



SOLIDProof

Bring trust into your projects

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

MADE IN GERMANY

LendLand

AUDIT

SECURITY ASSESSMENT

06. February, 2025



[SolidProof.io](https://solidproof.io)



[@solidproof_io](https://t.me/solidproof_io)

Introduction	4
Disclaimer	4
Project Overview	5
Summary	5
Social Medias	5
Audit Summary	6
File Overview	7
Imported packages	10
Audit Information	11
Vulnerability & Risk Level	11
Auditing Strategy and Techniques Applied	12
Methodology	12
Overall Security	13
Upgradeability	13
Ownership	14
Ownership Privileges	15
Minting tokens	15
Burning tokens	16
Blacklist addresses	17
Fees and Tax	18
Lock User Funds	19
Components	20
Exposed Functions	20
StateVariables	20
Capabilities	21
Inheritance Graph	22
Centralization Privileges	23
Audit Results	25
Critical issues	25
High issues	25



Medium issues	25
Low issues	25
Informational issues	26





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	LendLand
Website	https://lendland.io/
About the project	LendLand is a cutting-edge blockchain platform dedicated to revolutionizing the lending and borrowing experience within the Boba Network. Our mission is to provide users with intuitive tools and secure measures, making LendLand the preferred choice for DeFi enthusiasts.
Chain	BOBA Network
Language	Solidity
Codebase Link	Provided as a Private Repo
Forked Status	The contracts are 1:1 forked from Venus Protocol https://github.com/VenusProtocol/isolated-pools
Unit Tests	Provided

Social Medias

Telegram	https://t.me/Lendland_Portal
Twitter (X)	https://twitter.com/socialLendland
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	https://medium.com/@lendland
Discord	https://discord.gg/3SgKDzzyje
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	25. March 2024	<ul style="list-style-type: none"> Layout Project Automated- /Manual-Security Testing Summary
v1.2	06. February 2025	<ul style="list-style-type: none"> Reaudit

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/IPancakeswapV2Router.sol	bf8733c5c5ef44c0fab16a60432a0bcab02cd73b
contracts/ExponentialNoError.sol	f8124ef60f121ea8950463a3ddcae6c8c6981856
contracts/BaseKinkedRateModelV2.sol	127496e3d2441a7a0114b697c11585a1b036d98f
contracts/LendLandToken.sol	df57de0a8de60cfc1d14ad4650ed04caaf1d408d
contracts/LinearInterestRateModel.sol	85a8fc12757b5e073d062275456f0f6c92d3f840
contracts/Rewards/RewardsDistributor.sol	8e169a992ad42e5e265f54b138c1b7713f65f107
contracts/RiskFund/RiskFund.sol	588d39cc32c811a3c14387e76dd57b7e1fbee58
contracts/RiskFund/ReserveHelpers.sol	fb54baf36ea7429b25604ad809bcbf1da7861c00
contracts/RiskFund/IRiskFund.sol	fc859410835d7f9bc7b2dcc7c3ad08dbc858cb3c
contracts/MaxLoopsLimitHelper.sol	a131f4a9f29d397409f849d1830847fa2ca07020
contracts/ResilientOracle.sol	33a5ae9fda831cf0788076e8c68ae76cf30275c3
contracts/LeToken.sol	204065a10f52bc5fe7b9aee3fed936c4e6e2fad0
contracts/Pool/PoolRegistryInterface.sol	1096e46cf3186974d649984ac9c1699adf07ae58
contracts/Pool/PoolRegistry.sol	0e427c71a501a1693f6c1ed1173c1e921f4d637d
contracts/KinkedRateModelV2.sol	8b88293ea9b17ce5c3a909e925958b4ced822d81
contracts/ComptrollerStorage.sol	5b93c108ef91d2e2b10b6e5932fdf73201d12219
contracts/Shortfall/Shortfall.sol	71c7f189e60f33e1756257b02761ff0ee4636312
contracts/Gateway/INativeTokenGateway.sol	547cdc58ff6fad92572b4e7eb55113c2f47e3c5f
contracts/Lens/PoolLens.sol	857f02c46e8c9f23b3714631e43adf2acf782762
contracts/Gateway/NativeTokenGateway.sol	fb7b76b6859b9abd429ba740750c354fd56e992d
contracts/lib/TokenDebtTracker.sol	1080cb464181d3dcd68122d695929697fada6e00
contracts/lib/validators.sol	7cb7492cde6018892a18805fcceffefcc8d88c05
contracts/lib/constants.sol	e7def897bcfc8a7900975cbe8a95b0631db20c44



