



SOLIDProof

Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

HorizonDEX

AUDIT

SECURITY ASSESSMENT

07. July, 2023

FOR



SolidProof_io



@solidproof_io

Introduction	3
Disclaimer	3
Project Overview	4
Summary	4
Social Medias	4
Audit Summary	5
File Overview	6
Externally Imported packages	7
Audit Information	9
Vulnerability & Risk Level	9
Auditing Strategy and Techniques Applied	10
Methodology	10
Overall Security	11
Medium or higher issues	11
Upgradeability	12
Ownership	13
Ownership Privileges	14
Minting tokens	14
Burning tokens	15
Blacklist addresses	16
Fees and Tax	17
Lock User Funds	18
Components	19
Exposed Functions	19
Capabilities	20
Inheritance Graph	21
Centralization Privileges	22
Audit Results	23



Introduction

[SolidProof.io](https://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](https://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	HorizonDEX
Website	https://horizondex.io
About the project	Welcome to HorizonDEX! We're introducing a groundbreaking approach to trading on Linea. HorizonDEX is a Concantrated Liquidity DEX which let users allocate liquidiity within a custom price range, enabling traders to maximize their efficiency and minimize slippage.
Chain	Linea
Language	Solidity
Codebase Link	https://github.com/Horizon-Dex/contracts
Commit	888c8d2
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/horizondex_io
Twitter	https://twitter.com/horizondex_io
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	https://medium.com/@HorizonDEX_io
Discord	https://discord.com/invite/horizondex
Youtube	N/A
TikTok	N/A
LinkedIn	N/A



Audit Summary

Version	Delivery Date	Changelog
v1.0	06. July 2023	<ul style="list-style-type: none"> • Layout Project • Automated- /Manual-Security Testing • Summary

Note - This Audit report consists of a HorizonDEX smart contract security analysis. This analysis did not include functional testing (or unit testing) of the contract's logic.





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/periphery/base/ LiquidityHelper.sol	ab45b966008075f2b90471aff092651 4849e045e
contracts/periphery/base/ DeadlineValidation.sol	a53a72490545ffd31105c0cbbd12baf 0ca94e8c9
contracts/periphery/base/Multicall.sol	8021a8e6b73497b46905a66be1f9b6 f2a6753e84
contracts/periphery/base/ ERC721Permit.sol	c05a9788046ebbee0fddbc1953d6c5 ce606e15f4
contracts/periphery/base/ ImmutablePeripheryStorage.sol	f0278fea524b5127c92209e951fc180 8cf7d22ca
contracts/periphery/base/ RouterTokenHelperWithFee.sol	ef2a33bb0cad3cc222a89daf69e31cd 66c361e45
contracts/periphery/base/ RouterTokenHelper.sol	e8bf5b5767b2864b2d948ea2fb8623f b41aa6b58
contracts/periphery/TicksFeesReader.sol	139f80c2fa62d6db8ce8cb5c999962c 6e8cb1bbe
contracts/periphery/QuoterV2.sol	c80ec60a494a73f35bfcd27798b1e28 ea8d50630
contracts/periphery/ TokenPositionDescriptor.sol	7929947c3ec71e2fe050a421062e2c 3f96dbeb51
contracts/periphery/Router.sol	b96eb142d8c9e1074fa2f6edad2515 0c0b9fe17d
contracts/periphery/ BasePositionManager.sol	030dc19d161f426e11cf4bf0c68ce32 b4895f199
contracts/periphery/libraries/BytesLib.sol	2d48ce28b44c941d3c327b21addf33 45268d7b63
contracts/periphery/libraries/ AntiSnipAttack.sol	3de760084884c1bc69cc3109cc821a c355d49239
contracts/periphery/libraries/ PoolTicksCounter.sol	7e18f3f55863c33181377a49a7ea9fa 0563e4f93



contracts/periphery/libraries/ PoolAddress.sol	4bad7ccc7c5c466f6ee195dea6bab0 abec38336b
contracts/periphery/libraries/PathHelper.sol	ce022056c3f9948356557c6df4aff032 b4149401
contracts/periphery/libraries/ TokenHelper.sol	28ddaab351f4c4e51543eedde1626f acfc32a480
contracts/periphery/libraries/ LiquidityMath.sol	c9ba29a9caf65547e4221be7810bdc 0bb426655d
contracts/periphery/ AntiSnipAttackPositionManager.sol	9cd58f3aa939403bbc0f2209458343 0e966943d3
contracts/PoolTicksState.sol	d796fb7d4e08dc974cfc71280e02a24 c324eb6ed
contracts/PoolStorage.sol	af59d9e5a92446f1a6f32f3994f64132 7c04fd02
contracts/oracle/PoolOracle.sol	6ba86fdd39699311706a1e252778c3 1484e12ad8
contracts/Pool.sol	59026ed45adedec5b92a2fcb83819b 6ab09fd066
contracts/Factory.sol	c430cd381506d5909c8ac9e3909a9d 9ccdf321f3

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.

Externally Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	2
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/ IERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/utils/ SafeERC20Upgradeable.sol	1



@openzeppelin/contracts/proxy/Clones.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	8
@openzeppelin/contracts/token/ERC20/Utils/SafeERC20.sol	2
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ ERC721Enumerable.sol	1
@openzeppelin/contracts/Utils/Address.sol	1
@openzeppelin/contracts/Utils/Strings.sol	1
@openzeppelin/contracts/Utils/math/Math.sol	1
@openzeppelin/contracts/Utils/structs/EnumerableSet.sol	1

Note for Investors - This is a general 1:1 fork from Kyber Swap. Thus, all the contracts in the HorizonDEX codebase are identical to KyberSwap's contracts in terms of logic

- Contracts inside are the same as the pancake-smart-contracts directory
 - <https://github.com/KyberNetwork/ks-elastic-sc/tree/main/contracts>
 - Differences between HorizonDEX and KyberSwap contracts are the following:
 - Only the pragma version is changed, and minor formatting has been done

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 as possible.
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

Medium or higher issues

No critical Issues found

✓ Contract is safe to deploy

Description	The contract does not contain issues of high or medium criticality. This means that no known vulnerabilities were found in the source code.
Comment	N/A

Note - This project consists of the following forks

- **KyberSwap**

- Unit tests with at least 95% code coverage and a Whitepaper were not provided to SolidProof, so we cannot ensure the complete functional correctness of the code's logic.
- We recommend **HorizonDEX** team conduct unit and fuzz tests thoroughly to rule out the possibilities of unwanted logical and calculation errors.
- Read the whole report and modifiers section for more information
- The low issues that remain unfixed in the **Kyber Swap** codebase still exist in the forked code.
- We recommend using a multi-sig wallet for the owner's address to prevent any risk of the loss of a private key.
- Do your own research here before investing.



Upgradeability

Contract is not an upgradeable



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

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 variables that 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
10	7	1	8


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
80	18



External	Internal	Private	Pure	View
61	120	11	17	42

StateVariables

Total	 Public
52	27



Capabilities

Solidity Versions observed	 Transfers ETH	 Can Receive Funds	 Uses Assembl y	 Has Destroyable Contracts
0.8.9	YES	YES	YES	— — — —



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 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
All	❖ Privileges are also identical to the KyberSwap contracts.

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 Results

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

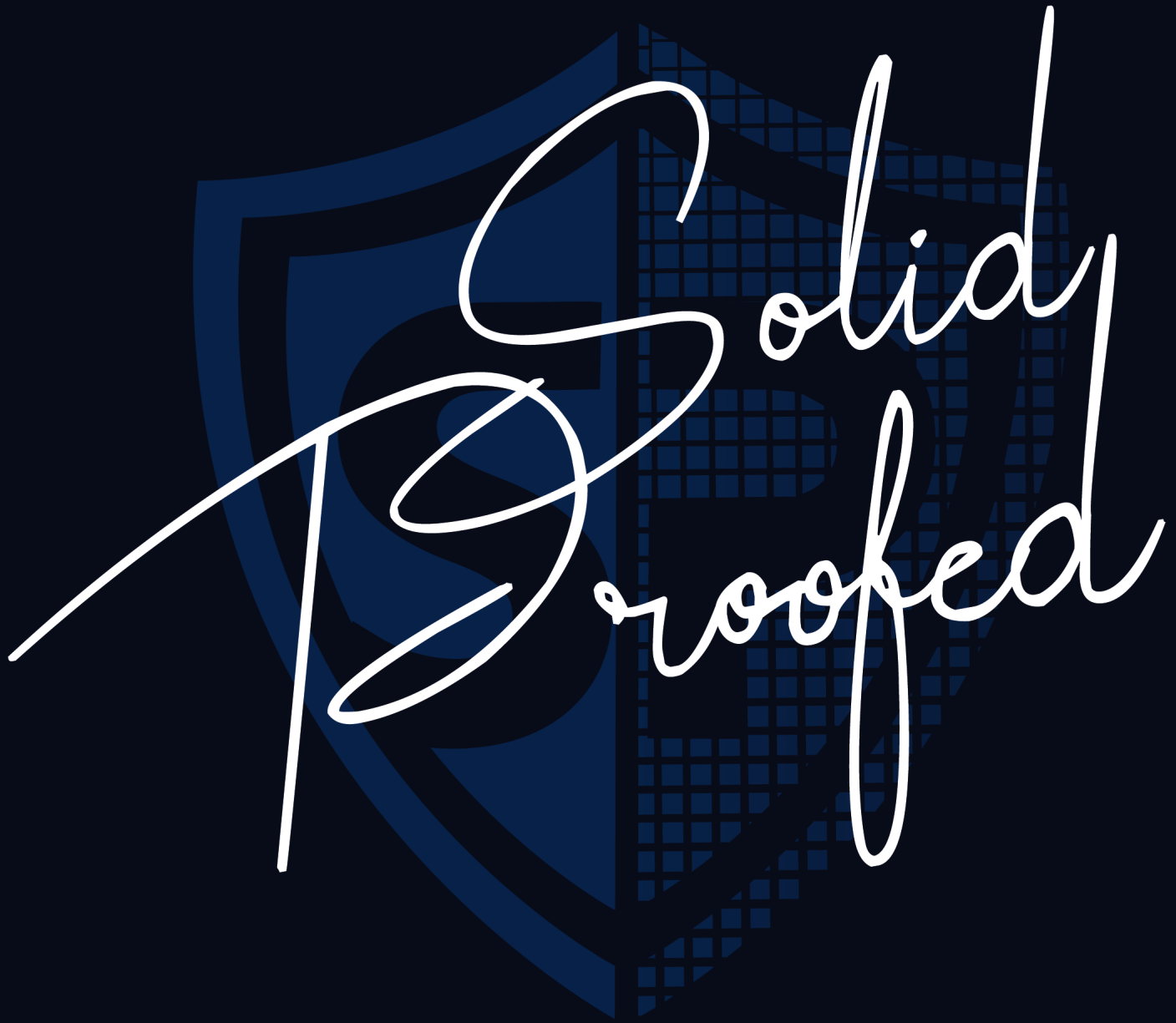
File	Severity	Location	Status
All	Informational	N/A	ACK

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.



**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY