CERTIK

MarginSwap

Core Implementation Contracts

Security Assessment

April 6th, 2021

Audited By:

Alex Papageorgiou @ CertiK

alex.papageorgiou@certik.org

Reviewed By:

Camden Smallwood @ CertiK

camden.smallwood@certik.org

Disclaimer

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	MarginSwap - Core Implementation Contracts
Description	A decentralized margin and spot trading exchange.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 4aed9c53c635e502a35ecd1067f33289b1227c83 2. c91d714c4651274cf0e2afde83ac297318c02e84

Audit Summary

Delivery Date	April 6th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	March 19th, 2021 - March 22nd, 2021

Vulnerability Summary

Total Issues	63
● **Total Critical**	1
● **Total Major**	5
● **Total Medium**	6
● **Total Minor**	17
● **Total Informational**	34

Executive Summary

We were tasked with auditing the codebase of MarginSwap and in particular the core contracts that facilitate their borrowing and lending cross-collateral protocol.

The system is novel and is not based on any other commonly utilized implementation. As a result, intricate care was taken to ensure that the action flows within the system conform to the expected sane behavior of the system. The documentation by the MarginSwap team was suboptimal and solely involved contract comments; to this end, we utilized the code-established relationships within the contracts to form our security assumptions and validate the expected behavior of the system.

Overall, the code is heavily unoptimized and boasts high gas costs in execution as many code segments have high complexity. Simplifications can be applied to the codebase and have been pointed out as optional informational exhibits.

Multiple vulnerabilities were pointed out in the codebase that can have severe ramifications to the system, one of which can allow an attacker to fully drain the protocol's funds. We strongly urge the MarginSwap team to promptly remediate all issues pointed out by the report and to enhance the documentation of the system via technical overviews and deployment layouts as the novelty of the code renders it illegible to new readers of the system.

Additionally, the system possesses a single point of failure which we advise be refactored as it can enable a hostile takeover of the full system with a single private key, a trait undesirable for any system that is meant to achieve a high TVL.

Certain concepts have also been misapplied, such as the notion of a Floating Point using `2**32` as the divisor. This notion has already been applied in battle-tested libraries such as the `ABDKMath64x64` and `ABDKMathQuad` libraries which we advise be utilized instead. In general, the comments of the code do not always conform with its actual functionality which we suggest be fixed in the codebase to ensure that the traits it is meant to possess can be properly validated.

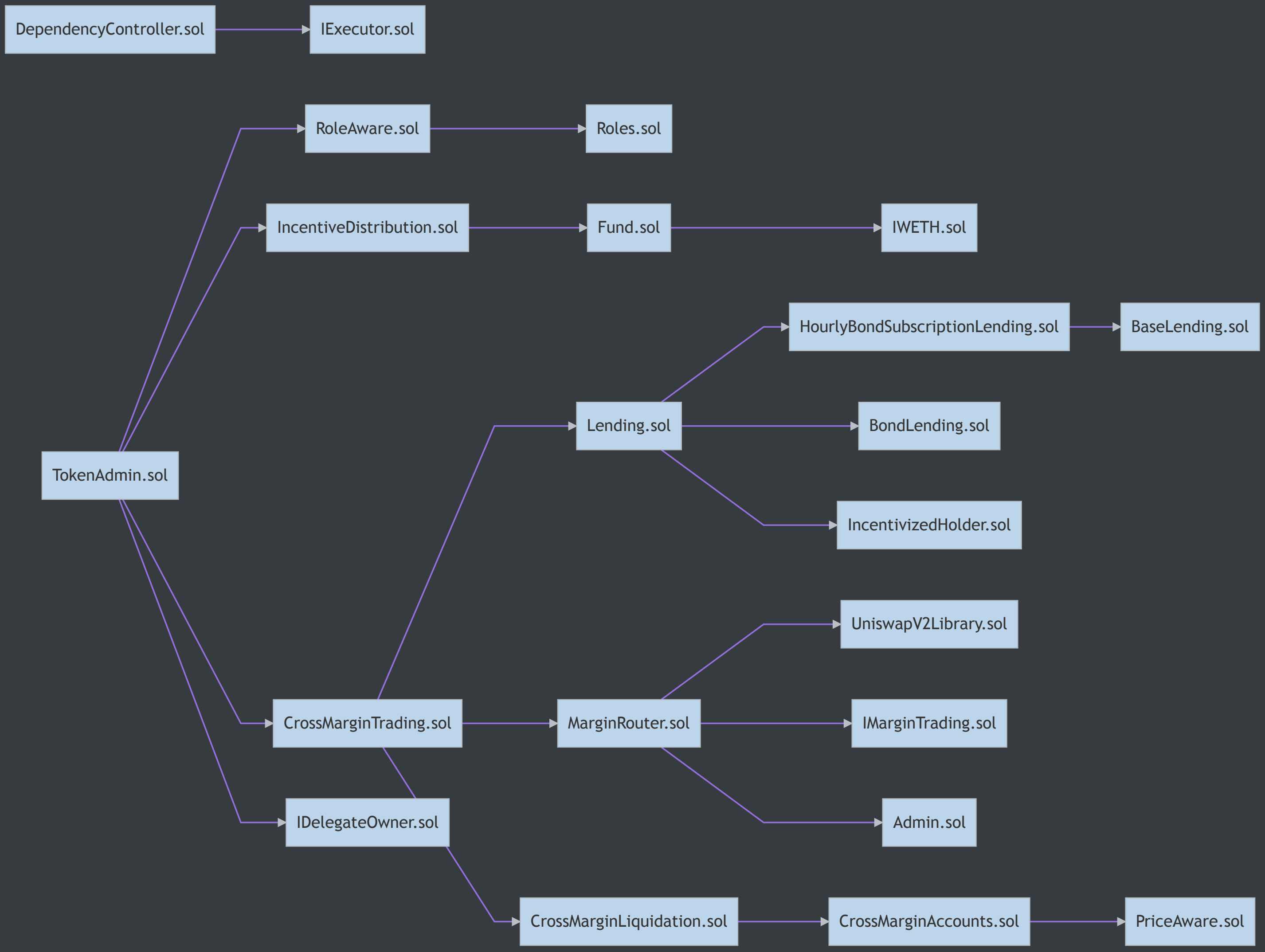
The ability to execute certain functions is also indirectly guarded instead of being explicitly guarded via `require` checks, rendering the validation of valid execution paths difficult. We advise that these checks are instead directly applied as the system performs multiple external calls within each user interaction that can easily be misassessed and lead to a successful execution of an invalid execution path.

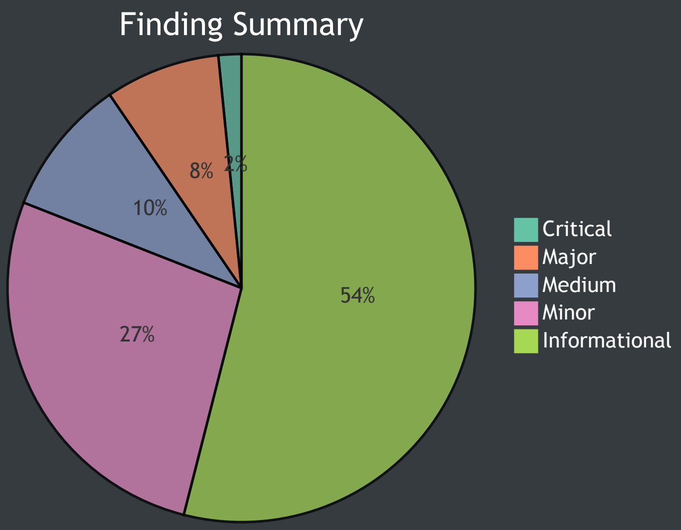
During our remediation round, we observed multiple changes across the codebase that refactored a lot of the functionality, replaced or entirely removed previously existent one and introduced comments indicating pending actions are meant to also be implemented on the codebase. Given the size of the adjustment and the multiple reworks that were performed across the codebase, we cannot attest to the security of newly introduced functions, contracts and logical paths that did not exist and simply validated that the findings that were identified during the second round were either alleviated or no longer applicable in the new version of the codebase.

Files In Scope

ID	Contract	Location
ADM	Admin.sol	contracts/Admin.sol
BLG	BaseLending.sol	contracts/BaseLending.sol
BLN	BondLending.sol	contracts/BondLending.sol
CMA	CrossMarginAccounts.sol	contracts/CrossMarginAccounts.sol
CML	CrossMarginLiquidation.sol	contracts/CrossMarginLiquidation.sol
CMT	CrossMarginTrading.sol	contracts/CrossMarginTrading.sol
DCR	DependencyController.sol	contracts/DependencyController.sol
FUN	Fund.sol	contracts/Fund.sol
HBS	HourlyBondSubscriptionLending.sol	contracts/HourlyBondSubscriptionLending.sol
IDN	IncentiveDistribution.sol	contracts/IncentiveDistribution.sol
IHR	IncentivizedHolder.sol	contracts/IncentivizedHolder.sol
LEN	Lending.sol	contracts/Lending.sol
LMR	LiquidityMiningReward.sol	contracts/LiquidityMiningReward.sol
MRR	MarginRouter.sol	contracts/MarginRouter.sol
PAE	PriceAware.sol	contracts/PriceAware.sol
RAE	RoleAware.sol	contracts/RoleAware.sol
ROL	Roles.sol	contracts/Roles.sol
TAN	TokenAdmin.sol	contracts/TokenAdmin.sol
IDO	IDelegateOwner.sol	interfaces/IDelegateOwner.sol
IER	IExecutor.sol	interfaces/IExecutor.sol
IMT	IMarginTrading.sol	interfaces/IMarginTrading.sol
IWE	IWETH.sol	interfaces/IWETH.sol
UVL	UniswapV2Library.sol	libraries/UniswapV2Library.sol

File Dependency Graph




Finding Summary
2%
8%
10%
54%
27%
Critical
Major
Medium
Minor
Informational

Manual Review Findings

ID	Title	Type	Severity	Resolved
ADM-01M	Invalid Assignment	Logical Fault	● Critical	✓
ADM-02M	Improper Evaluation of Stake	Logical Fault	● Medium	✓
ADM-03M	Variable Mutability Specifiers	Gas Optimization	● Informational	✓
ADM-04M	Invalid Operation	Coding Style	● Informational	✓
ADM-05M	Visibility Specifiers Missing	Language Specific	● Informational	✓
BLG-01M	Inefficient Storage Layout	Gas Optimization	● Informational	✓
BLG-02M	Unutilized Variable	Coding Style	● Informational	ⓘ
BLG-03M	Redundant Conditional	Gas Optimization	● Informational	✓
BLN-01M	Commented Out Functionality	Logical Fault	● Major	✓
BLN-02M	Incorrect Input Sanitization	Logical Fault	● Minor	✓
BLN-03M	Inefficient Storage Layout	Gas Optimization	● Informational	✓
BLN-04M	Inefficient Loop Data Type	Gas Optimization	● Informational	✓

CMA-01M	Insufficient Removal from System	Logical Fault	🟡 Medium	✓
CMA-02M	Redundant Statements	Dead Code	🟢 Informational	✓
CMA-03M	Visibility Specifiers Missing	Language Specific	🟢 Informational	✓
CML-01M	Always Failing Statement	Mathematical Operations	🟠 Major	✓
CML-02M	Incorrect Function Invocation	Logical Fault	🟡 Medium	✓
CML-03M	Calculation Inaccuracy	Logical Fault	🔵 Minor	✓
CML-04M	Unclear Handling of Code Branches	Logical Fault	🔵 Minor	✓
CML-05M	Visibility Specifiers Missing	Language Specific	🟢 Informational	ⓘ
CML-06M	Manual Zeroing of Storage	Coding Style	🟢 Informational	ⓘ
CMT-01M	Unutilized Liquidation Functionality	Logical Fault	🔵 Minor	✓
DCR-01M	Inconsistent Tracking of Ownership	Logical Fault	🟡 Medium	✓
DCR-02M	Misbehaving Evaluation	Language Specific	🔵 Minor	✓
DCR-03M	Redundant Assignment of Default Value	Coding Style	🟢 Informational	✓
DCR-	Inefficient Loop Limits	Gas Optimization	🟢 Informational	✓

04M				
DCR-05M	Optimization of Lookup	Gas Optimization	🟢 Informational	✓
FUN-01M	Potential Incompatibility of Standard	Standard Conformity	🟡 Medium	✓
FUN-02M	Usage of `transfer()` for sending Ether	Volatile Code	🔵 Minor	✓
FUN-03M	Variable Mutability Specifier	Gas Optimization	🟢 Informational	✓
FUN-04M	Potentially Incorrect Check	Coding Style	🟢 Informational	✓
HBS-01M	Incorrect Overwriting of Variable	Logical Fault	🔵 Minor	✓
HBS-02M	Visibility Specifier Missing	Language Specific	🟢 Informational	✓
IDN-01M	Checks-Effects-Interactions Pattern Inapplied	Logical Fault	🔵 Minor	✓
IDN-02M	Unguarded Function	Logical Fault	🔵 Minor	✓
IDN-03M	Comment Discrepancy	Coding Style	🟢 Informational	✓
IDN-04M	Visibility Specifiers Missing	Language Specific	🟢 Informational	✓
IDN-05M	Potentially Inconsistent Per-Member `delete`	Coding Style	🟢 Informational	✓
IDN-06M	Variable Mutability Specifier	Gas Optimization	🟢 Informational	✓
IHR-01M	Unsanitized Return	Logical Fault	🟡 Medium	✓

	Value			
IHR-02M	Inexistent Branch Handling	Logical Fault	🔵 Minor	❗
LEN-01M	Inexistent Bond Stake Creation	Logical Fault	🟠 Major	✓
LEN-02M	Inexistent Access Control	Logical Fault	🟠 Major	✓
LEN-03M	Checks-Effects-Interactions Pattern Inapplied	Logical Fault	🔵 Minor	✓
LEN-04M	Unrestricted Execution	Logical Fault	🔵 Minor	✓
LMR-01M	Improper Migration	Logical Fault	🔵 Minor	✓
LMR-02M	Checks-Effects-Interactions Pattern Inapplied	Logical Fault	🔵 Minor	✓
LMR-03M	Visibility Specifier Missing	Language Specific	🟢 Informational	✓
LMR-04M	Variable Mutability Specifiers	Gas Optimization	🟢 Informational	✓
MRR-01M	Inexistent Access Control	Coding Style	🟢 Informational	✓
MRR-02M	Variable Mutability Specifiers	Gas Optimization	🟢 Informational	✓
PAE-01M	Susceptible to Oracle Price Manipulation	Logical Fault	🟠 Major	❗
PAE-02M	Unchecked Bounds	Logical Fault	🔵 Minor	✓

PAE-03M	Visibility Specifiers Missing	Language Specific	● Informational	✓
PAE-04M	Redundant Duplicate Structure	Gas Optimization	● Informational	ⓘ
PAE-05M	Variable Mutability Specifiers	Gas Optimization	● Informational	✓
RAE-01M	Variable Mutability Specifier	Gas Optimization	● Informational	✓
RAE-02M	Redundant Declarations	Gas Optimization	● Informational	✓
ROL-01M	Single Point of Failure	Logical Fault	● Minor	✓
ROL-02M	Duplicate User-Specified Getter	Coding Style	● Informational	ⓘ
ROL-03M	Inefficient Key Type	Gas Optimization	● Informational	✓

Static Analysis Findings

ID	Title	Type	Severity	Resolved
RAE-01S	Unchecked Input Address	Logical Fault	● Minor	✓
TAN-01S	Variable Data Location Optimization	Gas Optimization	● Informational	✓

ADM-01M: Invalid Assignment

Type	Severity	Location
Logical Fault	● Critical	Admin.sol L91-L114

Description:

The `_withdrawStake` function should subtract the `amount` to be withdrawn from a user's `stakes` however the statement of L98 conducts an assignment (`=`) instead of a subtraction and assignment (`-=`) leading to the `amount` withdrawn to never become subtracted.

Recommendation:

We advise the proper operation to be introduced in the specified line as the system misbehaves in its current state.

Alleviation:

Stakes are properly subtracted from a user's `stakes` mapping entry, alleviating this issue.

ADM-02M: Improper Evaluation of Stake

Type	Severity	Location
Logical Fault	● Medium	Admin.sol L183-L185, L214-L216

Description:

The `getMaintenanceStakerStake` function contains a special branch path if the `staker` is the `lockedMFI` address, however, the `getUpdatedCurrentStaker` and `viewCurrentMaintenanceStaker` functions ignore this in the statements of their nested `else` blocks.

Recommendation:

We advise the proper stake to be utilized in the calculations of L183-L185 and L214-L216 as this can lead to improper system operation.

Alleviation:

The `getUpdatedCurrentStaker` and `viewCurrentMaintenanceStaker` functions were properly updated to use the result of `getMaintenanceStakerStake` instead of accessing the `stakes` mapping directly.

ADM-03M: Variable Mutability Specifiers

Type	Severity	Location
Gas Optimization	● Informational	Admin.sol L13, L18, L27, L36, L37, L39

Description:

The linked variables are assigned to only once during the contract's `constructor`.

Recommendation:

We advise their mutability specifier to be set to `immutable` greatly optimizing the gas cost of the contract.

Alleviation:

The `feesPer10k` variable was omitted from the codebase, however the other two variables had the `immutable` mutability specifier set thus optimizing the codebase greatly.

ADM-04M: Invalid Operation

Type	Severity	Location
Coding Style	● Informational	Admin.sol L139

Description:

The `subtractTradingFees` has a misleading function name as it still accumulates trading fees.

Recommendation:

We advise the function to be renamed to more accurately represent its intended usage as is within `MarginRouter`.

Alleviation:

The fee mechanism was completely removed as a concept from the contract thus rendering this exhibit null.

ADM-05M: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	● Informational	Admin.sol L13, L18

Description:

The linked declarations have no visibility specifiers explicitly set.

Recommendation:

We advise visibility specifiers to be defined for these variables as the current status is that they are automatically assigned by the compiler which can lead to compilation discrepancies between different versions.

Alleviation:

The `feesPer10k` variable was omitted from the codebase and an explicit visibility specifier was added to the `MFI` variable thus alleviating this exhibit.

BLG-01M: Inefficient Storage Layout

Type	Severity	Location
Gas Optimization	● Informational	BaseLending.sol L9-L12

Description:

The linked `mapping` declarations can instead be merged into a single `mapping` that associates a token with a `struct`.

Recommendation:

We advise it to be done so to increase the legibility of the codebase and optimize the gas cost involved in accessing multiple of these variables at once since each `mapping` lookup performs a `keccak256` operation underneath which can become costly.

Alleviation:

The `mapping` declarations were correctly grouped into a single one that points to a new `LendingMetadata` struct containing the variables of the previously set `mapping` declarations.

BLG-02M: Unutilized Variable

Type	Severity	Location
Coding Style	● Informational	BaseLending.sol L14, L95-L97

Description:

The `maxHourlyYieldFP` indicates an upper bound for the `yieldDiff` when an hour has passed, however, the limit remains un-imposed on the `updatedYieldFP` function.

Recommendation:

We advise it is relocated to the `HourlyBondSubscriptionLending` contract where it is appropriately utilized.

Alleviation:

The MarginSwap - Core Implementation Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

BLG-03M: Redundant Conditional

Type	Severity	Location
Gas Optimization	● Informational	BaseLending.sol L48-L51

Description:

The first clause of the `if` block guarantees the validity of the `else` clause.

Recommendation:

We advise the conditional of the `else` clause to be safely omitted from the codebase.

Alleviation:

The conditional of the `else` clause was correctly removed from the `if-else` structure.

BLN-01M: Commented Out Functionality

Type	Severity	Location
Logical Fault	● Major	BondLending.sol L223-L226

Description:

The linked `require` check of the `setRuntimeWeights` meant to guard their modification has been commented out enabling anyone to set the weights.

Recommendation:

We advise the access control to be re-enabled by un-commenting the linked code block.

Alleviation:

The `setRuntimeWeights` function was omitted from the codebase rendering this finding no longer applicable.

BLN-02M: Incorrect Input Sanitization

Type	Severity	Location
Logical Fault	● Minor	BondLending.sol L261-L264

Description:

The `setMinRuntime` function permits the newly set `minRuntime` to be higher than the existing `maxRuntime` causing a misconfiguration of the system.

Recommendation:

We advise such a scenario to be prohibited via a `require` check that ensures `runtime` is less-than the existing `maxRuntime`.

Alleviation:

The `setMinRuntime` function was omitted from the codebase rendering the exhibit no longer applicable.

BLN-03M: Inefficient Storage Layout

Type	Severity	Location
Gas Optimization	● Informational	BondLending.sol L31-L36, L235-L240

Description:

The linked `mapping` declarations are all initialized to the same-size `uint256` array inefficiently so.

Recommendation:

We advise a single `mapping` to be utilized that points to an array of `struct` types that hold the necessary data for each bucket.

Alleviation:

The numerous `mapping` declarations were grouped to a single one that points to a new `BondBucketMetadata` struct containing the corresponding variables of the previous `mapping` declarations.

BLN-04M: Inefficient Loop Data Type

Type	Severity	Location
Gas Optimization	● Informational	BondLending.sol L245-L255

Description:

The EVM is geared towards 32-byte data types and utilizes more gas to interact with less-than-32-byte data types.

Recommendation:

We advise the data type utilized in the linked loop to be changed to a `uint256` to optimize its gas cost.

Alleviation:

The `setRuntimeWeights` function was omitted from the codebase rendering this exhibit no longer applicable.

CMA-01M: Insufficient Removal from System

Type	Severity	Location
Logical Fault	● Medium	CrossMarginAccounts.sol L171-L174

Description:

The full extinguishment of a debt is not properly handled.

Recommendation:

When a debt is fully extinguished, it should be removed from the `borrowTokens` array and have its `borrowedYieldQuotientsFP` zeroed out as a consequent `borrow` will misbehave and utilize an outdated `borrowedYielQuotientsFP` since it is not updated in the case of full extinguishment.

Alleviation:

In the updated version of the codebase, the deletion of the `borrowedYieldQuotientsFP` is done so correctly but the removal of the token from the `borrowTokens` array is done in a gas-expensive manner. We advise that instead of decrementing the full array list, the last element of the array replaces the `debtToken` to remove and the `pop` instruction is called right after greatly optimizing the gas cost involved in this operation provided that the order of the tokens bears no effect to the protocol's functionality.

CMA-02M: Redundant Statements

Type	Severity	Location
Dead Code	● Informational	CrossMarginAccounts.sol L65, L87

Description:

The linked statements have no effect to the codebase.

Recommendation:

We advise them to be safely removed.

Alleviation:

The functions that encompassed these statements have been removed from the contract thus rendering this exhibit null.

CMA-03M: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	● Informational	CrossMarginAccounts.sol L38

Description:

The linked declarations have no visibility specifiers explicitly set.

Recommendation:

We advise visibility specifiers to be defined for these variables as the current status is that they are automatically assigned by the compiler which can lead to compilation discrepancies between different versions.

Alleviation:

The `internal` visibility specifier was properly set for the `marginAccounts` variable.

CML-01M: Always Failing Statement

Type	Severity	Location
Mathematical Operations	● Major	CrossMarginLiquidation.sol L169

Description:

The `_disburseLiqAttack` function will always fail if `attackerCut` is greater-than `0` as the `returnAmount` variable is never initialized defaulting to `0` and the statement of L169 subtracts the `attackerCut` from it which can error due to underflow.

Recommendation:

We advise the subtraction to be replaced by an addition as that is the intended behaviour of the system.

Alleviation:

The correct calculation was set to the `returnAmount` whereby `liqAttackRecord.amount - attackerCut` is performed rather than `returnAmount -= attackerCut` thereby fixing this exhibit.

CML-02M: Incorrect Function Invocation

Type	Severity	Location
Logical Fault	● Medium	CrossMarginLiquidation.sol L192

Description:

The `liquidateFromPeg` function invokes the `liquidateToPeg` function of `PriceAware` instead of the `liquidateFromPeg` function.

Recommendation:

We advise the current function to be utilized here as tokens are meant to be bought at this statement and not sold.

Alleviation:

The `liquidateToPeg` invocation was replaced by a `liquidateFromPeg` invocation correctly.

CML-03M: Calculation Inaccuracy

Type	Severity	Location
Logical Fault	● Minor	CrossMarginLiquidation.sol L307-L309

Description:

The variable `avgLiquidationPerBlock` is meant to represent the average liquidation that occurs per block, however, the variable is calculated by simply accumulating liquidations using multipliers and does not factor in the number of liquidations conducted in a block whatsoever.

Recommendation:

We advise the variable to be renamed or its calculations adjusted to properly reflect the liquidation conducted per block.

Alleviation:

The variable was renamed to `avgLiquidationPerCall` properly reflecting its purpose and what data it represents.

CML-04M: Unclear Handling of Code Branches

Type	Severity	Location
Logical Fault	● Minor	CrossMarginLiquidation.sol L228-L321

Description:

The code branch of `liquidate` that will execute if `isAuthorized` is `false` but `canTakeNow` is `true` does not actually transmit the attacker's cut "now" as `canTakeNow` indicates.

Recommendation:

We advise the code block to be better documented to detail what is meant to be achieved in such a case and if it is desirable to distribute the attacker's cut on invocation, such statements to be introduced as `else` clauses to the `if (!canTakeNow)` blocks.

Alleviation:

A new `if` block was introduced that evaluates whether `canTakeNow` is `true` and distributes the `maintainerCut` immediately.

CML-05M: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	● Informational	CrossMarginLiquidation.sol L30-L33

Description:

The linked declarations have no visibility specifiers explicitly set.

Recommendation:

We advise visibility specifiers to be defined for these variables as the current status is that they are automatically assigned by the compiler which can lead to compilation discrepancies between different versions.

Alleviation:

Visibility specifiers were added for all but the `liquidationAmounts` contract variable where we still advise visibility specifiers to be introduced.

CML-06M: Manual Zeroing of Storage

Type	Severity	Location
Coding Style	● Informational	CrossMarginLiquidation.sol L164-L167

Description:

The linked statements attempt to delete the `AccountLiqRecord` from `storage` by manually assigning zeroed out values of each member type of the `struct`.

Recommendation:

We advise these statements to be replaced by a single `delete` instruction to ensure compatibility of the code block even if the `struct` members are updated.

Alleviation:

The deletion of the record was instead relocated outside of the `_disburseLiqAttack` function and after the respective function call within `disburseLiqStakeAttacks`. We should note that the record is not properly removed in the case of the `if (isAuthorized)` branch of `calcLiquidationAmounts` which we believe not to be intended behavior and we advise is fixed.

CMT-01M: Unutilized Liquidation Functionality

Type	Severity	Location
Logical Fault	● Minor	CrossMarginTrading.sol L180-L193

Description:

The `registerLiquidation` function remains unutilized across the codebase.

Recommendation:

We advise this to be properly integrated within the necessary components, potentially within the router or the cross margin liquidator depending on the desired technical layout of the protocol.

Alleviation:

The function is now invoked by the `MarginRouter` implementation in a new function and was refactored thus rendering this exhibit null.

DCR-01M: Inconsistent Tracking of Ownership

Type	Severity	Location
Logical Fault	● Medium	DependencyController.sol L98-L100

Description:

The `delegateOwner` mapping is meant to associate a contract with its `owner` as done so at L170-L173, however, only ownership is transferred in the linked blocks.

Recommendation:

We advise the new owner to also be recorded in the `delegateOwner` mapping to prevent misbehavior of the system. In its current state, `executeAsOwner` does not appear to return ownership of contracts acquired via the `else` block of L98-L100.

Alleviation:

The relevant code segments were revamped rendering this exhibit no longer applicable.

DCR-02M: Misbehaving Evaluation

Type	Severity	Location
Language Specific	● Minor	DependencyController.sol L58-L64

Description:

The `ownsContract` function will misbehave for a contract with no `owner` that is also not recorded as the default value of `delegateOwner` will be the zero-address as would be the `owner` of an owner-less contract.

Recommendation:

We advise an additional check to be imposed on the third conditional of the `return` clause whereby `contrOwner == delegateOwner[contr]` is preceded by `delegateOwner[contr] != address(0) &&` to ensure that an entry exists for the specified contract.

Alleviation:

The relevant function was removed from the codebase rendering this exhibit null.

DCR-03M: Redundant Assignment of Default Value

Type	Severity	Location
Coding Style	● Informational	DependencyController.sol L19

Description:

All data types in Solidity possess a default value they are automatically assigned to.

Recommendation:

We advise the zero-address assignment of L19 to be omitted as that is the default value of an `address`.

Alleviation:

The redundant assignment was safely removed from the codebase.

DCR-04M: Inefficient Loop Limits

Type	Severity	Location
Gas Optimization	● Informational	DependencyController.sol L44-L47, L52-L55, L119-L128, L130-L132, L156-L158, L161-L168, L181-L185, L187-L191, L236-L238, L242-L244

Description:

The limits of the linked `for` loops are dynamically evaluated from storage instead of being assigned to an in-memory variable and compared from there.

Recommendation:

We advise it to be done so to greatly optimize the gas cost involved in the function's of these loops.

Alleviation:

The loop limits where still applicable were revised to properly extract the `length` in an in-memory variable prior to commencing the loop greatly optimizing their gas cost.

DCR-05M: Optimization of Lookup

Type	Severity	Location
Gas Optimization	● Informational	DependencyController.sol L42-L48, L50-L56

Description:

The linked functions evaluate whether the `manageContracts` are all either indirectly owned or strictly owned by the `DependencyController` respectively, in doing so iterating over all contracts when redundant.

Recommendation:

We advise the linked code blocks to return early if they identify a contract whose ownership is not valid to greatly optimize their gas cost.

Alleviation:

The relevant functions were removed from the codebase rendering this exhibit no longer applicable.

FUN-01M: Potential Incompatibility of Standard

Type	Severity	Location
Standard Conformity	● Medium	Fund.sol L32-L43, L45-L58, L65-L75

Description:

The `transfer` and `transferFrom` functions are meant to return a `bool` on successful invocation, however, not all ERC-20 tokens are fully compliant (i.e. Tether) and may not return a `bool` at all causing calls that expect a return variable to `revert` even if otherwise successful.

Recommendation:

We advise that a wrapper library like `SafeERC20` from OpenZeppelin is utilized here to ensure that the return variable is not mandated and is instead optionally evaluated.

Alleviation:

All `transfer` and `transferFrom` invocations were replaced by their safe counterpart from the OpenZeppelin `SafeERC20` library properly alleviating this exhibit.

FUN-02M: Usage of `transfer()` for sending Ether

Type	Severity	Location
Volatile Code	● Minor	Fund.sol L81

Description:

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300` . This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation:

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

Alleviation:

The transfer of Ether was replaced by `Address.sendValue` from the OpenZeppelin `Address` library. We advise it to wrap the `address payable` type directly to make the invocation more intuitive.

FUN-03M: Variable Mutability Specifier

Type	Severity	Location
Gas Optimization	● Informational	Fund.sol L13

Description:

The `WETH` variable is assigned to only once during the `constructor` of the contract.

Recommendation:

We advise the `immutable` mutability specifier to be introduced to the variable's declaration greatly optimizing the gas cost involved in utilizing it. Additionally, we advise the `public` visibility specifier to be omitted.

Alleviation:

The `immutable` specifier was introduced to the `WETH` variable greatly optimizing the codebase.

FUN-04M: Potentially Incorrect Check

Type	Severity	Location
Coding Style	● Informational	Fund.sol L51

Description:

The `require` check imposed on the `depositFor` function ensures that the `msg.sender` is a withdrawer instead of a borrower. Additionally, the error message states "Contract not authorized to deposit" which is not true as a contract can still deposit via the `deposit` function.

Recommendation:

We advise the error message to be adjusted and the getter function utilized to potentially be renamed as the check is valid due to both `isWithdrawer`-guarded functions be utilized by the `BondLending` for example but can be slightly misleading in the `depositFor` function.

Alleviation:

The access control of the function was revised to instead utilize `isFundTransferer` which is more legible and contains an explicit error message.

HBS-01M: Incorrect Overwriting of Variable

Type	Severity	Location
Logical Fault	● Minor	HourlyBondSubscriptionLending.sol L48-L51

Description:

The `setHourlyYieldAPR` function is meant to update an `hourlyBondYieldAccumulators` entry for a specified token to the new `aprPercent`, however, in doing so in the `else` branch it simply overwrites the previous `hourlyYieldFP`. In case a yield for a token hasn't been updated recently, the new hourly yield will retroactively apply which should not be the case.

Recommendation:

We advise the bond to be updated before being overwritten and the `lastUpdated` member to be set to the latest `block.timestamp`.

Alleviation:

The function was omitted from the codebase rendering this exhibit no longer applicable.

HBS-02M: Visibility Specifier Missing

Type	Severity	Location
Language Specific	● Informational	HourlyBondSubscriptionLending.sol L22

Description:

The linked declaration has no visibility specifier explicitly set.

Recommendation:

We advise a visibility specifier to be defined for this variable as the current status is that it is automatically assigned by the compiler which can lead to compilation discrepancies between different versions.

Alleviation:

The `public` visibility specifier was properly set for the `withdrawalWindow` variable.

IDN-01M: Checks-Effects-Interactions Pattern Inapplied

Type	Severity	Location
Logical Fault	● Minor	IncentiveDistribution.sol L251-L263

Description:

The `_withdrawReward` function calculates the `rewardAmount` to be paid out and distributes it before the assignment of the latest reward rate of the tranche to the `claim.startingRewardRateFP` which can cause a re-entrancy vulnerability whereby a user repeatedly withdraws their reward if they are informed as the recipient of the withdrawal.

Recommendation:

We advise the assignment of L262 to be conducted after the reward calculation and before the external call to ensure full compliancy with the Checks-Effects-Interactions pattern.

Alleviation:

The linked function was removed and replaced by a more detailed `withdrawReward` function that properly zeroes out the accrued rewards prior to paying out the rewards.

IDN-02M: Unguarded Function

Type	Severity	Location
Logical Fault	● Minor	IncentiveDistribution.sol L102-L104

Description:

The `forcePeriodTotalUpdate` function is meant to enable external parties to force an update of a period's totals, however, the function `updatePeriodTotals` is solely invoked in guarded functions that can be called only by an incentive reporter.

Recommendation:

We advise that the potential inclusion of the `isIncentiveReporter` check within the `forcePeriodTotalUpdate` function to be evaluated and if deemed necessary applied.

Alleviation:

The relevant functions were removed from the codebase rendering this exhibit no longer applicable.

IDN-03M: Comment Discrepancy

Type	Severity	Location
Coding Style	● Informational	IncentiveDistribution.sol L17, L19

Description:

The comment denotes the variable that follows it specifies the contraction per thousand, however, the variable is suffixed with `PerMil` indicating million.

Recommendation:

We advise either the comment or the variable to be adjusted to properly reflect what they are meant to depict.

Alleviation:

A clarifying comment was added above the declaration that points to the `Per Mille` notation that the `PerMil` suffix is meant to represent alleviating this exhibit.

IDN-04M: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	● Informational	IncentiveDistribution.sol L16, L19, L21, L22, L23, L40

Description:

The linked declarations have no visibility specifiers explicitly set.

Recommendation:

We advise visibility specifiers to be defined for these variables as the current status is that they are automatically assigned by the compiler which can lead to compilation discrepancies between different versions.

Alleviation:

Most variables were omitted in the updated version of the codebase but those that remained had their visibility specifiers properly set.

IDN-05M: Potentially Inconsistent Per-Member `delete`

Type	Severity	Location
Coding Style	● Informational	IncentiveDistribution.sol L213-L215

Description:

The linked segment deletes each member of the `Claim` struct sequentially instead of deleting the full struct at once.

Recommendation:

We advise it to be done so to ensure the codebase remains up to date even if the `Claim` struct members are adjusted.

Alleviation:

The relevant functions were removed from the codebase rendering this exhibit no longer applicable.

IDN-06M: Variable Mutability Specifier

Type	Severity	Location
Gas Optimization	● Informational	IncentiveDistribution.sol L23, L30

Description:

The `MFI` variable is assigned to only once during the `constructor` of the contract.

Recommendation:

We advise it to be set to `immutable` to greatly benefit from the gas optimizations it brings.

Alleviation:

The `immutable` trait was properly introduced to the `MFI` variable greatly optimizing the codebase.

IHR-01M: Unsanitized Return Value

Type	Severity	Location
Logical Fault	● Medium	IncentivizedHolder.sol L34-L37

Description:

The `startClaim` function is meant to return `0` in case the system is paused in which case the `stakeClaim` function should revert as is the behaviour in the `LiquidityMiningReward` contract.

Recommendation:

We advise a similar `require` check to be introduced here to ensure no stakes are ultimately lost in the contract.

Alleviation:

The logic of the function was migrated to the `IncentiveDistribution` contract thus rendering this exhibit null.

IHR-02M: Inexistent Branch Handling

Type	Severity	Location
Logical Fault	● Minor	IncentivizedHolder.sol L40-L52

Description:

The `withdrawClaim` function does not contain a branch of execution in case the full claim is meant to be withdrawn.

Recommendation:

We advise such a branch to be introduced whereby a claim is ended via the corresponding function on the `IncentiveDistribution` contract.

Alleviation:

The MarginSwap - Core Implementation Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

LEN-01M: Inexistent Bond Stake Creation

Type	Severity	Location
Logical Fault	● Major	Lending.sol L122-L144

Description:

The `buyBond` function is meant to create a bond and a corresponding stake for the bond. However, the functions within can silently fail leading to the creation of a `stakeClaim` with no bond to back it.

Recommendation:

We advise that an error handling route is added to the returned `bondIndex` whereby it is evaluated to be different than `0` as the `_makeBond` function of `BondLending` can silently fail if the `minReturn` is not achieved.

Alleviation:

Proper handling of the silent failures was introduced thus preventing unbacked bonds from being created.

LEN-02M: Inexistent Access Control

Type	Severity	Location
Logical Fault	● Major	Lending.sol L27-L41

Description:

The `applyBorrowInterest` function has no access control imposed enabling anyone to arbitrarily increase the `totalBorrowed` of a particular token by supplying malicious `balance` and `yieldQuotientFP` variables.

Recommendation:

We advise proper access control to be imposed on the function as it can completely break the system.

Alleviation:

Access control was properly imposed to the function via the `isBorrower` evaluation newly introduced.

LEN-03M: Checks-Effects-Interactions Pattern Inapplied

Type	Severity	Location
Logical Fault	● Minor	Lending.sol L147-L161

Description:

The `withdrawBond` function conducts an outward transfer of a user's bond within `super._withdrawBond` of `BondLending` without zeroing out the `bond` via a `delete` operation which can cause a re-entrancy vulnerability whereby a user repeatedly withdraws their bond if they are informed as the recipient of the withdrawal.

Recommendation:

We advise that the `delete` statement that follows `super._withdrawBond` is relocated to precede it and that the `super._withdrawBond` function is adjusted to accept a `memory` argument rather than a `storage` argument which L148 should be set to to ensure full compliancy with the Checks-Effects-Interactions pattern.

Alleviation:

The Checks-Effects-Interactions pattern was applied by revamping the codebase and re-ordering the function calls contained within.

LEN-04M: Unrestricted Execution

Type	Severity	Location
Logical Fault	● Minor	Lending.sol L163-L169

Description:

The `initBorrowYieldAccumulator` function can be invoked at any time regardless of whether the underlying `accumulatorFP` is non-zero which can lead to discrepancies in the yields accumulated.

Recommendation:

We advise the function to introduce another `require` check that ensures the `accumulatorFP` is equal to zero before being initialized to a non-zero value.

Alleviation:

Proper access control in both the party that initiates it as well as when it is possible to do so was added to the `initBorrowYieldAccumulator` function.

LMR-01M: Improper Migration

Type	Severity	Location
Logical Fault	🔵 Minor	LiquidityMiningReward.sol L28-L33

Description:

The `migrateIncentiveDistributor` simply overwrites the previous `incentiveDistributor` with the new one without conducting a proper migration process, such as revoking "withdrawer" rights from the previous incentive distribution.

Recommendation:

We advise that the migration process is properly conducted by completely deleting the previous `incentiveDistributor` contract from the system by revoking its roles.

Alleviation:

The relevant function has been removed from the codebase rendering this exhibit no longer applicable.

LMR-02M: Checks-Effects-Interactions Pattern Inapplied

Type	Severity	Location
Logical Fault	● Minor	LiquidityMiningReward.sol L59-L76

Description:

The `withdrawStake` function conducts an outward transfer of a user's stake without subtracting it first from their `stakeAmounts` which can cause a re-entrancy vulnerability whereby a user repeatedly withdraws their stake if they are informed as the recipient of the withdrawal.

Recommendation:

We advise the subtraction of the withdrawn amount from the `stakeAmounts` to be done before the external call to ensure full compliancy with the Checks-Effects-Interactions pattern.

Alleviation:

The Checks-Effects-Interactions pattern was properly applied by relocating the outward transfer at the end of the function block.

LMR-03M: Visibility Specifier Missing

Type	Severity	Location
Language Specific	● Informational	LiquidityMiningReward.sol L15

Description:

The linked declaration has no visibility specifier explicitly set.

Recommendation:

We advise a visibility specifier to be defined for this variable as the current status is that it is automatically assigned by the compiler which can lead to compilation discrepancies between different versions.

Alleviation:

The relevant variable was omitted from the codebase rendering this exhibit no longer applicable.

LMR-04M: Variable Mutability Specifiers

Type	Severity	Location
Gas Optimization	● Informational	LiquidityMiningReward.sol L12, L16, L24, L25

Description:

The linked variables are assigned to only once during the contract's `constructor`.

Recommendation:

We advise them to be set as `immutable` to benefit from the gas optimization it brings.

Alleviation:

Both linked variables were set to `immutable` greatly optimizing the gas cost of the contract.

MRR-01M: Inexistent Access Control

Type	Severity	Location
Coding Style	● Informational	MarginRouter.sol L243-L270, L273-L303

Description:

The `crossSwapTokensForExactTokens` and `crossSwapExactTokensForTokens` functions possess no access control and provide full access to the underlying liquidity of the protocol to the caller to perform arbitrary swaps with the underlying assets using a user-specified `path`.

Recommendation:

Although validation occurs within the code path of `MarginRouter.registerTrade` -> `CrossMarginTrading.registerTradeAndBorrow` -> `CrossMarginAccounts.adjustAmounts`, we advise a more explicit validation path to be imposed at the start of the function to prevent gas waste and increase the legibility of the codebase.

Alleviation:

The `registerTrade` functions within are meant to impose this access control as denoted in the newly introduced comments above them thus rendering this exhibit null.

MRR-02M: Variable Mutability Specifiers

Type	Severity	Location
Gas Optimization	● Informational	MarginRouter.sol L18, L59

Description:

The linked variables are assigned to only once during the contract's `constructor`.

Recommendation:

We advise their mutability specifier to be set to `immutable` greatly optimizing the gas cost of the contract.

Alleviation:

The linked variable was properly set to `immutable` greatly benefitting from the gas optimizations it brings.

PAE-01M: Susceptible to Oracle Price Manipulation

Type	Severity	Location
Logical Fault	● Major	PriceAware.sol L74-L100

Description:

The `getUpdatedPriceInPeg` relies on a price update based on UniSwap using the `getAmountsOut` function which can yield unreliable results in case a flash loan has been deposited temporarily to the pair.

Recommendation:

Although ranges do exist within L91-L94 that are meant to guarantee the fluctuation of price is within a minimum and maximum range, the function can still be exploited whereby the price is kept "still" or even slightly moved within the bounds as this can be pre-calculated. We advise the calculation of price based on TWAPs (Time-Weighted Average Prices) rather than spot prices to ensure security in the protocol. A confidence rating is utilized in L130-L143, however, sharp changes in price can still overpower the "weighted" average imposed here as it is not sufficiently parameterized to prevent such an attack.

Alleviation:

The MarginSwap - Core Implementation Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

PAE-02M: Unchecked Bounds

Type	Severity	Location
Logical Fault	● Minor	PriceAware.sol L184, L206

Description:

The `confidentUpdatePriceInPeg` invocation of `liquidateToPeg` and `liquidateFromPeg` does not perform the same bound checks as `getUpdatedPriceInPeg` while being susceptible to the same price fluctuations.

Recommendation:

We advise the same bounds to be applied here or omitted depending on how the oracle attack issue is dealt with.

Alleviation:

The function invocations were removed from the codebase alleviating this exhibit.

PAE-03M: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	● Informational	PriceAware.sol L23, L24

Description:

The linked declarations have no visibility specifiers explicitly set.

Recommendation:

We advise visibility specifiers to be defined for these variables as the current status is that they are automatically assigned by the compiler which can lead to compilation discrepancies between different versions.

Alleviation:

The `public` visibility specifier was added to both linked variables alleviating this exhibit.

PAE-04M: Redundant Duplicate Structure

Type	Severity	Location
Gas Optimization	● Informational	PriceAware.sol L152-L153

Description:

The `inverseLiquidationPath` variable is redundant as it can be processed by traversing the `liquidationPath` in inverse.

Recommendation:

We advise its omittance from the codebase and replacement with an inverse traversal path of `liquidationPath`.

Alleviation:

The MarginSwap - Core Implementation Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

PAE-05M: Variable Mutability Specifiers

Type	Severity	Location
Gas Optimization	● Informational	PriceAware.sol L17, L27

Description:

The linked variables are assigned to only once during the contract's `constructor`.

Recommendation:

We advise their mutability specifier to be set to `immutable` greatly optimizing the gas cost of the contract.

Alleviation:

The `peg` variable was properly set as `immutable` greatly optimizing the codebase.

RAE-01M: Variable Mutability Specifier

Type	Severity	Location
Gas Optimization	● Informational	RoleAware.sol L34

Description:

The `roles` variable is assigned to only once during the `constructor` of the contract.

Recommendation:

We advise the `immutable` mutability specifier to be introduced to the variable's declaration greatly optimizing the gas cost involved in utilizing it.

Alleviation:

The `roles` variable was properly set as `immutable` according to our recommendation.

RAE-02M: Redundant Declarations

Type	Severity	Location
Gas Optimization	● Informational	RoleAware.sol L9-L18, L20-L27

Description:

The linked declarations of the `RoleAware` contract are meant to represent the IDs of various roles of the system inefficiently so.

Recommendation:

Firstly, the `public` specifier for these variables is redundant and generates a lot of unnecessary bytecode as they do not need to be externally query-able. Additionally, these declarations can be replaced by `enum` constructs that are functionally equivalent. Lastly, the current paradigm does not allow a single address to possess two roles simultaneously which may be against what the system is meant to achieve.

Alleviation:

The declarations were relocated to the `Roles.sol` contract to be file-level declarations and the team added comments denoting why they opted to not utilize an `enum` thus alleviating this exhibit.

ROL-01M: Single Point of Failure

Type	Severity	Location
Logical Fault	● Minor	Roles.sol L6-L30

Description:

The `Roles` contract is the ultimate authority of the protocol, governing roles such as those able to withdraw capital from the system.

Recommendation:

We advise that the `Roles` contract is instead actuated on by a multi-signature implementation as utilizing a single `owner` address can have devastating effects to the overall protocol.

Alleviation:

The team has clarified that the `owner` of the contract will initially be the team's multisignature wallet and consequently will be the project's governance system thus alleviating this pain point and decentralizing the SPoF.

ROL-02M: Duplicate User-Specified Getter

Type	Severity	Location
Coding Style	● Informational	Roles.sol L7, L27-L29

Description:

The `roles` contract level variable of `Roles` is declared as `public` yet also has a user-specified getter function defined.

Recommendation:

We advise the user-specified getter to be removed as user-specified getters may come to be out-of-sync with the rest of the codebase.

Alleviation:

The MarginSwap - Core Implementation Contracts development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

ROL-03M: Inefficient Key Type

Type	Severity	Location
Gas Optimization	● Informational	Roles.sol L7, L8

Description:

The `mapping` declarations of the `Roles` contract utilize the `uint16` data type as a key argument inefficiently so.

Recommendation:

We advise that the argument is changed to a `uint256` as using a less-than-256-bit data type results in more gas utilized.

Alleviation:

The `mapping` key types were properly adjusted to be `uint256`.

RAE-01S: Unchecked Input Address

Type	Severity	Location
Logical Fault	● Minor	RoleAware.sol L33-L35

Description:

The `constructor` of the contract accepts an `address` argument that remains unchecked.

Recommendation:

We advise a zero-address check to be imposed on it to ensure that the system cannot be misconfigured.

Alleviation:

A zero-address check has been properly imposed on the input address.

TAN-01S: Variable Data Location Optimization

Type	Severity	Location
Gas Optimization	● Informational	TokenAdmin.sol L42

Description:

The `activateToken` function contains a `memory` array and is `external`.

Recommendation:

We advise the `memory` array to be set to `calldata` greatly optimizing the gas cost in utilizing it.

Alleviation:

The input argument was properly set to `calldata` optimizing the function's gas cost.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete` .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.