contracts/lib/ApproveOrRevert.sol	c0ba6a24ed591c4bf3719bedfbd8f1fafb06236f
contracts/lib/imports.sol	97cb19a84500bba7848e44fc13bbe8f8a260f7c7
contracts/InterestRateModel.sol	b81390ceddce17ae278fb309b815c511417a1b3c
contracts/ErrorReporter.sol	03cf892635603d0a25b69feaa766fa599057ea81
contracts/Comptroller.sol	b1cdd061a25b8965807d62090a6017e2b89053b9
contracts/oracles/PythOracle.sol	40eab865fdd7aea0b34b62998e803946e6355324
contracts/oracles/SequencerChainlinkOracle.sol	0946e9bf212a9a635142e32ec621c01c1b7edf2f
contracts/oracles/BoundValidator.sol	d7783417092fbdf8894417691bbe1b5928481992
contracts/LeTokenInterfaces.sol	389d59cab1cd4c417c1d8551abdbcc2db50ba6ea
contracts/ComptrollerInterface.sol	e940510e0ba2ec69d3e9de3887d06c5549e5265b
contracts/oracles/BinanceOracle.sol	b4aaa8706fa26f374865cf47ce95d34302ee150c
contracts/oracles/ChainlinkOracle.sol	2a717f6e49943a5b0cd6fdc6dc2a3e924d05609b
contracts/oracles/WstETHOracle.sol	fdc7610729129c96d5f0b7b2d9ac03f439a4f1a5
contracts/swapRouter/uniswap/ UniswapV2Router02.sol	3a87cc4d40052750a294205a5fd83381d0e6c568
contracts/swapRouter/uniswap/UniswapV2Pair.sol	bfebcfb0b95c89aefdc5f74d82c08a9a1e6bcf60
contracts/swapRouter/uniswap/weth.sol	6dad8697868b9a8f3255406013d4c3bdbe1a04aa
contracts/swapRouter/uniswap/ UniswapV2ERC20.sol	1488ce009678b617eb78115885b4d1f5e99f7003
contracts/swapRouter/uniswap/ UniswapV2Factory.sol	4fc657d98d75fdefccfc6ecb537b6ffb16977a2a
contracts/swapRouter/interfaces/ IUniswapV2Callee.sol	3d9551ff64125a537d1a3a96f9f1a9de32f48536
contracts/swapRouter/libraries/UQ112x112.sol	14f29ae61607aef5b3583cda7180056cbd55eb7f
contracts/swapRouter/libraries/SafeMath.sol	8ec8f7024f040294fe26bca4abcbca18b78f222b
contracts/swapRouter/libraries/UniswapV2Library.sol	0fa2c05304d2e3b49b0f9955141c20e5101219dc
contracts/swapRouter/interfaces/IUniswapV2Pair.sol	7d13cddd06b7998fd574689599d697684e9c62f0
contracts/swapRouter/interfaces/ IUniswapV2Factory.sol	dedb60b101bdc60eb4f18833b56384c671827911



contracts/swapRouter/interfaces/IWETH.sol	b2a88e378e7e15b7e1a021ff8f81133f973f2452
contracts/swapRouter/interfaces/IUniswapV2Router02.sol	b1c85963ff9ceb715e9111403ff808c747579e25
contracts/swapRouter/interfaces/IUniswapV2ERC20.sol	05b9e9134b0bff6908732ff5acf574847146815e
contracts/swapRouter/interfaces/IUniswapV2Router01.sol	f05af3e2200912be81240d175acc872524597e72
contracts/swapRouter/interfaces/IERC20.sol	994f7c1e870f364bb3371a45b5a4bc49e6f758f0
contracts/swapRouter/libraries/Math.sol	d76d2450398af53d05e45c132d8b87de94c9ca86
contracts/swapRouter/libraries/TransferHelper.sol	8d4a62dc63b92072dc82bc34b555bdfd95ea7d62
contracts/oracles/PythOracle1.sol	68413c3f9d0a49a5d012ef575bf18ecff77aef8b

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
@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol	2
@openzeppelin/contracts-upgradeable/access/Ownable2StepUpgradeable.sol	6
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	9
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	7
@openzeppelin/contracts/access/Ownable2Step.sol	1
@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol	1
@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol	1
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	2
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	2
@openzeppelin/contracts/utils/math/SafeCast.sol	1
@openzeppelin/contracts/utils/math/SignedMath.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

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 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.

Example

We assume that you have funds in the contract and it has been audited by any security audit firm. Now the audit has passed. After that, the deployer can upgrade the contract to allow him to transfer the funds you purchased without any approval from you. This has the consequence that your funds can be taken by the creator.

