



SOLIDProof

Bring trust into your projects

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

MADE IN GERMANY

Pound Swap

AUDIT

SECURITY ASSESSMENT

05. August, 2023

FOR



**POUND
SWAP**



SolidProof_io



@solidproof_io

Introduction	3
Disclaimer	3
Project Overview	4
Summary	4
Social Medias	4
Audit Summary	5
File Overview	6
Imported packages	8
Components	10
Exposed Functions	10
Capabilities	11
Inheritance Graph	12
Audit Information	13
Vulnerability & Risk Level	13
Auditing Strategy and Techniques Applied	14
Methodology	14
Overall Security	15
Upgradeability	15
Ownership	16
Ownership Privileges	17
Minting tokens	17
Burning tokens	18
Blacklist addresses	19
Fees and Tax	20
Lock User Funds	21
Centralization Privileges	22
Audit Results	23

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	Pound Swap
Website	https://www.poundswap.com/
About the project	Pound Swap is a community-driven, ve(3,3) Liquidity Book Decentralized Exchange (DEX), providing a powerful platform for DeFi liquidity and token exchange. As a fork of Traderjoe Liquidity Book, Pound Swap is similar in design and functionality to Traderjoe LB, offering users a zero-slippage, capital-efficient AMM solution.
Chain	Linea
Language	Solidity
Codebase Link	https://github.com/PoundSwap/poundswap-core/tree/main/src
Commit	071db04
Unit Tests	Provided

Social Medias

Telegram	N/A
Twitter	https://twitter.com/poundswap?s=21
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
Youtube	N/A
TikTok	N/A
LinkedIn	N/A



Audit Summary

Version	Delivery Date	Changelog
v1.0	05. August 2023	<ul style="list-style-type: none"> • 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. 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 it was not functionally tested by us.



File Overview

The project provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an S H A - 1 H a s h .

File Name	SHA-1 Hash
src/interfaces/IJoePair.sol	d6b66fa7e7621cc4d666c2bbf54476a912848ff4
src/interfaces/ILBLegacyFactory.sol	65834e8b6a47176de0bbe0a50c5e4f53aff0c6bc
src/interfaces/IJoeRouter01.sol	479b3966c78f68322d6ba98b6fbdb3b8d8578f24
src/interfaces/IJoeRouter02.sol	538a3e5c2ef7c4dce52c5f34a8a09d91dcb cfa3e
src/interfaces/ILBToken.sol	42fb778665c05ca7299e67db5bbfe9c7ecc eef74
src/interfaces/ILBPair.sol	90bcc58d024b3e71968f330ec904a6c2f77 b27cc
src/interfaces/ ILBFlashLoanCallback.sol	be2ddfa083e5f29c26ff40da6468cf650e59 82c3
src/interfaces/ILBLegacyPair.sol	827c1ab4dc156248060b2c30692b2154d9 6a17ae
src/interfaces/ILBLegacyToken.sol	7c981e1db0c50409f17f3189bddd494be47 ea79d
src/interfaces/ILBRouter.sol	b6ff28995eb748daaf089524657c93f3f582 a8cb
src/interfaces/ILBLegacyRouter.sol	80b731d96d1a8ff343118e4c08d3254b8fcf 7caf
src/interfaces/IJoeFactory.sol	ba919e62c8c95062b1fec5ff8988de77f328 c029
src/interfaces/ILBFactory.sol	1a1819ecd419842ebd4c9a21854abe76bb e6c7fe
src/interfaces/IWNATIVE.sol	0aed12b73cd20b230742d12d42495075cd 9ac8cc
src/interfaces/IPendingOwnable.sol	9f11672445ba7d6ad72ea9ebad506c41a7 372da4



