sol

Recommendation These errors could be made more useful by adding certain parameters into them.

Client Comment *This contract is out of scope, it is still under development.*

```
10 error InvalidToken();
```

```
32 error TimelockNotElapsed();
```

```
35 error AlreadyPending();
```

```
38 error InvalidImplementation();
```

CVF-180 INFO

- **Category** Procedural
- **Source** AccessManagerV2.sol

Description We didn't review this file.

Client Comment *They were audit in other competition.*

```
4 import "../../v1/access/AccessManager.sol";
```

CVF-181 INFO

- **Category** Procedural
- **Source** ITermMaxMarketV2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
4 import {OrderConfig} from "../v1/storage/TermMaxStorage.sol";
  import {ITermMaxOrder} from "../v1/ITermMaxOrder.sol";
```

CVF-182 INFO

- **Category** Procedural
- **Source** TermMaxOrderV2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
11 import {ITermMaxOrder, IERC20} from "../v1/ITermMaxOrder.sol";
   import {ITermMaxMarket} from "../v1/ITermMaxMarket.sol";
   import {IGearingToken} from "../v1/tokens/IGearingToken.sol";
   import {Constants} from "../v1/lib/Constants.sol";
   import {TermMaxCurve, MathLib} from "../v1/lib/TermMaxCurve.sol";
   import {OrderErrors} from "../v1/errors/OrderErrors.sol";
   import {OrderEvents} from "../v1/events/OrderEvents.sol";
   import {OrderConfig, MarketConfig, CurveCuts, CurveCut, FeeConfig}
      ↪ from "../v1/storage/TermMaxStorage.sol";
   import {ISwapCallback} from "../v1/ISwapCallback.sol";
20 import {TransferUtils} from "../v1/lib/TransferUtils.sol";
```

CVF-183 INFO

- **Category** Suboptimal
- **Source** TermMaxOrderV2.sol

Description Solidity compiler is smart enough to precompute constant hash expressions.

Recommendation Use hash expressions instead of hard-coded hashes.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
55 // keccak256(abi.encode(uint256(keccak256("termmax.tsstorage.order.
      ↪ ftReserve")) - 1)) & ~bytes32(uint256(0xff))
   uint256 private constant T_FT_RESERVE_STORE =
   0x2a12e4f8e6ef46c978fd57eac04c67eb8dcdc9f0eec6327794e0ab372ed36000;
   // keccak256(abi.encode(uint256(keccak256("termmax.tsstorage.order.
      ↪ xtReserve")) - 1)) & ~bytes32(uint256(0xff))
   uint256 private constant T_XT_RESERVE_STORE =
60 0x2688c417e2e5b2d9eeee9f203d038d530482d06e29ca0badc05b91bdc9593000;
```

CVF-184 FIXED

- **Category** Bad naming
- **Source** TermMaxOrderV2.sol

Description The names sound like getters, not modifies.

Recommendation Rename to something like “onlyBorrowingIsAllowed”.

```
85  modifier isBorrowingAllowed(OrderConfig memory config) {
```

```
93  modifier isLendingAllowed(OrderConfig memory config) {
```

```
101 modifier isOpen() {
```

CVF-185 FIXED

- **Category** Bad naming
- **Source** TermMaxOrderV2.sol

Description The semantics of these arguments is unclear.

Recommendation Give descriptive names to the arguments. At least in comments.

```
126 address,
    IERC20[3] memory,
    IGearingToken,
    uint256,
130 ISwapCallback,
    CurveCuts memory,
    MarketConfig memory
```

CVF-186 INFO

- **Category** Bad naming
- **Source** TermMaxOrderV2.sol

Description The semantics of the returned values is unclear.

Recommendation Give descriptive names to the returned values and/or describe in the documentation comment.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
176 function tokenReserves() public view override returns (uint256,
       ↪ uint256) {
```

CVF-187 INFO

- **Category** Procedural
- **Source** TermMaxOrderV2.sol

Recommendation Brackets around multiplication are redundant.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
196         ((lendVFtReserve * Constants.DECIMAL_BASE * Constants.
               ↪ DAYS_IN_YEAR) / (lendVXtReserve * daysToMaturity));
```