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 - 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 can 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 contract's cost, 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 restricting access to certain tokens or assets for a specified period. 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

User funds cannot be locked but the owner has the ability to pause/unpause the auction

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
30	7	24	5


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
448	22




External	Internal	Private	Pure	View
388	499	11	74	163

StateVariables

Total	 Public
191	141



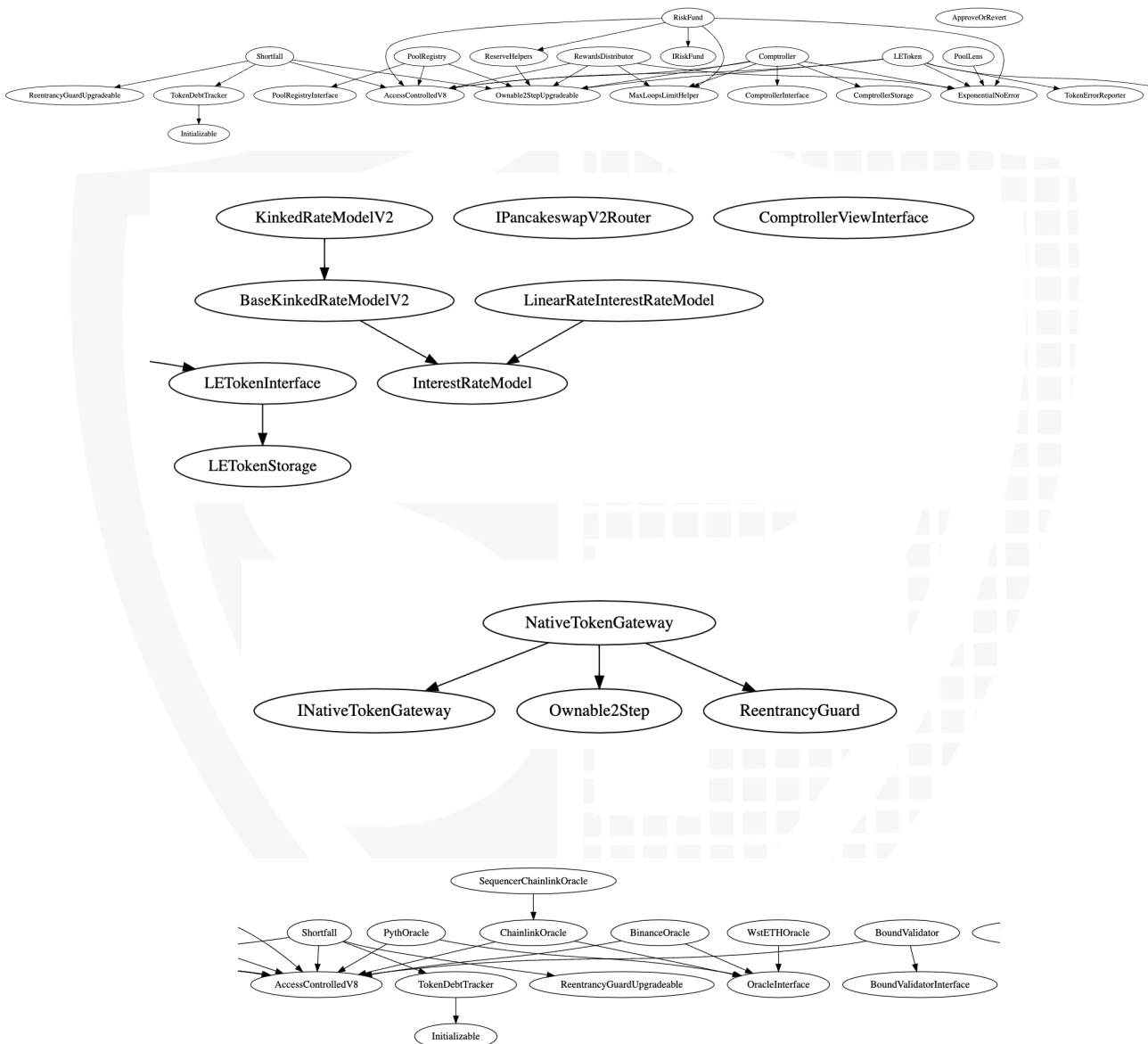
Capabilities

Solidity Versions observed	Transfers ETH	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
0.8.13 ^0.5.16 0.8.20 =0.6.12 =0.5.16 ^0.6.12 >=0.5.0 >=0.6.2 >=0.6.0	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.



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, some authorities have access to the following functions:

File	Privileges
RewardsDistributor	<ul style="list-style-type: none"> • Send Reward token to any recipient manually • Set Reward Token Speed • Set the loop limits
RiskFund	<ul style="list-style-type: none"> • Set the loop limits • Set Pool Registry and shortfall contract Address • Set Pancake Router address
Comptroller	<ul style="list-style-type: none"> • Set close factor, collateral factor, and liquidation incentive values • Set Market Borrow and Supply Caps • Pause/Unpause auctions • Set Collateral threshold for non-batch liquidations • Add rewards distributor and Price oracle address • Set Prime Token address
LETOKEN	<ul style="list-style-type: none"> • Set Protocol Share Reserve and shortfall address
PythOracle1.sol	<ul style="list-style-type: none"> • Any arbitrary address can set config.

Recommendations

To avoid potential hacking risks, the client should 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

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

#1 | Missing Events

File	Severity	Location	Status
PythOracle1.sol	Low	32-38	Open

Description - Make sure to emit events for all the critical parameter changes in the contract to ensure the transparency and trackability of all the state variable changes.

Informational issues

#1 | NatSpec documentation missing

File	Severity	Location	Status
All	Informational	N/A	Open

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



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