src/LBQuoter.sol	1b30618882a7195988ac47a0c763299c6fc02759
src/LBToken.sol	e9e2ed4416de6521b00f66868be2e091855f93fd
src/LBRouter.sol	cbb9eb712382bfb73ba498d0fd40e6a503aafaf3
src/LBPair.sol	df0f9a0e19e6401da437509329ba56be1d61c5aa
src/LBFactory.sol	38e3d78efa0516abf8c01d892165122376e8b680
src/libraries/ImmutableClone.sol	21a7b787a496db383a4c6afda2f9f07866d228cf
src/libraries/PriceHelper.sol	edaeadd93aae5eb7fd88079d7e6f61013742d147
src/libraries/Constants.sol	b53687ff8ff5c3769b9c68de33a625ff1dcf1cae
src/libraries/JoeLibrary.sol	ab35746901c1f91ecaa22a0149c7f408f9190135
src/libraries/FeeHelper.sol	42af45602459e8e83bf57499f1fa98dd46b184a3
src/libraries/BinHelper.sol	6c29cfe6a8f0dd3b801b59d78a716c1356ba94f2
src/libraries/math/SafeCast.sol	9f07c9cb1ec097d5da73a4483672abb21973b365
src/libraries/math/Uint128x128Math.sol	1bcf36cb1308b8802c0f5b0bddf068c1316f67e5
src/libraries/math/TreeMath.sol	955229c0406311ad4f9267472c31b613ebfceb6a
src/libraries/math/LiquidityConfigurations.sol	9533c10f5e11369100685dac0220164b30bd04bb
src/libraries/math/BitMath.sol	90d713e3474d4dba6d6f171e6def882171486ac6
src/libraries/math/SampleMath.sol	cb7e78e54200c8721c2c4f981ea06356a58a4198
src/libraries/math/Uint256x256Math.sol	290c4fc902f562cbd6f5aeeccf9f12da3cf5d4230



src/libraries/math/PackedUint128Math.sol	6f02a087cf139057155de2709db213ffee4d2207
src/libraries/math/Encoded.sol	6a11acea42abee14cda9d15c450885d20c3cf284
src/libraries/Clone.sol	bd9722b474583640c05b72956f5a9460430513bc
src/libraries/PendingOwnable.sol	92bd902e856d3c1a5b96855168ffa85809ae0361
src/libraries/OracleHelper.sol	90fea94a60229bfc5b015a20ab55096a52346e64
src/libraries/AddressHelper.sol	84547459e0247aec06b11ef5eecfb49692d4a350
src/libraries/TokenHelper.sol	018a0453517b9586e837c6f6812c7e13c525264e
src/libraries/ReentrancyGuard.sol	8b19a2c26b75cf13649a6434eb1b79c6af3d5f35
src/libraries/PairParameterHelper.sol	378201a8fd009c768206ff1d607403e6ea827f54

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/token/ERC20/IERC20.sol	14
openzeppelin/utils/introspection/IERC165.sol	1
openzeppelin/utils/structs/EnumerableMap.sol	1
openzeppelin/utils/structs/EnumerableSet.sol	1



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.

Moreover, The code is a 1:1 fork from trader Joe. The forked repo can be found on the following link - <https://github.com/traderjoe-xyz/joe-v2/tree/main/src>



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
5	19	15	3


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
338	18





External	Internal	Private	Pure	View
316	369	20	175	175

StateVariables

Total	 Public
87	0



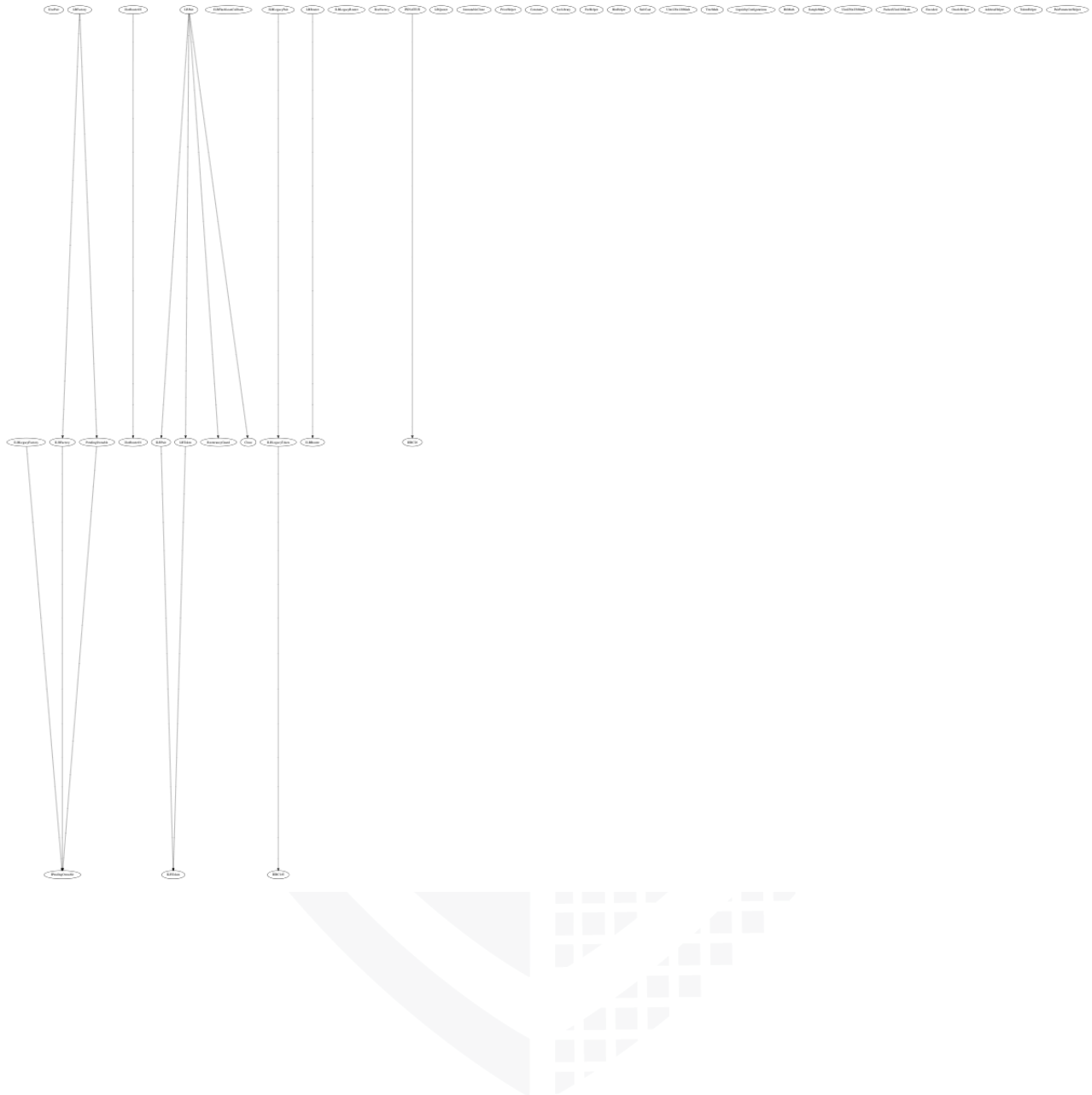
Capabilities

Solidity Versions observed	 Transfers ETH	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
^0.8.10	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 methods from 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 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

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

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
1. LBFactory.sol	<ul style="list-style-type: none"> ❖ onlyOwner <ul style="list-style-type: none"> - Set LB Pair Implementation and Ignored addresses - Set Preset parameters of a bin step - Set preset status - Remove Preset - Set Fee Parameter of a LBPair - Set Fee Recipient Address - Set Flash Loan Fee - Add/Remove Quote Asset - Manually Force the decay of the volatility reference variables
2. LBRouter.sol	<ul style="list-style-type: none"> ❖ onlyFactoryOwner <ul style="list-style-type: none"> - Withdraw stuck tokens from the contract including the LB Tokens

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 | Floating Pragma

File	Severity	Location	Status
All	Informational	L2	Open

Description - The current pragma Solidity directive is “^0.8.10”. Contracts should be deployed with the same compiler version and flag that they have been tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using other versions

#2 | 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.



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

MADE IN GERMANY