SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

LemonX Trade

AUDIT

SECURITY ASSESSMENT

04. August, 2024

FOR

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement"or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	LemonX Trade
Website	https://www.lemonx.trade/
About the project	LemonX combines the essentials of perpetual trading, offering upto 50x leverage, deep liquidity, and the lowest fees, all wrapped up in a platform that operates at lightning speed.
Chain	TBA
Language	Solidity
Codebase	https://github.com/lemonx-trade/contracts
Forked Status	This project is inspired from GMX contract; here is the link to the contracts: https://github.com/gmx-io/gmx-contracts/tree/master/contracts
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/lemonx_trade
Twitter	https://twitter.com/lemonx_trade
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	https://discord.com/invite/esS6rZxRUP
YouTube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	04. August 2024	· Layout Project · Automated/ Manual-Security Testing · Summary

Note – The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts\core\BaseOrderManager.sol	e5aaa1380d634650b1fef83b50d470d025120ee4
contracts\access\Governable.sol	35ffa6979c86dc839d9126aded63d382372f1103
contracts\core\LemonXUSDC.sol	81acaf434f530ac6722d1ef8fbaaf478190b6601
contracts-main-3\contracts-main\contracts\core\LLP.sol	9d8dd8feec040be16364e07932058d8095004ff7
contracts\libraries\token\Address.sol	3f9b1cf61d71b0ff1ea354f9270feb0898440b4b
contracts\libraries\token\BaseToken.sol	2ed913348fd95f27a05df557e02f3824cb60ef77
contracts\core\YieldToken.sol	c5bafedb8088518c8206b1986b87092d13309f5c
contracts\core\Vault.sol	f0f4f739f4d02101b9fc006d3987c41156ef5f97
contracts\core\Utils.sol	38e0acabd6ee139bc4bf083f35b4ca91d1e51c1a
contracts\core\USDL.sol	bc195b08a793d407c2a8fae7415762d9b4d1fc27
contracts\core\TimeLock.sol	42790fbb6381188b98430626a24765a8dcb84500
contracts\core\TierBasedTradingFees.sol	5aa82ced66a668164a2722a23fd3f2a54e5a2e5c
contracts\core\RewardTracker.sol	59f0ca10be30e46a27fd3a9c14bc7592ee94cd63
contracts\core\RewardRouter.sol	8a82b7bc35b46344c5de64bf11f9690a6bc9c221
contracts\core\ReaderContract.sol	f39da63674dae9d74048b3060444fd31adba53c7
contracts\core\ReaderCache.sol	ab95a70e7a73156ab1344fee4dc24d55fb0e9856
contracts\core\PythPriceFeed.sol	7db54b7c9d40b27891496560263a60a5415b5efb
contracts\core\PriceFeed.sol	022a8b573db5a3ca9a3251482b6395342259a622
contracts\core\OrderManager.sol	7ef16492b860c99c3a82086a23c392e8997e15a2
contracts\core\MultiCall.sol	3e36f9a5cb69a508246f5efb374392d66ce26c91
contracts\core\LLPManager.sol	acdaef900989d551261c6b398b3d978bbb263438
contracts\core\interfaces\IPriceEvents.sol	9264157c9f6c581b7dd9f74c3662495814b1472f
contracts\core\interfaces\IPriceFeed.sol	5f82c9b816755b8668db1a4ed4014bbbe64f70a3
contracts\core\interfaces\IOrderManager.	9b57663fcbd9a8bb9a843291fff7818be8dc03c0

File Name	SHA-1 Hash
sol	
contracts\core\interfaces\IMintable.sol	330400f404776cd5af262fc69c5844bdbb4168e2
contracts\libraries\utils\Structs.sol	1c74c39a62869450a8b59c85b19b5c591f216cf8
contracts\core\interfaces\ILlpManager.sol	292061ef5804ffd832b04c34d756cedad9ab94c7
contracts\libraries\token\SafeERC20.sol	f6c3a5ef8b2fc7abc040066e245473e642b52635
contracts\libraries\utils\ReentrancyGuard.sol	863720793f904c972d10c58685177f238d2ae711
contracts\core\interfaces\ILLP.sol	f97b05798a7d326fa2d93ca2cff77c8587604b8c
contracts\libraries\token\MintableBaseToken.sol	b442e12f26831d8e8b7d47714e49ab8fa8bc609e
contracts\core\interfaces\IRewardDistributor.sol	63b5da839826138cac2372c59bdd5693a35d693d
contracts\core\interfaces\IReaderContract.sol	53d2bd544a2d36abd126a841f6c3a79ce13517f2
contracts\core\interfaces\IHandler.sol	486c68afbc8dea5c5115bfb3627c3a9aea0ce44e
contracts\libraries\utils\EnumerableSet.sol	0e2b27c74e9000d1ed5cf363de116c72fe0fb9b0
contracts\libraries\token\IMintable.sol	b366595ae8722988bc3f67fd4678733a2b498725
contracts\core\interfaces\IRewardRouter.sol	f117d05f80cac11e2a04b1c35af570f6ecf51cb8
contracts\libraries\token\IERC20Permit.sol	c08b9d3aa94d69b7df86326a6b99d02339028317
contracts\core\interfaces\IAdmin.sol	0889c8befc49fb90dd43e52dfcfff4e381c0e6b9
contracts\libraries\token\IERC20.sol	8c8ac0fb212a25af32e5685164092dae200187bf
contracts\libraries\utils\Address.sol	0f82f6ae5f3854a426dd8584032daeec4e7cea35
contracts\libraries\token\IBaseToken.sol	a8dcae85e72c0584319405dd2d92cd512c1ce151
contracts\core\interfaces\IRewardTracker.sol	7f32c0529d3563f838b0715b398b5986bc1909e8
contracts\core\interfaces\IYieldToken.sol	61b5571808545d95c4a95b73ed8d3c8e4abdc99f
contracts\core\interfaces\IVault.sol	8855e8cff47914900e998b615ccec564988f4cce
contracts\core\interfaces\IUtils.sol	5a3cdd70ad303c80b552a96998588f947e77a234
contracts\core\interfaces\IUSDL.sol	9e3603373e3a9a1134dc4f733715f4c5b9b9717b

File Name	SHA-1 Hash
contracts\core\interfaces\ITimelockTarget.sol	2ead5e03c156ac72ad5b7b5a6897cefc1f693da7
contracts\core\interfaces\ITimeLock.sol	8e2d78aaebdfcfe5e7b6029470884ff29a0e2fed
contracts\core\interfaces\ITierBasedTradingFees.sol	6494b1f198fa0622acab69e71bfaab56d2f71818

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages.

Used code from other Frameworks/Smart Contracts.

Dependency / Import Path	Count
pyth-sdk-solidity/IPyth.sol	2
pyth-sdk-solidity/PythStructs.sol	2

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
21	5	24	1

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
548	14

External	Internal	Private	Pure	View
439	495	54	19	242

StateVariables

Total	🌐 Public
246	237

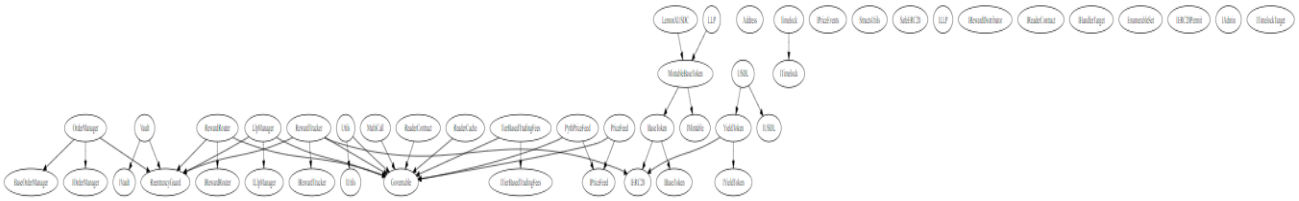
Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`^0.8.19` `0.8.19`	`----------`	`Yes`	`yes` (6 asm blocks)	`----------`

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`		`yes`	`yes`		

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the vulnerability and the impact of that event on the organization or system.The risk level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executingthe contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers andsmart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   a. Reviewing the specifications, sources, and instructions provided to
   SolidProof to ensure we understand the size, scope, and functionality of the
   smart contract.
   b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
   c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
   a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
   b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is not an upgradable	✅ **Deployer cannot update the contract with new functionalities.**
Description	The contract is not an upgradeable contract. The Deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

Contract ownership is not renounced.	❌ **The ownership is not renounced.**
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations.
Comment	N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner can mint new tokens.	❌ **The owner can mint new tokens.**
Description	The owner is able to mint new tokens once the contract is deployed.
Comment	The contract contains the functionality in which the governor can mint new tokens in the contract which is not recommended as this can change the supply of the tokens. There must be a maximum threshold amount in the contract so the tokens cannot be minted to an indefinite amount.

File/Line(s): L24-26

Codebase: MintableBaseToken.sol

```
function mint(address _account, uint256 _amount) external override onlyMinter {
    _mint(_account, _amount);
}
```

File/Line(s): L28-30

Codebase: USDL.sol

```
function mint(address _account, uint256 _amount) external override onlyVault {
    _mint(_account, _amount);
}
```

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner can burn tokens	❌ **The owner can burn tokens.**
Description	The owner is able burn tokens without any allowances.
Comment	The contract contains the functionality in which the owner can burn the tokens from other wallets without any allowance which is not recommended as this can cause the loss of funds for the user. There must be a check where only the allowed tokens from the contract should be burned from the wallets.

File/Line(s): L32-34
Codebase: USDL.sol

```
ftrace | funcSig
function burn(address _account1, uint256 _amount1) external override onlyVault {
    _burn(_account1, _amount1);
}
```

File/Line(s): L28-30
Codebase: MintableBaseToken.sol

```
ftrace | funcSig
function burn(address _account1, uint256 _amount1) external override onlyMinter {
    _burn(_account1, _amount1);
}
```

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner cannot blacklist addresses.	✅ **The owner cannot blacklist wallets.**
Description	The owner cannot blacklist wallets from transferring of tokens.
Comment	N/A

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort indeveloping and maintaining the contract.

Contract owner can levy high taxes	❌ **The owner can set fees more than 25%.**
Description	The owner can set fees of more than 25%.
Comment	The contract contains functionality in which the governor can update the fees to more than 100% which is not recommended as there should not be any fees in the contract that can be more than 25% in the contract.

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner can lock function.	❌ **The owner can lock function.**
Description	The owner can lock the contract.
Comment	The contract contains functionality in which the contract's admin can cease trading and liquidation activity in the vault contract for an indefinite period of time. This is not recommended as it can lock the contract permanently, and there is no time limit in which the trading and liquidity activity will auto-enable. There must be functionality in which the trading and liquidation activity cannot be locked up indefinitely.

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example,if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
Governable	➢ The governor can update the governor's address.
BaseOrderManager	➢ The admin can update the admin, vault, and utils address to any arbitrary address including zero. ➢ The admin can set any arbitrary value in the deposit fees in the contract. ➢ The admin can withdraw fees from the contract.
LLPManager	➢ The governor can update the utils contract address. ➢ The governor can add/remove the handler from the contract. ➢ The governor can whitelist/un-whitelist tokens in the contract. ➢ The governor can update any arbitrary amount in the max pool value in the contract. ➢ The owner can update the cool-down duration period to not more than 48 hours. ➢ The owner can update any arbitrary address as the vault, usdl, and llp contract address. ➢ The handler in the contract can add/remove liquidity for the account if the LP activity is not disabled by the governor.
Multicall	➢ The governor can whitelist addresses to aggregate and cancel orders. ➢ The whitelisted addresses can aggregate and cancel orders in bulk.
OrderManager	➢ The admin can update any arbitrary amount in the minimum purchase USD amount market and limit order. ➢ The admin can update the position keeper in the contract. ➢ The admin can update the max TP multiplier amount

	to any arbitrary amount. ➢ The admin can update any arbitrary amount in the min execution fee increase/decrease market and limit order value. ➢ The admin can update any arbitrary address as the price feed contract address. ➢ The admin can update the minimum block delay keeper, time delay, and max time delay to any arbitrary value. ➢ The admin can add/remove order keepers, and liquidators in the contract. ➢ The position keeper can increase/decrease positions in the contract. ➢ The admin can withdraw stuck tokens from the contract. ➢ The order keeper can execute orders in the contract. ➢ The liquidator can liquidate multiple positions in the contract.
Pricefeed	➢ The governor can update the order manager, and reward router contract address. ➢ The governor can add/remove updaters in the contract. ➢ The governor can add any arbitrary amount in the max allowed delay, max allowed delta, and slippage value in the contract. ➢ The governor can add price Ids to the tokens. ➢ The updater can set prices and execute orders in the contract. ➢ The updater can execute positions and orders in the contract. ➢ The updater can liquidate the position from the contract. ➢ The governor can withdraw funds from the contract. ➢ The updater can execute the LP requests from the contract.
PythPriceFeed	➢ The governor can update the order manager, and reward router contract address. ➢ The governor can add/remove updaters in the contract. ➢ The governor can update the pyth contract address.

	➢ The governor can add any arbitrary amount in the max allowed delay, and max allowed delta value in the contract. ➢ The updater can set prices and execute orders in the contract. ➢ The updater can execute positions and orders in the contract. ➢ The updater can liquidate the position from the contract. ➢ The governor can withdraw funds from the contract. ➢ The updater can execute the LP requests from the contract.
ReaderCache	➢ The governor can update the vault, utils, reader, LLP, and FLLP contract address.
ReaderContract	➢ The governor can update the Tier1, Tier2, Tier3, Tier1 factor, Tier2 factor, Tier3 factor. ➢ The governor can update the tier borrowing rate factor, funding factor for less and high OI side, Tier borrowing rate start time. ➢ The governor can update the vault, utils and USDC contract address.
RewardRouter	➢ The governor can update the feeLLptracker, LLP manager, LLP contract address. ➢ The governor can update the minimum execution fees in the contract. ➢ The governor can add/remove keepers in the contract. ➢ The governor can withdraw tokens from the contract. ➢ The keeper can execute the mint and burn requests.
RewardTracker	➢ The governor can initialize the contract only once. ➢ The governor can add/remove the depositing token. ➢ The governor can update any arbitrary amount in the reward precision. ➢ The governor can add/remove the handler from the contract. ➢ The governor can update any arbitrary amount in the cumulative reward per Lptokens value in the contract. ➢ The governor can withdraw tokens from the contract. ➢ The handler can stake/un-stake tokens for accounts.
TierBasedTradingFees	➢ The governor can add/remove the tier updater in the contract.

Timelock

- The tier updater can update the tier-based trading basis points for the trader.
- The admin can update the admin address.
- The admin can update the admin of other contract addresses.
- The admin can add/remove the contract handler.
- The admin can add/remove the keepers in the contract.
- The admin can update the buffer period in the contract to not more than 5 days.
- The admin can update the max leverage to not less than 500000.
- The keepers and admin update the max borrowing and funding rate factor to not more than 0.02%.
- The keepers and admins can update the token configuration in the contract in which the minimum profit BPS is set to 0.05%.
- The keeper and admin can mint and burn the tokens in the USDL contract.
- The admin can update the cool-down duration to not more than 2 hours.
- The admin can update the max global long, and short size in the vault contract.
- The admin can remove the admin from the yield token contract.
- The admin can update the utils in the vault contract.
- The admin can update the max gas price to not more than 5000000000 value in the vault contract.
- The admin can withdraw fees from the vault contract.
- The keepers and admins can withdraw fees from the vault contract in bulk.
- The admin can enable/disable the private liquidation mode in the vault contract.
- The admin can add a liquidator to the vault contract.
- The admin can transfer tokens to the contract address from other wallets.
- The admin can approve the spender to transfer tokens.
- The admin can signal to withdraw tokens from the contract.
- The admin can signal to update the governor in the

	contract. ➢ The admin can signal to update the handler in the contract. ➢ The admin can signal to update the price feed and update the price fees in the vault contract. ➢ The admin can signal to redeem USDL and redeem the USDL amount from the vault contract. ➢ The admin can update the configurations in the vault contract. ➢ The admin can cease the trading activity for an indefinite period of time. ➢ The admin can disable the LP activity in the contract for an indefinite period of time. ➢ The admin can clear all the ongoing actions in the contract.
USDL	➢ The governor can add/remove the vault in the contract. ➢ The vault contract can mint and burn an unlimited number of tokens from the contract.
Utils	➢ The governor can update tier1, tier2, and tier3 size in the contract. ➢ The governor can update the tier1factor, tier2factor, and tier3factor in the contract. ➢ The governor can update the tier borrowing rate factor to any arbitrary amount. ➢ The governor can update the funding factor for the less and high OI sides in the contract. ➢ The governor can update the tier-based trading fees contract address. ➢ The governor can update the vault, and price feed contract address. ➢ The governor can update the borrowing rate and funding rate precision in the contract. ➢ The governor can update the borrowing rate start time in the contract. ➢ The governor can update the maintenance margin in the token contract. ➢ The governor can update the token premium position fee in the contract.
Vault	➢ The governor can update the utils contract address.

	➢ The governor can update the new governor in the contract. ➢ The governor can disable trading activity. ➢ The governor can disable LP activity. ➢ The governor can update the order manager's contract address. ➢ The governor can update the USDL contract address. ➢ The governor can add the OI Balance threshold to the contract. ➢ The governor can update the liquidator in the contract. ➢ The governor can update the maximum gas price. ➢ The governor can update the price fees contract address. ➢ The governor can update the pool safety factor in BPS. ➢ The governor can update the maximum leverage. ➢ The governor can update the max global short size, global short limit, long size limit, and long size BPS. ➢ The governor can update the mint fee basis point, margin fees basis point, liquidation fee in USD, liquidator factor, and minimum profit time. ➢ The governor can update the borrowing and funding rates. ➢ The governor can update the token configuration. ➢ The governor can withdraw fees from the contract.
YieldToken	➢ The governor can update the governor's address. ➢ The governor can update the name and symbol of the contract. ➢ The governor can add/remove admins from the contract. ➢ The governor can withdraw tokens and send them to the wallets. ➢ The owner can enable/disable whitelist mode. ➢ The governor can update the whitelist handler in the contract. ➢ The admin can add/remove the non-staking account.
BaseToken	➢ The governor can update the governor's address. ➢ The governor can update the name and symbol of the contract. ➢ The governor can add/remove admins from the

MintableBaseToken	contract. ➢ The governor can withdraw tokens and send them to the wallets. ➢ The governor can enable/disable private transfer mode. ➢ The governor can add/remove the handler in the contract. ➢ The governor can add/remove minters in the contract. ➢ The minter can mint an unlimited number of tokens to any arbitrary address. ➢ The minter can burn tokens from other wallets without any allowance.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

Critical Issues

No critical issues

High Issues

#1 | The admin can lock trading and liquidation activity.

File	Severity	Location	Status
Timelock.sol	High	L56-58	Open

Description – The contract contains functionality in which the contract's admin can cease trading and liquidation activity in the vault contract for an indefinite period of time. This is not recommended as it can lock the contract permanently, and there is no time limit in which the trading and liquidity activity will auto-enable. There must be functionality in which the trading and liquidation activity cannot be locked up indefinitely.

#2 | The Manager can lock the selling of USDL.

File	Severity	Location	Status
Vault.sol	High	L459-483	Open

Description – The manager of the contract can lock the selling of USDL tokens for an indefinite period of time which is not recommended as this can cause the lock of funds. There must be a check so that the tokens should not get locked permanently in the contract.

Medium Issue

#1 | The owner can set fees of more than 25%.

File	Severity	Location	Status
BaseOrderManager.sol	Medium	L56-58	Open
RewardRoter.sol	Medium	L70-72	Open
Vault.sol	Medium	L296-314	Open

Description – The contract contains functionality in which the governor can update the fees to more than 100% which is not recommended as there should not be any fees in the contract that can be more than 25% in the contract.

#2 | Missing 'isContract' check.

File	Severity	Location	Status
BaseOrderManager.sol	Medium	L44-54	Open
LLPManger.sol	Medium	L72-74, L97-107	Open
PriceFeed.sol	Medium	L54-60	Open
PythPriceFeed.sol	Medium	L59-65, L75-77	Open
ReaderCache.sol	Medium	L27-45	Open
ReaderContract.sol	Medium	L75-85	Open
RewardRouter.sol	Medium	L58-68	Open
USDL.sol	Medium	L20-22	Open
Utils.sol	Medium	L94-104	Open
Vault.sol	Medium	L188-191, L208-216, L260-263	Open

Description – The contract contains the functionality in which the owner can set any arbitrary address in the place of the contract address, as this can lock the functionalities if the owner has set any arbitrary address in the place of the contract address.

Remediation – Add a 'require' check that the address can only be set to the contract address.

#3 | Missing Threshold.

File	Severity	Location	Status
LLPManger.sol	Medium	L88-90	Open
OrderManager.sol	Medium	L226-229, L236-238, L240-254, L260-268	Open
PriceFeed.sol	Medium	L70-80	Open
ReaderContract.sol	Medium	L35-73	Open
RewardRouter.sol	Medium	L66-72	Open
RewardTracker.sol	Medium	L73-75, L81-83	Open
Utils.sol	Medium	L58-92, L110-120	Open

Description – The governor can update any arbitrary amount in the value, including zero, as this can fail the functionalities in the contract if the value is set to zero. There must be a check so that the value in the contract cannot be set to zero to avoid these circumstances in the contract.

Low Issue

#1 | Missing events

File	Severity	Location	Status
BaseOrderManager.sol	Low	L44-46, L48-50, L52-54, L56-58	Open
LLPManager.sol	Low	L72-86, L88-90, L92-95, L97-107	Open
OrderManager.sol	Low	L226-229, L236-238, L240-254	Open
Pricefeed.sol	Low	L59-65, L75-77, L62-68, L70-80	Open
PythPriceFeed	Low	L59-65, L75-77, L79-85	Open
ReaderCache.sol	Low	L27-45	Open
ReaderContract.sol	Low	L35-73, L75-85	Open
RewardRouter.sol	Low	L58-68	Open
RewardTracker.sol	Low	L73-75, L81-83	Open
Timelock.sol	Low	L84-151, L175-194	Open
Vault.sol	Low	--	Open

Utils.sol	Low	--	Open
YieldToken.sol	Low	L48-63, L70-88	Open
BaseToken.sol	Low	L47-62, L69-75	Open

Description – It is recommended that all the critical parameter changes in the contract be emitted.

#2 | Floating pragma solidity version.

File	Severity	Location	Status
BaseOrderManager.sol	Low	L3	Open

Description – Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

#3 | Missing visibility

File	Severity	Location	Status
OrderManager.sol	Low	L26	Open
RewardTracker.sol	Low	L40	Open

Description – It is recommended to add 'public', 'private', or 'internal' visibility while declaring or initializing a state variable or a mapping in the contract.

#4 | Missing zero or dead address check.

File	Severity	Location	Status
PriceFeed.sol	Low	L62-68	Open
RewardRouter.sol	Low	L74-76	Open
RewardTracker.sol	Low	L69-71, L77-79	Open
TierBasedTradingFees.sol	Low	L17-19, L21-23	Open
Timelock.sol	Low	L84-99, L210-212	Open

Vault.sol	Low	L193-196	Open
YieldToken.sol	Low	L48-50, L57-63	Open
BaseToken.sol	Low	L47-62, L73-75	Open
MintableBaseToken.sol	Low	L20-22	Open

Description – It is recommended to check that the address cannot be set to zero or dead address.

Informational Issue

#1 | NatSpec Documentation missing.

File	Severity	Location	Status
All	Informational	--	Open

Description – If you started to comment on your code, also comment on all other functions, variables, etc.

#2 | Contract doesn't import packages from source (like OpenZeppelin etc.)

File	Severity	Location	Status
All	Informational	--	Open

Description – We recommend to import all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY