Report

v. 1.0

Customer

Term Structure Labs

Smart Contract Audit

TermMax. Phase II

7th July 2025

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	07.07.25	A. Zveryanskaya	Initial Draft
0.2	07.07.25	A. Zveryanskaya	Minor revision
1.0	07.07.25	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.

Term Structure is a decentralized bond protocol powered by zkTrue-up, a platform that enables peer-to-peer lending and borrowing with fixed interest rates.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	ITermMaxMarketV2.sol	TermMaxMarketV2.sol	TermMaxOrderV2.sol
events/			
	VaultEventsV2.sol	FactoryEventsV2.sol	GearingToken EventsV2.sol
	TermMaxToken Events.sol		
lib/			
	TransactionReentrancy Guard.sol	MarketConstantsV2.sol	
storage/			
	TermMaxStorageV2.sol		
vault/			
	ITermMaxVaultV2.sol	OrderManagerV2.sol	TermMaxVaultV2.sol
	VaultStorageV2.sol		
factory/			
	ITermMaxVault FactoryV2.sol	TermMaxVault FactoryV2.sol	TermMaxFactoryV2.sol
	TermMaxPriceFeed FactoryV2.sol		
access/			
	AccessManagerV2.sol		
extensions/aave/			
	IAaveV3Minimal.sol		

errors/			
	VaultErrorsV2.sol	TermMaxTokenErrors.sol	
oracle/			
	IOracleV2.sol	OracleAggregatorV2.sol	
oracle/priceFeeds/			
	ITermMaxPriceFeed.sol	TermMax ERC4626PriceFeed.sol	TermMaxPriceFeed Converter.sol
	TermMax PTPriceFeed.sol		
router/			
	ITermMaxRouterV2.sol	TermMaxRouterV2.sol	
router/swapAdapters/			
	TermMax SwapAdapter.sol	TermMax TokenAdapter.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 provided the client with some recommendations.

	Info	Fixed
Moderate	4	10
Minor	39	15

6 Moderate Issues

CVF-3 INFO

- **Category** Suboptimal
- **Source** TransactionReentrancyGuard.sol

Description These modifiers don't clear the flag, so they not only prevent reentrancy, but also prevent several sequential calls in one transaction.

Recommendation Allow sequential calls in one transaction as they are harmless and could reduce gas consumption.

Client Comment *Our goal is to restrict deposits and withdrawals from being made simultaneously in one transaction, to prevent attackers from manipulating prices through flash loans to make profits. You can still make multiple deposits or multiple withdrawals.*

```
16 modifier nonTxReentrant() {
```

```
22 modifier nonTxReentrantBetweenActions(uint256 actionId) {
```

CVF-4 FIXED

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

Description There is no check to ensure these arrays are of the same length. If some array is longer than orders", the extra elements are silently ignored.

Recommendation Implement an appropriate length check.

```
54 ITermMaxOrder[] memory orders,
   int256[] memory changes,
   uint256[] memory maxSupplies,
   CurveCuts[] memory curveCuts
```

CVF-5 FIXED

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

Description This check is redundant, as it will anyway be performed inside the "safeTransfer" call. Also, this check doesn't guarantee successful transfer.

Recommendation Remove this check.

177
```
uint256 assetBalance = asset.balanceOf(address(this));
if (assetBalance >= amount) {
```

180
```
} else {
    revert InsufficientFunds(assetBalance, amount);
```

CVF-6 INFO

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

Description Using annualized interest is inefficient.

Recommendation Use per second interest.

Client Comment *Thanks for your suggestion, we do not consider changing the design of this part. Because the interest per second also needs to be enlarged to reduce the loss of accuracy.*

240
```
uint256 interest = (_annualizedInterest * (endTime - startTime)) /
    ↪ 365 days;
```

326
```
    uint256 deltaAnnualizedInterest = ftChanges * 365 days / uint256
        ↪ (maturity - block.timestamp);
```

334
```
    uint256 deltaAnnualizedInterest = (ftChanges * 365 days) /
        ↪ uint256(maturity - block.timestamp);
```

CVF-7 FIXED

- **Category** Bad datatype
- **Source** TermMaxVaultV2.sol

Recommendation Usually, the “bytes32” type is used for hashes.

```
58 uint256 private constant ACTION_DEPOSIT = uint256(keccak256("
      ↪ ACTION_DEPOSIT"));
   uint256 private constant ACTION_WITHDRAW = uint256(keccak256("
      ↪ ACTION_WITHDRAW"));
```

CVF-8 FIXED

- **Category** Bad naming
- **Source** TermMaxVaultV2.sol

Description The modifier name looks like a precondition checker, while actually the modifier has a side effect and in some cases may modify the storage.

Recommendation Rename to modifier to make the side effect clear, or remove the side effect.

```
84 modifier marketIsWhitelisted(address market) {
```

```
86         _marketWhitelist[market] = true;
```

CVF-9 FIXED

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Recommendation The “SetMarketWhitelist” event should be emitted here.

```
86 _marketWhitelist[market] = true;
```

CVF-10 FIXED

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Recommendation Unchained initializers should be used here.

```
109 __ERC20_init(params.name, params.symbol);
110 __Ownable_init(params.admin);
__ERC4626_init(params.asset);
__ReentrancyGuard_init();
__Pausable_init();
```

CVF-11 FIXED

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Description There are no range checks for these values.

Recommendation Add appropriate range checks.

```
115 _setPerformanceFeeRate(params.performanceFeeRate);
```

```
118 _setMinApy(params.minApy);
_setMinIdleFundRate(params.minIdleFundRate);
```

```
121 _setCapacity(params.maxCapacity);
```

CVF-12 FIXED

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Description The value returned here may be inconsistent with the value actually used in the “marketIsWhitelisted” modifier, as the modifier in some cases updates the store whitelist flag for a market.

Recommendation Make this getter consistent with the modifier.

```
160 function marketWhitelist(address market) external view virtual
    ↪ returns (bool) {
     return _marketWhitelist[market];
```

CVF-13 INFO

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Description The message sender logging along with event here could be anybody, as the “acceptPending...” function don’t have any access control.

Recommendation Remove the message sender form event and use the address of the original submitter.

Client Comment *Thank you for your suggestion, we will not modify this section for the time being.*

```
356 emit VaultEventsV2.SetMinApy(_msgSender(), newMinApy);
```

```
362 emit VaultEventsV2.SetMinIdleFundRate(_msgSender(),
      ↪ newMinIdleFundRate);
```

CVF-14 FIXED

- **Category** Bad naming
- **Source** TermMaxVaultV2.sol

Description The error name is misleading, as the function burns shares, rather than mints anything.

Recommendation Use different error.

```
511 revert ERC4626ExceededMaxMint(recipient, shares, maxShares);
```

CVF-15 INFO

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Description Annualized interest is inefficient.

Recommendation Consider using per-second interest.

Client Comment *Thank you for your suggestion, we will not modify this section for the time being.*

```
813 {
```

CVF-16 FIXED

- **Category** Procedural
- **Source** VaultErrorsV2.sol

Description Four different errors have the same comment, and the comment doesn't seem relevant to either of the errors.

Recommendation Accurately describe each error with a comment.

```
11 /// @notice Error thrown when an invalid or unsupported functions is
      ↪ called
   error UseVaultInitialParamsV2();
   /// @notice Error thrown when an invalid or unsupported functions is
      ↪ called
   error SupplyQueueNoLongerSupported();
   /// @notice Error thrown when an invalid or unsupported functions is
      ↪ called
   error WithdrawalQueueNoLongerSupported();
   /// @notice Error thrown when an invalid or unsupported functions is
      ↪ called
   error UseApyInsteadOfApr();
```

7 Recommendations

CVF-17 INFO

- **Category** Procedural
- **Source** VaultErrorsV2.sol

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

Client Comment *Thanks for your suggestion, I think the operator understands his parameters.*

```
10 error CollateralIsAsset();
```

CVF-18 INFO

- **Category** Procedural
- **Source** FactoryEventsV2.sol

Description We didn't review this file.

Client Comment *This contract is just a storage structure.*

```
5 import {MarketInitialParams} from "../../v1/storage/TermMaxStorage.
    ↪ sol";
```

CVF-19 FIXED

- **Category** Bad naming
- **Source** FactoryEventsV2.sol

Recommendation Events are usually named via nouns, such as "Market", "Vault", or "PriceFeed".

```
20 event CreateMarket(
```

```
30 event CreateVault(address indexed vault, address indexed creator,
    ↪ VaultInitialParamsV2 initialParams);
```

```
36 event PriceFeedCreated(address indexed priceFeed);
```

CVF-20 INFO

- **Category** Bad datatype
- **Source** FactoryEventsV2.sol

Recommendation The type for address parameters should be more specific.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
21 address indexed market, address indexed collateral, IERC20 indexed
   ↪ debtToken, MarketInitialParams params
```

CVF-21 INFO

- **Category** Bad datatype
- **Source** FactoryEventsV2.sol

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

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
30 event CreateVault(address indexed vault, address indexed creator,
   ↪ VaultInitialParamsV2 initialParams);
```

CVF-22 INFO

- **Category** Bad datatype
- **Source** FactoryEventsV2.sol

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

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
36 event PriceFeedCreated(address indexed priceFeed);
```

CVF-23 INFO

- **Category** Procedural
- **Source** ITermMaxVaultFactoryV2.sol

Description UPPER_CASE names are conventionally used for constants.

Recommendation Rename using camelCase.

Client Comment *Thanks for your suggestion, we will not consider modifying it for now because it is a static value.*

```
15 function TERMMAX_VAULT_IMPLEMENTATION() external view returns (
      ↪ address);
```

CVF-24 INFO

- **Category** Bad datatype
- **Source** ITermMaxVaultFactoryV2.sol

Recommendation The return type should be more specific.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
15 function TERMMAX_VAULT_IMPLEMENTATION() external view returns (
      ↪ address);
```

CVF-25 INFO

- **Category** Bad datatype
- **Source** ITermMaxVaultFactoryV2.sol

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

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
28 address asset,
```

CVF-26 INFO

- **Category** Bad datatype
- **Source** ITermMaxVaultFactoryV2.sol

Recommendation The return type should be more specific.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
32 ) external view returns (address vault);
```

CVF-27 INFO

- **Category** Bad datatype
- **Source** ITermMaxVaultFactoryV2.sol

Recommendation The return type should be more specific.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
40 function createVault(VaultInitialParamsV2 memory initialParams,
     ↪ uint256 salt) external returns (address);
```

CVF-28 INFO

- **Category** Procedural
- **Source** ITermMaxVaultV2.sol

Recommendation Consider specifying as "^0.8.0" unless there is something special regarding this particular version. Also relevant for: OrderManagerV2.sol, TermMaxVaultFactoryV2.sol, TermMaxVaultV2.sol, TransactionReentrancyGuard.sol, VaultStorageV2.sol.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

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

CVF-29 INFO

- **Category** Procedural
- **Source** ITermMaxVaultV2.sol

Description We didn't review this file.

Client Comment *This library is referenced from the morpho library.*

```
4 import {PendingAddress, PendingUint192} from "../../v1/lib/
    ↪ PendingLib.sol";
```

CVF-30 FIXED

- **Category** Documentation
- **Source** ITermMaxVaultV2.sol

Description The number format for the returned value is unclear.

Recommendation Explain in a documentation comment.

```
26 function apy() external view returns (uint256);
```

```
33 function minApy() external view returns (uint64);
```

```
40 function minIdleFundRate() external view returns (uint64);
```

```
47 function pendingMinApy() external view returns (PendingUint192
    ↪ memory);
```

```
54 function pendingMinIdleFundRate() external view returns (
    ↪ PendingUint192 memory);
```

CVF-31 FIXED

- **Category** Documentation
- **Source** ITermMaxVaultV2.sol

Description The number format of the arguments is unclear.

Recommendation Explain in a documentation comment.

```
61 function submitPendingMinApy(uint64 newMinApy) external;
```

```
68 function submitPendingMinIdleFundRate(uint64 newMinIdleFundRate)
      ↪ external;
```

CVF-33 INFO

- **Category** Procedural
- **Source** OrderManagerV2.sol

Description We didn't review these files.

Client Comment *These contracts have been audited by Cantina. The reports is here https://github.com/term-structure/audits/blob/main/TermMax/TermMax-Cantina-competition-20250320.pdf*

```
 6 import {ITermMaxMarket} from "../../v1/ITermMaxMarket.sol";
   import {CurveCuts, OrderConfig} from "../../v1/storage/
      ↪ TermMaxStorage.sol";
   import {VaultErrors} from "../../v1/errors/VaultErrors.sol";
   import {VaultEvents} from "../../v1/events/VaultEvents.sol";
10 import {ITermMaxOrder} from "../../v1/ITermMaxOrder.sol";
   import {TransferUtils} from "../../v1/lib/TransferUtils.sol";
   import {Constants} from "../../v1/lib/Constants.sol";
   import {ArrayUtils} from "../../v1/lib/ArrayUtils.sol";
   import {MathLib} from "../../v1/lib/MathLib.sol";
   import {LinkedList} from "../../v1/lib/LinkedList.sol";
   import {IOrderManager} from "../../v1/vault/IOrderManager.sol";
   import {ISwapCallback} from "../../v1/ISwapCallback.sol";
```

CVF-34 FIXED

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

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

```
54 ITermMaxOrder[] memory orders,
   int256[] memory changes,
   uint256[] memory maxSupplies,
   CurveCuts[] memory curveCuts
```

CVF-35 FIXED

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

Description This check makes the “asset” argument redundant.

Recommendation Consider removing the “asset” argument.

```
98 if (asset != debtToken) revert InconsistentAsset();
```

CVF-36 FIXED

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

Description The case when “changes” is exactly zero isn’t handled.

Recommendation Handle zero changes case separately or clearly explain, why it shouldn’t be handled.

```
128 if (changes < 0) {
```

```
135 } else {
        // deposit assets to order
```

CVF-37 INFO

- **Category** Bad datatype
- **Source** OrderManagerV2.sol

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

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
202 function dealBadDebt(address recipient, address collateral, uint256
    ↪ amount)
```

CVF-38 INFO

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

Description This check is partially superseded by the “amount > badDebtAmt” check.

Recommendation Remove this check and replace it with “amount > 0” requirement.

Client Comment *We do not consider removing it because if bad Debt is 0, then there may be a division by zero problem.*

```
209 if (badDebtAmt == 0) revert NoBadDebt(collateral);
```

CVF-39 INFO

- **Category** Procedural
- **Source** OrderManagerV2.sol

Recommendation Brackets are redundant.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
212 collateralOut = (amount * collateralBalance) / badDebtAmt;
```

CVF-40 INFO

- **Category** Procedural
- **Source** OrderManagerV2.sol

Recommendation Brackets areound multiplication are redundant.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
240 uint256 interest = (_annualizedInterest * (endTime - startTime)) /
      ↪ 365 days;
    uint256 _performanceFeeToCurator = (interest * _performanceFeeRate)
      ↪ / Constants.DECIMAL_BASE;
```

CVF-41 INFO

- **Category** Procedural
- **Source** OrderManagerV2.sol

Recommendation Brackets are redundant.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
244 _accretingPrincipal += (interest - _performanceFeeToCurator);
```

CVF-42 INFO

- **Category** Suboptimal
- **Source** OrderManagerV2.sol

Description Updating interest incrementally could accumulate rounding errors.

Recommendation Consider using per-second interest to avoid rounding.

Client Comment *Thanks for your suggestion, we do not consider changing the design of this part. Because the interest per second also needs to be enlarged to reduce the loss of accuracy.*

```
326 uint256 deltaAnnualizedInterest = ftChanges * 365 days / uint256(
      ↪ maturity - block.timestamp);

328 _maturityToInterest[maturity] += deltaAnnualizedInterest;

334 uint256 deltaAnnualizedInterest = (ftChanges * 365 days) / uint256(
      ↪ maturity - block.timestamp);

340 _maturityToInterest[maturity] -= deltaAnnualizedInterest;
```

CVF-43 FIXED

- **Category** Documentation
- **Source** TermMaxStorageV2.sol

Description The number format for these fields is unclear.

Recommendation Explain in a documentation comment.

```
15 uint64 performanceFeeRate;
   uint64 minApy;
   uint64 minIdleFundRate;
```

CVF-45 INFO

- **Category** Procedural
- **Source** TermMaxVaultFactoryV2.sol

Description We didn't review this file.

Client Comment *It only contains the definition of error*

```
5 import {FactoryErrors} from "../../v1/errors/FactoryErrors.sol";
```

CVF-46 INFO

- **Category** Procedural
- **Source** TermMaxVaultFactoryV2.sol

Description UPPER_CASE names are conventionally used for constants.

Recommendation Rename in camelCase.

Client Comment *Thanks for your suggestion, we will not consider modifying it for now because it is a static value.*

```
19 address public immutable TERMMAX_VAULT_IMPLEMENTATION;
```

```
21 constructor(address TERMMAX_VAULT_IMPLEMENTATION_) {
```

CVF-47 INFO

- **Category** Bad datatype
- **Source** TermMaxVaultFactoryV2.sol

Recommendation The type for this variable should be more specific.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
19 address public immutable TERMMAX_VAULT_IMPLEMENTATION;
```

CVF-48 INFO

- **Category** Bad datatype
- **Source** TermMaxVaultFactoryV2.sol

Recommendation The argument type should be more specific.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
21 constructor(address TERMMAX_VAULT_IMPLEMENTATION_) {
```

CVF-49 FIXED

- **Category** Suboptimal
- **Source** TermMaxVaultFactoryV2.sol

Description This check is redundant, as it is anyway possible to pass a dead address.

Recommendation Remove this check.

```
22 if (TERMMAX_VAULT_IMPLEMENTATION_ == address(0)) {
```

CVF-50 INFO

- **Category** Bad datatype
- **Source** TermMaxVaultFactoryV2.sol

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

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
33 address asset,
```

CVF-51 INFO

- **Category** Bad datatype
- **Source** TermMaxVaultFactoryV2.sol

Recommendation The return type should be “ITermMaxVaultV2”.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
37 ) external view returns (address vault) {
```

CVF-52 INFO

- **Category** Bad datatype
- **Source** TermMaxVaultFactoryV2.sol

Recommendation The return type should be “ITermMaxVaultV2”.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
46 function createVault(VaultInitialParamsV2 memory initialParams,
      ↪ uint256 salt) public returns (address vault) {
```

CVF-54 INFO

- **Category** Procedural
- **Source** TermMaxVaultV2.sol

Description We didn't review these files.

Client Comment *These contracts have been audited by Cantina. The reports is here https://github.com/term-structure/audits/blob/main/TermMax/TermMax-Cantina-competition-20250320.pdf*

```
20 import {PendingLib, PendingAddress, PendingUint192} from "../../v1/
      ↪ lib/PendingLib.sol";
   import {ITermMaxMarket} from "../../v1/ITermMaxMarket.sol";
   import {VaultInitialParams, CurveCuts} from "../../v1/storage/
      ↪ TermMaxStorage.sol";
```

```
24 import {ITermMaxOrder} from "../../v1/ITermMaxOrder.sol";
   import {VaultConstants} from "../../v1/lib/VaultConstants.sol";
   import {TransferUtils} from "../../v1/lib/TransferUtils.sol";
   import {ISwapCallback} from "../../v1/ISwapCallback.sol";
   import {VaultErrors} from "../../v1/errors/VaultErrors.sol";
   import {VaultEvents} from "../../v1/events/VaultEvents.sol";
```

```
31 import {IOrderManager} from "../../v1/vault/IOrderManager.sol";
```

```
33 import {Constants} from "../../v1/lib/Constants.sol";
   import {ITermMaxVault} from "../../v1/vault/ITermMaxVault.sol";
```

CVF-55 FIXED

- **Category** Flaw
- **Source** TermMaxVaultV2.sol

Recommendation This file should be imported as "./VaultStorageV2.sol".

```
32 import {VaultStorageV2, OrderInfo} from "../../v2/vault/
      ↪ VaultStorageV2.sol";
```

CVF-56 INFO

- **Category** Procedural
- **Source** TermMaxVaultV2.sol

Description UPPER_CASE names are conventionally used for constants, but not for variables.

Recommendation Rename the variable using camelCase.

Client Comment *Thanks for your suggestion, we will not consider modifying it for now because it is a static value.*

```
56 address public immutable ORDER_MANAGER_SINGLETON;
```

CVF-57 INFO

- **Category** Procedural
- **Source** TermMaxVaultV2.sol

Description UPPER_CASE names are conventionally used for constants, but not for arguments.

Recommendation Rename the argument using camelCase.

Client Comment *Thanks for your suggestion, we will not consider modifying it for now because it is a static value.*

```
102 constructor(address ORDER_MANAGER_SINGLETON_) {
```

CVF-58 FIXED

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Recommendation This check is redundant, as it is anyway possible to pass a dead order manager address.

```
103 if (ORDER_MANAGER_SINGLETON_ == address(0)) revert
    ↪ InvalidImplementation();
```

CVF-59 FIXED

- **Category** Unclear behavior
- **Source** TermMaxVaultV2.sol

Description This function should emit some event.

```
129 function _setPerformanceFeeRate(uint64 newPerformanceFeeRate)
       ↪ internal {
```

CVF-60 FIXED

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Description The "_accretingPrincipal" variable is read twice.

Recommendation Read once and reuse.

```
275 if (_accretingPrincipal == 0) return 0;
    return (_annualizedInterest * (Constants.DECIMAL_BASE -
       ↪ _performanceFeeRate)) / (_accretingPrincipal);
```

CVF-61 INFO

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Recommendation It would be more efficient to pass a single array of structs with four fields, rather than four parallel arrays.

Client Comment *Thank you for your suggestion, we are not considering changing this method in this version.*

```
391 ITermMaxOrder[] memory orders,
    int256[] memory changes,
    uint256[] memory maxSupplies,
    CurveCuts[] memory curveCuts
```

CVF-62 INFO

- **Category** Bad datatype
- **Source** TermMaxVaultV2.sol

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

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
500 function dealBadDebt(address collateral, uint256 badDebtAmt, address
    ↪  recipient, address owner)
```

CVF-63 INFO

- **Category** Suboptimal
- **Source** TermMaxVaultV2.sol

Recommendation This check is redundant, as it will anyway be performed inside the “_burn” call.,

Client Comment *The _burn method is derived from the ERC20 contract and does not contain the 'MaxRedeem' check.*

```
509 uint256 maxShares = maxRedeem(owner);
510 if (shares > maxShares) {
```

CVF-64 FIXED

- **Category** Procedural
- **Source** TermMaxVaultV2.sol

Description Unlike other similar cases, here even is emitted outside the setter call.

Recommendation Emit even inside the setter.

```
579 if (newPerformanceFeeRate < _performanceFeeRate) {
580     _setPerformanceFeeRate(uint256(newPerformanceFeeRate).toUint64()
        ↪ );
```

CVF-65 INFO

- **Category** Procedural
- **Source** TermMaxVaultV2.sol

Recommendation Brackets areound multiplication are redundant.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
813 {
        uint256 interest = (previewAnnualizedInterest * (endTime -
            ↪ startTime)) / 365 days;
```

CVF-67 FIXED

- **Category** Documentation
- **Source** TransactionReentrancyGuard.sol

Description It is unclear how the remark in brackets is related to the main comment. Does it mean that “invalid ID” means an attempt to use a reserved ID?

Recommendation Rephrase the comment.

```
9 /// @notice Error thrown when an invalid action ID is provided. (0
    ↪ and 1 are reserved for the guard itself)
```

CVF-68 INFO

- **Category** Suboptimal
- **Source** TransactionReentrancyGuard.sol

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

Recommendation Define the constant with a hash expression, rather than a hardcoded hash value.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
12 // keccak256(abi.encode(uint256(keccak256("termmax.tsstorage.
    ↪ TransactionReentrancyGuard")) - 1)) & ~bytes32(uint256(0xff))
uint256 private constant T_FLAG_STORE =
0x55d65f3b5821c66716708cd5119fc8b654f479bd23b96d0911cee85241904700;
```

CVF-69 INFO

- **Category** Suboptimal
- **Source** TransactionReentrancyGuard.sol

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

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
10 error InvalidActionId();
```

CVF-70 INFO

- **Category** Bad naming
- **Source** VaultErrorsV2.sol

Description These names doesn't sound like errors.

Recommendation Rename these errors.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
12 error UseVaultInitialParamsV2();
```

```
18 error UseApyInsteadOfApr();
```

CVF-71 INFO

- **Category** Bad naming
- **Source** VaultEventsV2.sol

Recommendation Events are usually named via nouns, such as “MinApySubmission” or “MinApy”.

Client Comment *Thanks for your suggestion, we are not considering changing it for now as it is optional.*

```
14 event SubmitMinApy(uint64 newMinApy, uint64 validAt);

21 event SetMinApy(address indexed caller, uint64 newMinApy);

28 event SubmitMinIdleFundRate(uint64 newMinIdleFundRate, uint64
      ↪ validAt);

35 event SetMinIdleFundRate(address indexed caller, uint64
      ↪ newMinIdleFundRate);

41 event RevokePendingMinApy(address indexed caller);

47 event RevokePendingMinIdleFundRate(address indexed caller);

54 event AccruedInterest(uint256 newAccretingPrincipal, uint256
      ↪ newPerformanceFee);
```

CVF-73 INFO

- **Category** Procedural
- **Source** VaultStorageV2.sol

Description We didn’t review these files.

Client Comment *These contracts have been audited by Cantina. The reports is here https://github.com/term-structure/audits/blob/main/TermMax/TermMax-Cantina-competition-20250320.pdf*

```
4 import {PendingAddress, PendingUint192} from "../../v1/lib/
     ↪ PendingLib.sol";
  import {OrderInfo} from "../../v1/vault/VaultStorage.sol";
```

CVF-74 FIXED

- **Category** Bad naming
- **Source** VaultStorageV2.sol

Description The semantics of the keys in this mapping is unclear.

Recommendation Explain in a documentation comment.

```
34 mapping(address => OrderInfo) internal _orderMapping;
```

CVF-75 INFO

- **Category** Suboptimal
- **Source** VaultStorageV2.sol

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

Client Comment *In fact, the '_maturityMapping' is a one-way linked list, represented by a mapping structure, so they cannot be merged.*

```
39 mapping(uint64 => uint64) internal _maturityMapping;
```

```
41 mapping(uint64 => uint256) internal _maturityToInterest;
```

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