```
207         (borrowVFtReserve * Constants.DECIMAL_BASE * Constants.
               ↪ DAYS_IN_YEAR)
                / (borrowVXtReserve * daysToMaturity)
```

```
473 feeAmt = (tokenAmtOut * (Constants.DECIMAL_BASE + uint256(feeConfig.
       ↪ borrowMakerFeeRatio))) / nif - tokenAmtOut;
```

```
487 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
524 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
537 feeAmt = (debtTokenAmtOut * (Constants.DECIMAL_BASE + uint256(
       ↪ feeConfig.borrowMakerFeeRatio))) / nif
```

```
660 feeAmt = (negDeltaFt * (Constants.DECIMAL_BASE + uint256(feeConfig.
       ↪ borrowMakerFeeRatio))) / nif - negDeltaFt;
```

```
674 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
742 feeAmt = deltaFt - (deltaFt * (Constants.DECIMAL_BASE - uint256(
       ↪ feeConfig.lendMakerFeeRatio))) / nif;
```

```
759 feeAmt = (negDeltaFt * (Constants.DECIMAL_BASE + uint256(feeConfig.
       ↪ borrowMakerFeeRatio))) / nif - negDeltaFt;
```

```
770 uint256 debtAmtToIssue = ((targetFtReserve - ftReserve) * Constants.
       ↪ DECIMAL_BASE)
        / (Constants.DECIMAL_BASE - market.mintGtFeeRatio());
```

CVF-188 INFO

- **Category** Suboptimal
- **Source** TermMaxOrderV2.sol

Description The references to “newCurveCuts.lendCurveCuts[i]∏ and “newCurve-Cuts.lendCurveCuts[i - 1]” are calculated several times.

Recommendation Calculate once and reuse.

Client Comment *Thanks for the suggestion, we may improve it in the future but not now.*

```
257 newCurveCuts.lendCurveCuts[i].liqSquare == 0
        || newCurveCuts.lendCurveCuts[i].xtReserve <= newCurveCuts.
           ↪ lendCurveCuts[i - 1].xtReserve

267 newCurveCuts.lendCurveCuts[i].liqSquare

269         newCurveCuts.lendCurveCuts[i - 1].liqSquare

273                             newCurveCuts.lendCurveCuts[i].xtReserve.
                                   ↪ plusInt256(
                                    newCurveCuts.lendCurveCuts[i].offset

279                             newCurveCuts.lendCurveCuts[i].xtReserve.
                                   ↪ plusInt256(
280                                 newCurveCuts.lendCurveCuts[i - 1].offset
```

CVF-189 FIXED

- **Category** Procedural
- **Source** TermMaxOrderV2.sol

Recommendation Some brackets are redundant.

```
!= (
    newCurveCuts.lendCurveCuts[i - 1].liqSquare
        * (
            (
                (
                    newCurveCuts.lendCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.lendCurveCuts[i].offset
                    )
                ) ** 2 * Constants.DECIMAL_BASE
            )
                / (
                    newCurveCuts.lendCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.lendCurveCuts[i - 1].offset
                    ) ** 2
                )
        )
) / Constants.DECIMAL_BASE
```

```
!= (
    newCurveCuts.borrowCurveCuts[i - 1].liqSquare
        * (
            (
                (
                    newCurveCuts.borrowCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.borrowCurveCuts[i].offset
                    )
                ) ** 2 * Constants.DECIMAL_BASE
            )
                / (
                    newCurveCuts.borrowCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.borrowCurveCuts[i - 1].offset
                    ) ** 2
                )
        )
) / Constants.DECIMAL_BASE
```

CVF-190 INFO

- **Category** Suboptimal
- **Source** TermMaxOrderV2.sol

Description the “Constants.DECIMAL_BASE” value is in both, numerator and denominator, so it can be reduced.

Recommendation Reduce the formula.

Client Comment *It is used to improve calculation accuracy.*

```
    newCurveCuts.lendCurveCuts[i - 1].liqSquare
        * (
            (
                (
                    newCurveCuts.lendCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.lendCurveCuts[i].offset
                    )
                ) ** 2 * Constants.DECIMAL_BASE
            )
                / (
                    newCurveCuts.lendCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.lendCurveCuts[i - 1].offset
                    ) ** 2
                )
        )
) / Constants.DECIMAL_BASE
```

```
!= (
    newCurveCuts.borrowCurveCuts[i - 1].liqSquare
        * (
            (
                (
                    newCurveCuts.borrowCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.borrowCurveCuts[i].offset
                    )
                ) ** 2 * Constants.DECIMAL_BASE
            )
                / (
                    newCurveCuts.borrowCurveCuts[i].xtReserve.
                       ↪ plusInt256(
                        newCurveCuts.borrowCurveCuts[i - 1].offset
                    ) ** 2
                )
        )
) / Constants.DECIMAL_BASE
```

CVF-191 INFO

- **Category** Suboptimal
- **Source** TermMaxOrderV2.sol

Description The references to “newCurveCuts.borowCurveCut[i]” and “newCurveCuts.borrowCurveCuts[i - 1]” are calculated several times.

Recommendation Calculate once and reuse.

Client Comment *Thanks for the suggestion, we may improve it in the future but not now.*

294
```
newCurveCuts.borrowCurveCuts[i].liqSquare == 0
    || newCurveCuts.borrowCurveCuts[i].xtReserve <= newCurveCuts.
       ↪ borrowCurveCuts[i - 1].xtReserve
```

300
```
newCurveCuts.borrowCurveCuts[i].liqSquare
```

302
```
        newCurveCuts.borrowCurveCuts[i - 1].liqSquare
```

306
```
                        newCurveCuts.borrowCurveCuts[i].xtReserve.
                           ↪ plusInt256(
                            newCurveCuts.borrowCurveCuts[i].offset
```

312
```
                        newCurveCuts.borrowCurveCuts[i].xtReserve.
                           ↪ plusInt256(
                            newCurveCuts.borrowCurveCuts[i - 1].
                               ↪ offset
```

CVF-192 INFO

- **Category** Documentation
- **Source** TermMaxOrderV2.sol

Description The semantics of the arguments and the return values is unclear.

Recommendation Add argument names and return value names as comments. Also, explain the “func” semantics in a documentation comment.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
439 function(uint, uint, uint, OrderConfig memory) internal view returns
    ↪  (uint, uint, IERC20) func
```

```
497 function(uint, uint, uint, OrderConfig memory) internal view returns
    ↪  (uint, uint, IERC20) func
```

```
626 function(uint, uint, uint, OrderConfig memory) internal view returns
    ↪  (uint, uint, IERC20) func
```

```
709 function(uint, uint, uint, OrderConfig memory) internal view returns
    ↪  (uint, uint, IERC20) func
```

CVF-193 INFO

- **Category** Bad naming
- **Source** TermMaxOrderV2.sol

Description The semantics of the returned values is unclear.

Recommendation Give descriptive names to the returned values and/or explain in a documentation comment.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
440 ) internal returns (uint256, uint256) {
```

```
498 ) internal returns (uint256, uint256) {
```

```
627 ) internal returns (uint256, uint256) {
```

```
710 ) internal returns (uint256, uint256) {
```

CVF-194 INFO

- **Category** Procedural
- **Source** TermMaxMarketV2.sol

Description We didn't review these files.

Client Comment *They were audit in other competition.*

```
12 import {IGearingToken} from "../v1/tokens/IGearingToken.sol";
   import {IFlashLoanReceiver} from "../v1/IFlashLoanReceiver.sol";
   import {ITermMaxOrder} from "../v1/ITermMaxOrder.sol";
   import {Constants} from "../v1/lib/Constants.sol";
```

```
17 import {MarketErrors} from "../v1/errors/MarketErrors.sol";
   import {MarketEvents} from "../v1/events/MarketEvents.sol";
   import {StringUtil} from "../v1/lib/StringUtil.sol";
```

```
27 } from "../v1/storage/TermMaxStorage.sol";
   import {ISwapCallback} from "../v1/ISwapCallback.sol";
   import {TransferUtils} from "../v1/lib/TransferUtils.sol";
30 import {ITermMaxMarket, IMintableERC20, IERC20} from "../v1/
      ↪ ITermMaxMarket.sol";
```

CVF-195 INFO

- **Category** Bad datatype
- **Source** TermMaxMarketV2.sol

Recommendation The type for these variables should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
52 address immutable MINTABLE_ERC20_IMPLEMENT;
   address immutable TERMMAX_ORDER_IMPLEMENT;
```

```
56 address private collateral;
```

CVF-196 INFO

- **Category** Bad datatype
- **Source** TermMaxMarketV2.sol

Recommendation The type for these variable should be "IMintableERC20V2".

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
58 IMintableERC20 private ft;
   IMintableERC20 private xt;
```

CVF-197 FIXED

- **Category** Procedural
- **Source** TermMaxMarketV2.sol

Recommendation Brackets around multiplication are redundant.

```
80  return (daysToMaturity * uint256(_config.feeConfig.mintGtFeeRatio) *
        ↪  uint256(_config.feeConfig.mintGtFeeRef))
```

```
308 uint128 issueFee = ((debt * mintGtFeeRatio()) / Constants.
        ↪ DECIMAL_BASE).toUint128();
```

```
338 uint128 issueFee = ((debt * mintGtFeeRatio()) / Constants.
        ↪ DECIMAL_BASE).toUint128();
```

```
367 uint256 proportion = (ftAmount * Constants.DECIMAL_BASE_SQ) / (ft.
        ↪ totalSupply() - ft.balanceOf(address(this)));
```

```
419 uint256 proportion = (ftAmount * Constants.DECIMAL_BASE_SQ) / ft.
        ↪ totalSupply();
```

CVF-198 INFO

- **Category** Bad naming
- **Source** TermMaxMarketV2.sol

Description The semantics of the returned values is unclear.

Recommendation Give descriptive names to the returned values and/or explain in the documentation comment.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
156 returns (IMintableERC20, IMintableERC20, IGearingToken, address,
        ↪ IERC20)
```

CVF-199 INFO

- **Category** Bad datatype
- **Source** TermMaxMarketV2.sol

Recommendation The type for the fourth returned value should be more specific.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
156 returns (IMintableERC20, IMintableERC20, IGearingToken, address,
    ↪ IERC20)
```

CVF-200 FIXED

- **Category** Procedural
- **Source** TermMaxMarketV2.sol

Recommendation This check should be performed earlier. Before any state updates.

```
170 _checkFee(newConfig.feeConfig);
```

CVF-201 INFO

- **Category** Bad datatype
- **Source** TermMaxMarketV2.sol

Recommendation The value “5 * DECIMAL_BASE” should be a named constant.

Client Comment *Thanks for the suggestion, I thought it is optional.*

```
181 || fee.mintGtFeeRatio >= Constants.MAX_FEE_RATIO || fee.mintGtFeeRef
    ↪  > 5 * Constants.DECIMAL_BASE
```

CVF-202 INFO

- **Category** Procedural
- **Source** TermMaxMarketV2.sol

Recommendation Brackets around multiplication are redundant.

```
273 uint128 debt = ((xtAmt * Constants.DECIMAL_BASE) / (Constants.
    ↪ DECIMAL_BASE - mintGtFeeRatio())).toUint128();
```

CVF-203 FIXED

- **Category** Suboptimal
- **Source** TermMaxMarketV2.sol

Description The same code is used twice.

Recommendation Refactor to reduce code duplication.

```
359 uint256 liquidationDeadline =
360     gt.liquidatable() ? mConfig.maturity + Constants.
        ↪ LIQUIDATION_WINDOW : mConfig.maturity;
    if (block.timestamp < liquidationDeadline) {
        revert CanNotRedeemBeforeFinalLiquidationDeadline(
        ↪ liquidationDeadline);
    }
```

```
406 uint256 liquidationDeadline =
        gt.liquidatable() ? mConfig.maturity + Constants.
        ↪ LIQUIDATION_WINDOW : mConfig.maturity;
    if (block.timestamp < liquidationDeadline) {
        revert CanNotRedeemBeforeFinalLiquidationDeadline(
        ↪ liquidationDeadline);
410 }
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting