SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

MUX Degen Protocol

AUDIT

SECURITY ASSESSMENT

25. March, 2024

FOR

SolidProof_io @solidproof_io

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	MUX Degen Protocol
Website	https://mux.network/#/
About the project	MUX Protocol offers the first decentralized perpetual trading aggregator; it provides traders with deep aggregated liquidity, optimized trading cost, up to 100x leverage, diverse market options and unique aggregator features like smart position routing, aggregated position, leverage boosting and liquidation price optimization.
Chain	Arbitrum
Language	Solidity
Codebase Link	Provided as Files
Commit	N/A
Unit Tests	Provided

Social Medias

Telegram	https://t.me/muxprotocol
Twitter	https://twitter.com/muxprotocol
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	https://medium.com/@muxprotocol
Discord	https://discord.gg/bd88NrzN3N
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	25. March 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/Types.sol	059e046dea4451f1ba831fb7686ff56e84a1b9b3
contracts/orderbook/Admin.sol	ec60e01ed39e05c96a2f1dc76414b332e3215545
contracts/orderbook/Types.sol	02d868cca7d174fdab42685dcab1bbe565e7f378
contracts/orderbook/OrderBook.sol	c503ed78b03e76be9d9829b3ea27549affb6d741
contracts/orderbook/Storage.sol	fa509fefc3e45f1076a8616ea54565f458562b68
contracts/orderbook/Getter.sol	f5d0cf053bf53df86602ed007e23b32a913294b3
contracts/peripherals/DegenPOL.sol	801bc0fa769b4af3decc0c38dd9e45e240db1c87
contracts/peripherals/ReferralTiers.sol	bd70725c81dc9253b10fbfba2d3de508b1c647e3
contracts/peripherals/MlpToken.sol	16bd3cc58eaed4d3edd5a3114dde3d9c214d946c
contracts/peripherals/DegenFeeDistributor.sol	4640c004a4c8e5f491e8cac255747c142b9a30ba
contracts/DegenPoolStorage.sol	f924ea62211283037b4cd9083ce9ba3f6f21d366
contracts/libraries/LibMath.sol	b769eee7daf17c77a695db023f42fd37cbaafa97
contracts/libraries/LibAccount.sol	84594f882099a62c04ae5e334f0e1f8c42821b6f
contracts/libraries/LibConfigKeys.sol	5fde62801552ef56c7fe1832b96be14dfccf878a
contracts/libraries/LibAsset.sol	20bedf200297a33d0f0f019ff0fdc0c6344f7f7e
contracts/libraries/LibPoolStorage.sol	3c5a5778d431ef476dbd21fd5ed6da7ec369ec96
contracts/libraries/LibOrder.sol	0680e2cc84dc9a9acd8bc1a0f3a65bba1a1bcc5a
contracts/libraries/LibTypeCast.sol	673337123a154250f319ca5925d401355dced34b
contracts/libraries/LibReferenceOracle.sol	1b76417262444e73936af38600bb21e766d835b3
contracts/libraries/LibSubAccount.sol	43d4a8f67913f0a6e689592c56c4e913a22732ca
contracts/libraries/LibOrderBook.sol	8610d9243378a3254c8e583105e7afe1998d804a
contracts/facets/Admin.sol	0788c73d0ac73ac660c153765e0586b29e86ada7
contracts/facets/Trade.sol	9e1d9eb7fc7587cddc3c4a992b44457bc504c593
contracts/facets/Getter.sol	40a3d68f416bdee8cf9071ef1d137223d143daf0
contracts/facets/Account.sol	9ab5d78bb57a86dbe540d80372eebb5eebd839f6
contracts/facets/Liquidity.sol	a92b8918219f470cfae65cc1bc4073bb53874404

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 (direct imports).

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	6
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	11
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	9
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	8

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

Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit 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 executing the 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 and smart 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 an upgradeable	❌ **Deployer can update the contract with new functionalities**
Description	The deployer can replace the old contract with a new one with new features. Be aware of this, because the owner can add new features that may have a negative impact on your investments.
Comment	N/A

Ownership

The ownership is not renounced	❌ **The owner is not renounce**
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 - If the contract is not deployed then we would consider the ownership to be not renounced. Moreover, if there are no ownership functionalities then the ownership is automatically considered renounced.

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 cannot mint new tokens	✅ **The owner cannot mint new tokens**
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	N/A

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 cannot burn tokens	✅ **The owner cannot burn tokens**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

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 addresses**
Description	The owner is not able blacklist addresses to lock funds.
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 in developing and maintaining the contract.

Contract owner cannot set fees more than 25%	✅ **The owner cannot levy unfair taxes**
Description	The owner is not able to set the fees above 25%
Comment	N/A

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 tokens 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.

Owner cannot lock the contract	✅ **The owner cannot lock the contract**
Description	The owner is not able to lock the contract by any functions or updating any variables.
Comment	N/A

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

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
14	10	3	0

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
90	0

External	Internal	Private	Pure	View
83	215	6	40	103

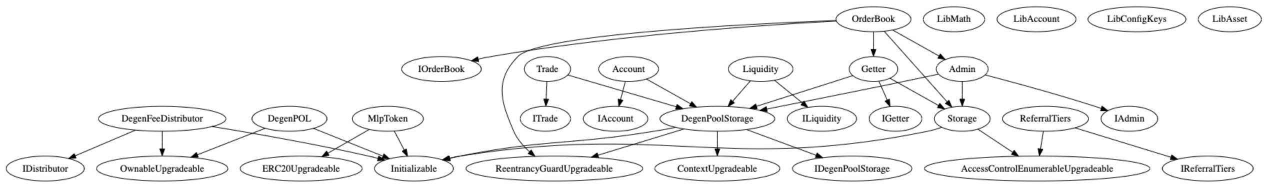
StateVariables

Total	🌐Public
65	14

`0.8.19`				

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 methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



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 a single entity controls the contract or if certain participants have special permissions or abilities that others do not.

In the project, some authorities have access to the following functions:

File	Privileges
facets/Admin	• Set Maintenance Parameters • Add Asset and its parameters • Set Pool Parameters • Set Asset Flags
facets/liquidity	• The orderbook address can add/remove liquidity
facets/Trade	• The orderbook address has access to all the functions in the cotnract
Orderbook/OrderBook	• The broker role address can fill and cancel orders at any time
DegenFeeDistributor	• Set Maintainer Address • Withdraw any token from the contract's balance

Recommendations

To avoid potential hacking risks, the client should carefully manage the private key of the privileged account. 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 Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

| Missing balance check

File	Severity	Location	Status
DegenFeeDistributor	Low	L143	Open

Description - Make sure to validate that the contract has enough balance to withdraw.

Informational issues

#1 | NatSpec documentation missing

File	Severity	Location	Status
All	Informational	N/A	Open

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

#2 | Disable initializing

File	Severity	Location	Status
All	Informational	N/A	Open

Description - If the owner updates the contract, a disableInitializer call in the constructor must be implemented. This prevents calling the initialize function again to set the state variables in the contract. This should be implemented only if the contract was deployed before. Otherwise, the owner cannot call the initialize function to set the variables.

Recommendation - If the contract hasn't been deployed, remove the disableInitializer in the constructor. Otherwise, you are not able to initialize the contract. When the contract has a deployed version already, leave it as it is.

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

File	Severity	Location	Status
All	Informational	N/A	Open

Description - We recommend importing 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.

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY