CERTIK

Security Assessment

Multiplex

Jun 1st, 2022

Table of Contents

Summary

Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

Findings

GLOBAL-01 : Financial Models

ACC-01 : Logical issue of function `depositCollateral()`

ACC-02 : Logic of funding fee

ACC-03 : Strategies for handling indebted users

ACC-04 : Code Simplify in function `_isAccountSafe()`

ADI-01 : Centralization Related Risks in liquidity

ADM-01 : High risks operations in function `setAssetParams()`

ADM-02 : Logical issue in function `setAssetFlags()`

COM-01 : Centralization Related Risks in components

CON-01 : Transactions are based on value rather than amount

CON-02 : Missing Emit Events

COR-01 : Centralization Related Risks in core

LAB-01 : Potential Out-of-Gas Exception

LIQ-01 : MlpPrice has upper and lower bounds

LIQ-02 : Logical issue of function `removeLiquidity()`

LMU-01 : Investment strategies

LRO-01 : Third Party Dependencies on Chainlink

LRO-02 : Logical issue of function `checkPrice()`

MTC-01 : Centralization Risks in MuxTimelockController.sol

OBB-01 : Third Party dependency - broker

OBB-02 : Logical issue about receiving native tokens

OBB-03 : Check-effects pattern not used

OBB-04 : Lack of input validation in function `placePositionOrder`

OBB-05 : Logic issue on liquidity fee

ORD-01 : Centralization Related Risks in orderBook

ORD-02 : Improper Usage of `public` and `external` Type

TOK-01 : Initial Token Distribution

TRA-01 : Logic issue on entryPrice in function `openPosition`

TRA-02 : Logic issue on function `_isAccountSafe()`

Appendix

Disclaimer

About

Summary

This report has been prepared for Multiplex to discover issues and vulnerabilities in the source code of the Multiplex project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Multiplex
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/multiplex-cool/mux-protocol
Commit	f3afcac089cdd7546a73f45b02fa2de3b15aca83

Audit Summary

Delivery Date	Jun 01, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0	0
● Major	7	0	0	6	0	0	1
● Medium	5	0	0	3	0	0	2
● Minor	7	0	0	3	0	0	4
● Informational	10	0	0	4	0	0	6
● Discussion	0	0	0	0	0	0	0

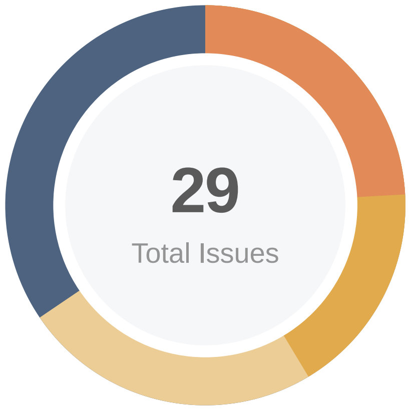
Audit Scope

ID	File	SHA256 Checksum
LPB	core/LiquidityPool.sol	2716da739cb7b184c6e0859801ec1a9264b5a14a5a01425680761b9c56ff1e87
SOB	components/SafeOwnable.sol	9ce96df19e1d12fc60d0e50694a23895867112bb31900e46537d39ea7bd9bceb
LMB	libraries/LibMath.sol	2799a047364e7a92e73c055c93d39fc47d25dda2bb12ea61017fe6c8e3d0e377
ADM	core/Admin.sol	99dcd003e8164735e162223dad7e0fc5af7bc1c84a6d57606a70aaaedaa1ec96
GOV	governance	
LIQ	core/Liquidity.sol	809549e6e7eaab6f85ee8fc9849c5e1606a86bc8ba8fa5cf47f90a81e72f0161
TYP	core/Types.sol	c1363bc3e0c63458047eff97f07b35b7f38149e8d3eb2647395d70e5d0ecb891
MTB	tokens/MlpToken.sol	1e7ab9b145e7fb2a838e6d14838f7f54bd79778a9dab468ec567cb3242f4c962
LIU	core/LiquidityPoolHop2.sol	2a24f509ede1cc89570bfe99675bc083cb7018c4c5cf9f49903216a7ac847888
ADN	orderbook/Admin.sol	3d593346472097db58ab9ff19b65f6998d2923b04971b2bf2330642c5aab4f61
COM	components	
OBB	orderbook/OrderBook.sol	a9dea41d09e4166bcd545b4a528f752f0306704499a5de10230d61b3c5c65d27
LMU	liquidity/LiquidityManager.sol	3c5a6af14b7a6a7523538192eb3eacc714de63909860c55b308f3dbaf4aede18
MTU	tokens/MuxToken.sol	6f5c1275d1b77697c0a6583c994de6f11404197540d9afacc520dacdcb66edee
TYS	orderbook/Types.sol	113339775cab5507f899b535d2be48f92cbcd943d8d40ad6ea5e03fe2f85764f
MCB	liquidity/ModuleCall.sol	0410b7ca1793f2f12a19ba8f265bd19da03a718265295006479 31a6e33798114
COR	core	
LAB	libraries/LibAsset.sol	4326c22f2a9a966c2d08642d7694fe18a1caa6c9ef3cadf0e757369aa9834b90

ID	File	SHA256 Checksum
		99c278114b3edcbafde1192f5a34ff44cc3bb4d203a36513fe4fe282231154bf
GET	core/Getter.sol	
TYE	liquidity/Types.sol	855887d3207e3450f46d81b4d4c2829ce727237a33230af7ffb19fbdd465a6e5
MTC	governance/MuxTimelockController.sol	67709ccbbfa790317aac170838a846a092f13ca6404660da8efeab92f6c45969
EVE	core/Events.sol	d6d72247f13eacc3ee449feca793c1303062afba67a1976a039cd7e92538aa0b
ORD	orderbook	
ACC	core/Account.sol	2a0cf093752487789c9b6a3239d42684073fd10977e5aa3aeb94fd5430a64d90
LPH	core/LiquidityPoolHop1.sol	2a83c5779d6c921102eb304b256f9ce72e6f93903eafcb05751843d073881b7f
LUB	libraries/LibUtils.sol	9f51392b713a7f4fbb48f618578f45cc788722d7de0c77d5143e57b535712342
ADI	liquidity/Admin.sol	c154b6911d024a319e275ed8d1bfe5b4ac4b94217d0d2341497a141e41541b82
LOB	libraries/LibOrder.sol	24442d5af121f3036211e47215506041040692a9bffaa48f75bed1ea0ef13690
LRO	libraries/LibReferenceOracle.sol	4d7106ee54680ba6e708d9ac48459fd1910b97db90121d47e0bc2c61050508ea
LSA	libraries/LibSubAccount.sol	5b040954d1887571b055c64090c94bbe863b4d29c054cacd237fbe00ece2291a
LCP	libraries/LibChainedProxy.sol	d0215a0fa326955389d3d8a403a29dc3e328cd8a31c5cced4ced27d135ee7e44
TOK	tokens	
STA	orderbook/Storage.sol	f401ccaf5d7cc2c3df24698ce6c624d9ae94a63ff2aefd5f3e2bb28b4985f48c

ID	File	SHA256 Checksum
LIB	libraries	
SOU	components/SafeOwnableUpgradeable.sol	033bab3405f336dd9d49282e56a72652cb87fd2d7dab9fdd1c15c055edc7f3db
STR	liquidity/Storage.sol	2d8e7226f3be402486f8ef13a0d5c2b2f9dff29831ced4721ef2fa0238834295
TRA	core/Trade.sol	f249261c7cc13c3ae28ce34fe52ccbd2789302ef93f7ae563da1fc46270f2b8b
STO	core/Storage.sol	47fe42bd172d586dd8086b6601b68688d90b0ca9da454ce1c4c28953e2909c12

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**7** (24.14%)
🟨 **Medium**	**5** (17.24%)
🟨 **Minor**	**7** (24.14%)
🟦 **Informational**	**10** (34.48%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Financial Models	Logical Issue	● Medium	ⓘ Acknowledged
ACC-01	Logical Issue Of Function `depositCollateral()`	Logical Issue	● Medium	✓ Resolved
ACC-02	Logic Of Funding Fee	Logical Issue	● Informational	✓ Resolved
ACC-03	Strategies For Handling Indebted Users	Control Flow	● Informational	ⓘ Acknowledged
ACC-04	Code Simplify In Function `_isAccountSafe()`	Coding Style	● Informational	✓ Resolved
ADI-01	Centralization Related Risks In Liquidity	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
ADM-01	High Risks Operations In Function `setAssetParams()`	Logical Issue	● Major	✓ Resolved
ADM-02	Logical Issue In Function `setAssetFlags()`	Control Flow	● Medium	ⓘ Acknowledged
COM-01	Centralization Related Risks In Components	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
CON-01	Transactions Are Based On Value Rather Than Amount	Logical Issue	● Minor	✓ Resolved
CON-02	Missing Emit Events	Coding Style	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
COR-01	Centralization Related Risks In Core	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
LAB-01	Potential Out-of-Gas Exception	Logical Issue	● Minor	✓ Resolved
LIQ-01	MlpPrice Has Upper And Lower Bounds	Logical Issue	● Minor	ⓘ Acknowledged
LIQ-02	Logical Issue Of Function `removeLiquidity()`	Logical Issue, Control Flow	● Minor	✓ Resolved
LMU-01	Investment Strategies	Coding Style	● Informational	ⓘ Acknowledged
LRO-01	Third Party Dependencies On Chainlink	Volatile Code	● Minor	ⓘ Acknowledged
LRO-02	Logical Issue Of Function `checkPrice()`	Logical Issue	● Minor	ⓘ Acknowledged
MTC-01	Centralization Risks In MuxTimelockController.sol	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
OBB-01	Third Party Dependency - Broker	Logical Issue	● Major	ⓘ Acknowledged
OBB-02	Logical Issue About Receiving Native Tokens	Logical Issue	● Medium	✓ Resolved
OBB-03	Check-effects Pattern Not Used	Logical Issue	● Minor	✓ Resolved
OBB-04	Lack Of Input Validation In Function `placePositionOrder`	Logical Issue	● Informational	✓ Resolved
OBB-05	Logic Issue On Liquidity Fee	Logical Issue	● Informational	✓ Resolved
ORD-01	Centralization Related Risks In OrderBook	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
ORD-02	Improper Usage Of `public` And `external` Type	Gas Optimization	● Informational	✓ Resolved
TOK-01	Initial Token Distribution	**Centralization / Privilege**	● **Medium**	ⓘ Acknowledged
TRA-01	Logic Issue On EntryPrice In Function `openPosition`	Logical Issue	● Informational	ⓘ Acknowledged

ID	Title	Category	Severity	Status
TRA-02	Logic Issue On Function `_isAccountSafe()`	Logical Issue	● Informational	ⓘ Acknowledged

GLOBAL-01 | Financial Models

Category	Severity	Location	Status
Logical Issue	● Medium		ⓘ Acknowledged

Description

There are two main roles in the protocol.

- Traders, pledge assets to open positions to gain profits.
- Liquidity providers, provide funds in the pool to gain fees and traders' losses.

The program is scheduled by the broker, the broker provides the price of the asset or lpToken to determine the users' profits or positions. And the admin of the protocol can use the funds in the pool to earn profits outside.

The overall profit strategy and pool funding allocation are determined by the operations of the team. The project party needs to maintain sufficient pool funds through various settings of the project to protect both parties' profits.

Recommendation

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

Alleviation

The team acknowledged the issue and stated the following:

"The power of moving assets to different chains is limited. The design is a balance between product utility and data alignment across all chains.

There will be multisig controlled to avoid single point management and timelock to delay operations."

ACC-01 | Logical Issue Of Function `depositCollateral()`

Category	Severity	Location	Status
Logical Issue	● Medium	core/Account.sol: 36	✓ Resolved

Description

The function does not record the entryFunding of the subAsset when it is first opened. If the variable is not initialized, the user has to pay the fee calculated with the entire cumulativeFundingRate.

Recommendation

We recommend the team check the logic and fix the issue.

Alleviation

The team stated that the variable is initialized when opening a position and it is indeed as they say that fundingFee is correctly calculated in all cases.

Below are their explanations:

"A trader calls depositCollateral first, and calls a function that calculates _getFundingFeeUsd. The result will be A new trader has entryFunding = 0 and size = 0. So funding = (cumulativeXX - entryFunding) * size = 0. It is always 0, so entryPrice is useless. An old trader keeps entryFunding as the previous value when calling depositCollateral. Only subAccount.collateral increases. This is also correct. When calling _getFundingFeeUsd next time, the trader will also get fundingFee for the entire time span.

By the way, we decided to record entryFunding when opening a position instead of depositing, because once the entryFunding is modified, the fundingFee must be charged immediately. In depositCollateral we can only charge the fee from collateral. We tend to deduct fees from the profit first, which is much more complicated (see withdrawProfit). So ignoring fees in depositCollateral is simple and effective.

Tldr; a subAccount has 2 states: size =0 and size != 0. When size = 0, entryFunding is useless because fundingFee = 0. When size != 0, entryFunding is important and we initialized it when opening this position."

ACC-02 | Logic Of Funding Fee

Category	Severity	Location	Status
Logical Issue	● Informational	core/Account.sol: 201	✓ Resolved

Description

The funding fee is controlled by the broker off-chain to decrease the impacts brought by some assets with stable development trends. If some asset's price is stably increasing or decreasing, the funding fee is used to bring more cost considerations to those positions with a clear trend of lopsidedness.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design. And it's recommended to elaborate on the functionality of the funding fee rules because the funding rate is updated by the broker.

Alleviation

The team acknowledged the issue and stated the following:

"The funding rate is designed to be proportional to the utilization of the liquidity pool across multiple chains. We programmed this rule into the smart contract (_getFundingRate in Liquidity.sol) to improve the transparency.

Liquidity is distributed across multiple chains so smart contracts cannot know the utilization. Broker provides this information which is auditable (because liquidity and positions are observable across all chains)."

ACC-03 | Strategies For Handling Indebted Users

Category	Severity	Location	Status
Control Flow	● Informational	core/Account.sol: 86	ⓘ Acknowledged

Description

If the collateral cannot afford the fee, the sub account will be in debt. If the user choose not to add the collateral, the fee still keeps increasing due to the increasing funding rate. But it does not affect that the user invests with other sub accounts.

Recommendation

We would like to be clear about the strategy of handing indebted users. It's recommended to elaborate on the issue.

Alleviation

The team acknowledged the issue and stated the following:

"The SubAccount will be liquidated by a Keeper when (Collateral + PNL < Maintenance margin). In most cases `the collateral cannot afford the fee` will not happen because PositionFeeRate is smaller than MaintenanceMarginRate. Even if it happened, one SubAccount will not affect other SubAccounts. Because LiquidityPool is the only counterparty of all Traders. LiquidityPool earns less money than normal if the keeper is too late to liquidate."

ACC-04 | Code Simplify In Function `_isAccountSafe()`

Category	Severity	Location	Status
Coding Style	● Informational	core/Account.sol: 182~188	✓ Resolved

Description

The codes mentioned can be simplified as:

```
if(hasProfit)
  return collateralUsd + pnlUsd >= thresholdUsd;
else
  return collateralUsd >= thresholdUsd + pnlUsd;
```

Because when `collateralUsd < pnlUsd`, the value of the expression `collateralUsd >= thresholdUsd + pnlUsd` is false.

Recommendation

We recommend the team change the code to make the account safety concept more clear.

Alleviation

The team heeded our advice and fixed the issue in commit 7713809ba5a0578621f4ea5d743963cbf4609f5e.

ADI-01 | Centralization Related Risks In Liquidity

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	liquidity/Admin.sol	ⓘ Acknowledged

Description

In the contract `Admin`, the role `owner` has authority over the following functions:

- function addExternalAccessor() and removeExternalAccessor(), to add or remove an accessor.
- function addDexSpotConfiguration(), to add a configuration for dex.
- function setDexWeight(), to modify the weight of a dex configuration.
- function installGenericModule(), to install a generic module.
- function installDexModule() and uninstallDexModule, to install or uninstall a dex module.

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and bring unpredictable damages to the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign ($2/3$, $3/5$) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged the issue and stated the following:

"The owner of the ‘LiquidityManager’ will have a 48-hours timelock controlled by a multi-signature wallet.

Once a strategy is added to the manager by the time lock, what that strategy can do will be restricted by the smart contract, preventing it from performing operations other than those expected.

In the future, functions such as adding a configuration of dex will be governed by veMUX holders."

ADM-01 | High Risks Operations In Function `setAssetParams()`

Category	Severity	Location	Status
Logical Issue	● Major	core/Admin.sol: 41~42	✓ Resolved

Description

The owner of the contract `core/Admin` has the authority to change the `initialMarginRate` and `maintenanceMarginRate` of the asset. The two attributes are very sensitive. If the two values are raised, many `subAccounts` are at risk of being forced into liquidation.

Recommendation

We recommend the team check the logic and elaborate on the issue.

Alleviation

The team heeded our advice and resolved this issue in commit `0dbdf5d8ad05264bf25e7719eafc768dee37b274`.

ADM-02 | Logical Issue In Function `setAssetFlags()`

Category	Severity	Location	Status
Control Flow	● Medium	core/Admin.sol: 41~42, 77~81, 139, 154	ⓘ Acknowledged

Description

Ideally, the attributes of the asset are supposed to keep unchanged. If the attributes like `isTradable`, `isOpenable`, `isShortable` are changed, the inventory transactions are affected.

Recommendation

We recommend the team elaborate on the strategies of migrating the status of an asset while protecting users' transactions.

Alleviation

The team acknowledged the issue and stated the following:

"The attributes of setAssetFlags() – especially isEnable – will be mainly used for system maintenance needs, such as adding a new blockchain, or suspending trading when an emergency security issue occurs."

COM-01 | Centralization Related Risks In Components

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	components/SafeOwnable.sol; components/SafeOwnableUpgradeable.sol	ⓘ Acknowledged

Description

In the contract `SafeOwnable` and `SafeOwnableUpgradeable`, the role `owner` has authority over the following functions:

- function transferOwnership(), to transfer ownership of the contract.
- function renounceOwnership(), to renounce the ownership.

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and modify contract ownership, call admin functions to cause serious damage to the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged the issue and stated the following:

"When launching, Timelock will be the contract owner and it is controlled by multisign, to avoid single point management risks."

CON-01 | Transactions Are Based On Value Rather Than Amount

Category	Severity	Location	Status
Logical Issue	● Minor	core/Liquidity.sol; core/Account.sol; core/Trade.sol; orderbook/OrderBook.sol	✓ Resolved

Description

All the transactions are based on the USD value of the assets. So the transactions and subAccount safety statuses are all affected by the price of assets, collaterals, muxToken and mlpToken. If these token prices are unstable, all transactions need caution.

Recommendation

We recommend the team check the logic and consider its impacts.

Alleviation

The team acknowledged the issue and stated the following:

"Different from spot swap protocol, the value of the assets can only be determined by the spot price, which needs to be derived from oracle. In a margin trading protocol, price is the critical factor to make the system work."

CON-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	orderbook/OrderBook.sol: 381, 407; liquidity/LiquidityManager.sol: 57, 62; core/LiquidityPoolHop1.sol: 36	✓ Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The team heeded our advice and fixed the issue in commit b400b7c49302d9742093578a3c32f2e74eb70d0c.

COR-01 | Centralization Related Risks In Core

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	core/Account.sol; core/Admin.sol; core/Liquidity.sol; core/LiquidityPoolHop1.sol; core/Trade.sol	ⓘ Acknowledged

Description

In the contract `Account`, the role `orderBook` has authority over the following functions:

- function depositCollateral(), to add the collateral amount for an arbitrary account.
- function withdrawCollateral() and withdrawAllCollateral(), to withdraw collateral for an arbitrary account.

AND

In the contract `Admin.sol`, the role `owner` has authority over the following functions:

- function addAsset(), to add a kind of asset.
- function setAssetParams(), setFundingParams() and setAssetFlags(), to set params for the asset.
- function pauseAll(), to disable all assets.
- function setReferenceOracle(), to set referenceOracle for the asset.
- function setNumbers(), to set fundingInterval, mlpPriceBounds, liquidityFeeRate.

The role `liquidityManager` has authority over the following functions:

- function transferLiquidityOut() and transferLiquidityIn(), to transfer liquidity.

AND

In the contract `Liquidity`, the role `orderBook` has authority over the following functions:

- function addLiqudity() and removeLiquidity(), to add or remove liquidity for the liquidity pool.
- function redeemMuxToken(), to redeem muxtoken into original tokens.
- function updateFundingState(), to update funding.

AND

In the contract `LiquidityPoolHop1`, the role `owner` has authority over the following function:

- function upgradedChainedProxy(), to change the chained proxy.

AND

In the contract `Trade`, the role `orderBook` has authority over the following functions:

- function openPosition() and closePosition(), to change the position for the subAccount.
- function liquidate(), to liquidate the position for the subAccount.
- function withdrawProfit(), to withdraw profits for the subAccount.

Any compromise to the priviledged accounts may allow a hacker to take advantage of this authority and bring unpredictable damages to the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged the issue and stated the following:

"The role `orderBook` will be the OrderBook contract and it cannot be modified. When launching, Timelock will be contract owner and it is controlled by multisign, to avoid single point management risks."

LAB-01 | Potential Out-of-Gas Exception

Category	Severity	Location	Status
Logical Issue	● Minor	libraries/LibAsset.sol: 25	✓ Resolved

Description

The lib `LibAsset` needs to interact with the `WETH` third party protocol. It will transfer the chain native token to the contract with a limit of 2300 gas in L25. This requires the final version of the contract's proxy not to have plenty of gas-consuming logic in the function `receive()`, otherwise the transfer will fail.

Recommendation

We recommend the team elaborate on the details of deployment and mind the detail when deploying it.

Alleviation

The team heeded our advice and resolved this issue in commit `664e9186dc425c0a0c96a15ecc212397c83a2b72`.

They added components/NativeUnwrapper.sol so that native tokens are withdrawn into an un-upgradable contract.

LIQ-01 | MlpPrice Has Upper And Lower Bounds

Category	Severity	Location	Status
Logical Issue	● Minor	core/Liquidity.sol: 41~42, 91~92	ⓘ Acknowledged

Description

As per the mlpPrice is set by the broker, and the price will have a range. So the users who want to add the liquidity, might frequently add or remove liquidity according to the price fluctuations to maximize the profits. This might lead to an unstable volume of the lp pool.

Recommendation

We recommend the team check the logic and elaborate the mechanism of the price range.

Alleviation

The team acknowledged the issue and stated the following:

"A Liquidity Provider can not frequently add or remove liquidity because there is a liquidityLockPeriod which is a delay between placeLiquidityOrder and fillLiquidityOrder. This lock is used to stop harmful high-frequency trading on MLP.

The price range, on the other hand, is used to protect against broker failures."

LIQ-02 | Logical Issue Of Function `removeLiquidity()`

Category	Severity	Location	Status
Logical Issue, Control Flow	● Minor	core/Liquidity.sol: 123~126	✓ Resolved

Description

The logic of the function `removeLiquidity()` default returns the MUX token when the balance of the pool is insufficient to cover the withdrawal. And it does not take the users' requirements into consideration. It's recommended to add a parameter as the switch of the operation and optionally use MUX token to pay debts.

Recommendation

We recommend the team check the logic and fix the issue.

Alleviation

The team heeded our advice and fixed the issue in commit 813a8d33d454208f5b887d0e88b39daa929a39a6.

And they also stated the following:

"Function removeLiquidity() now no longer sends MuxTokens. The Liquidity Provider shall bridge MLP into blockchains with sufficient liquidity."

LMU-01 | Investment Strategies

Category	Severity	Location	Status
Coding Style	● Informational	liquidity/LiquidityManager.sol	ⓘ Acknowledged

Description

The liquidityManager can use the funds in the pool to invest in other protocols. The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets.

The funds in the pool should be steadily increasing through the investment strategies.

Recommendation

We would like to be clear about the investment strategies of the team like what percent number of the funds will be used for investment and what kind of protocols are considered as investment targets.

Alleviation

The team acknowledged the issue and stated the following:

"The scalable investment strategy is one of the key features of MUX products.

We choose third-party platforms with considerable reliability to make investments and ensure LPs are aware of the investments.

The way the assets are allocated will be determined by the configurations (assetWeights) in LiquidityManager. The configurations are also managed by a timelock controller owned by the multi-signed wallet. Anyone is able to verify the liquidities with on-chain data."

LRO-01 | Third Party Dependencies On Chainlink

Category	Severity	Location	Status
Volatile Code	● Minor	libraries/LibReferenceOracle.sol	ⓘ Acknowledged

Description

The contracts loading the library `LibReferenceOracle` are serving as the underlying entities to interact with third-party chainlink protocols. The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets.

Recommendation

We understand that the business logic of LibReferenceOracle requires interaction with external chainlink. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue, and they will monitor the status of 3rd party Oracles.

LRO-02 | Logical Issue Of Function `checkPrice()`

Category	Severity	Location	Status
Logical Issue	● Minor	libraries/LibReferenceOracle.sol: 77	ⓘ Acknowledged

Description

According to the normal process, the price given by the broker should be truncated with the price of the chainlink on-chain. If the broker does not set oracle for certain assets, the price of these assets will not be restricted by chainlink prices.

Recommendation

We would like to confirm with the client whether if all the assets have the matched oracles and what tokens will be added into the asset list.

Alleviation

The team acknowledged the issue and stated the following:

"We will set third-party Oracles for more assets if they support it."

MTC-01 | Centralization Risks In MuxTimelockController.sol

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	governance/MuxTimelockController.sol: 67, 78	ⓘ Acknowledged

Description

In the contract `MuxTimelockController` the role `EXECUTOR_ROLE` has authority over the following function:

- function `executeQuickPath()`, to perform an external call.

AND

In the contract `TimelockController`, the role `EXECUTOR_ROLE` ha authority over the following functions:

- function execute() and executeBatch(), to execute instructions.

The role `PROPOSER_ROLE` has authority over the following functions:

- function schedule() and scheduleBatch(), to make schedules.

The role `CANCELLER_ROLE` has authority over the following functions:

- function cancel(), to cancel an operation.

The role `TIMELOCK_ADMIN_ROLE` or `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- function grantRole() and revokeRole(), to manage roles.

Any compromise to the privileged accounts may allow the hacker to take advantage of this authority and bring unpredictable damages to the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

The team acknowledged the issue and stated the following:

"When launching, timelock will be contract owner and it is controlled by multisign, to avoid single point management risks."

OBB-01 | Third Party Dependency - Broker

Category	Severity	Location	Status
Logical Issue	● Major	orderbook/OrderBook.sol: 271	ⓘ Acknowledged

Description

Usually the lptoken represents the share of the user takes in a liquidity pool. And the users can share the balance of the entire pool equally according to his/her shares.

According to the logic of the project, the amount of the tokens that users can get back depends on the given mlpPrice. The mlpPrice is a parameter of the function `fillLiquidityOrder()`, which means that the mlpPrice is determined by the broker. This part is out of the audit scope.

Recommendation

We recommend the team check the logic and elaborate on the pricing mechanism of the project.

Alleviation

The team acknowledged this issue and they stated:

"Liquidity is distributed across multiple chains so smart contracts cannot know the utilization. Broker provides this information which is observable across all chains.

To minimize trust, a decentralized network, which is operated by the veMUX holders, will provide the multiplexing layer services, including Oracle, in the future."

OBB-02 | Logical Issue About Receiving Native Tokens

Category	Severity	Location	Status
Logical Issue	● Medium	orderbook/OrderBook.sol: 437	✓ Resolved

Description

The place orders functions are entrances to receive users' funds. If users transferred ERC20 tokens into the contract intentionally, in the meantime the user could incorrectly transfer native tokens as well via this payable function.

Then these native tokens should be returned to the user, or the program should be rolled back and thrown the error messages.

Recommendation

We recommend the team add checks to prevent users' error payments.

Alleviation

The team heeded our advice and fixed the issue in commit f21c111591cfa90dd62afaf5b9cc39d6ec9e724c.

OBB-03 | Check-effects Pattern Not Used

Category	Severity	Location	Status
Logical Issue	● Minor	orderbook/OrderBook.sol: 113	✓ Resolved

Description

State variables are changed after the transfer or external call, which might leads to a re-entrancy issue.

Recommendation

It is recommended to follow checks-effects-interactions pattern for cases like this. It shields public functions from re-entrancy attacks. It's always a good practice to follow this pattern. `checks-effects-interactions` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

Refer https://docs.soliditylang.org/en/develop/security-considerations.html?highlight=check-effects%23use-the-checks-effects-interactions-pattern

Alleviation

The team heeded our advice and fixed the issue in commit e7b3b2658ebb51b8de5a21c07e70018d7de38623.

OBB-04 | Lack Of Input Validation In Function `placePositionOrder`

Category	Severity	Location	Status
Logical Issue	● Informational	orderbook/OrderBook.sol: 103, 105	✓ Resolved

Description

1. When the limit price is set, the `POSITION_MARKET_ORDER` flag in the parameter flags should be true to support the check.
2. When the profitTokenId is in use, the `POSITION_INCREASING` flag in the parameter flags should be false to close position.

Recommendation

We recommend the team add the check in the function to reduce program runtime uncertainty.

Alleviation

The team heeded our advice and fixed the issue in commit 52f936e9204c97eafe2065829f142e56e7a15681.

OBB-05 | Logic Issue On Liquidity Fee

Category	Severity	Location	Status
Logical Issue	● Informational	orderbook/OrderBook.sol: 272~273	✓ Resolved

Description

As is mentioned in the comments above the function, the currentAssetValue is the liquidity USD value of a single asset and the targetAssetValue represents `weight / Σ weight * total liquidity USD value`. And the fee will be increased or decreased according to the two values.

Recommendation

We recommend the team elaborate on the rules of liquidity fee.

Alleviation

The team heeded our advice and add more explanations in commit ab4c3dc411136286d872f34ce5ce7d40ec5e2c8a.

ORD-01 | Centralization Related Risks In OrderBook

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	orderbook/Admin.sol; orderbook/OrderBook.sol	ⓘ Acknowledged

Description

In the contract `Admin`, the role `owner` has authority over the following functions:

- function addBroker() and removeBroker(), to add or remove a broker.
- function setLiquidityLockPeriod(), to change the liquidity lock period.

AND

In the contract `OrderBook`, the role `broker` has authority over the following functions:

- function fillPositionOrder(), to fill a position order.
- function fillLiquidityOrder(), to fill a liquidity order.
- function fillWithdrawalOrder(), to fill a withdrawal order.
- function cancelOrder(), to cancel an order.
- function updateFundingState(), to update funding state.
- function liquidate(), to liquidate assets for the accounts.

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and bring unpredictable damages to the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged the issue and stated the following:

"When launching, timelock will be contract owner and it is controlled by multisign, to avoid single point management risks. To minimize trust, a decentralized network, which is operated by the veMUX holders, will provide the broker service in the future."

ORD-02 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	● Informational	orderbook/Admin.sol: 16, 22, 26; orderbook/OrderBook.sol: 64	✓ Resolved

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

The team heeded our advice and fixed the issue in commit 8ba28607449de0c800de49fa9363efa39388295f.

TOK-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	● **Medium**	tokens/MlpToken.sol: 12; tokens/MuxToken.sol: 12	ⓘ Acknowledged

Description

All of the `MlpToken`/`MuxToken` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute `MlpToken` tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

The team acknowledged this issue and they stated the following:

"We have to pre-mint MlpToken/MuxToken for technical reasons. Traders can bridge MlpToken/MuxToken across multiple chains. In “mint & burn model” (https://cbridge-docs.celer.network/introduction/canonical-token-bridge), PeggedToken can only be minted by bridge instead of LiquidityPool. So we transferred pre-minted MlpToken/MuxToken into LiquidityPool of each chain.

The unused pre-minted MlpToken/MuxToken will be saved in a multi-sign contract for future usage."

TRA-01 | Logic Issue On EntryPrice In Function `openPosition`

Category	Severity	Location	Status
Logical Issue	● Informational	core/Trade.sol: 63~64	ⓘ Acknowledged

Description

According to the logic in function _positionPnlUsd(), when the user's profit has not reached the minProfitRate and the time is within minProfitTime, the user has to give up the original entryPrice to increase the position.

Recommendation

We would like to confirm with the client if the current implementation aligns with the project design.

Alleviation

The team acknowledged the issue and stated that it does align with the project design.

TRA-02 | Logic Issue On Function `_isAccountSafe()`

Category	Severity	Location	Status
Logical Issue	● Informational	core/Trade.sol: 338	ⓘ Acknowledged

Description

The operations `openPosition`, `depositCollateral` and `withdrawProfit` require the sub account is safe in the initialMarginRate range. The operation `closePosition` requires the safe is safe in the maintenanceMarginRate range.

Recommendation

We would like to be clear about the functionality of the two rates. It's recommended to elaborate on the two rates.

Alleviation

The team acknowledged the issue and stated the following:

"OpenPosition, withdrawCollateral, withdrawProfit requires IM safe. depositCollateral does not need to meet any safe conditions. closePosition will succeed unless MM-unsafe which should be liquidated."

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings 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.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Coding Style

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

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

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

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

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

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

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK