SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

ArbiDEX

Audit

Security Assessment
05. April, 2023

For

SolidProof_io

@solidproof_io

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 security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	31. March 2023 - 5. April 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Arbitrum

Website
https://www.arbidex.fi/

Telegram
https://t.me/Abridexchat

Twitter
https://twitter.com/arbidex_fi

Discord
https://discord.gg/arbitrumexchange

Description

Arbitrum Exchange is a cutting-edge decentralized exchange (DEX) that utilizes an automated market-maker (AMM) on the Arbitrum network. Setting itself apart from other DEXs, Arbitrum Exchange boasts the **lowest fees** for swapping crypto assets. Yield Farming rewards on the platform are among the most lucrative on the entire Arbitrum network, providing users with a profitable opportunity to earn passive income. One of the unique features offered by Arbitrum Exchange is the ability for users to stake $ARX and earn **100% of the protocol's generated revenue**. Unlike other exchanges, Arbitrum Exchange is free from arbitrage bots, ensuring users receive the best exchange rates possible.

Project Engagement

During the 31th of March 2023, **ArbiDEX Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Links

v1.0

ARX Token: 0xD5954c3084a1cCd70B4dA011E67760B8e78aeE84

Masterchef: 0xd2bcFd6b84E778D2DE5Bb6A167EcBBef5D053A06

Router: 0x3E48298A5Fe88E4d62985DFf65Dee39a25914975

Factory: 0x1C6E968f2E6c9DEC61DB874E28589fd5CE3E1f2c

ARXPool: 0xee1D57aCE6350D70E8161632769d29D34B2FbfC8

ArbiFlexPool: 0x489732e4D028e500C327F1424931d428Ba695dF6

SmartChefFactory: 0x086CdB9aA631270F4d14E9360735eeE86c6505e9

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. 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 evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

ARXPool

- Context
- Ownable
- IERC20
- Address
- SafeERC20
- Pausable
- IMasterChefV2

ARXToken

- SafeMath
- IBEP20
- Address
- SafeBEP20
- Context
- Ownable
- BEP20

Factory

- IArbDexFactory
- IArbDexPair
- IArbDexERC20
- SafeMath
- ArbDexERC20
- Math
- UQ112x112
- IERC20
- IArbDexCallee
- ArbDexPair

Router

- TransferHelper
- IArbiDexRouter01
- IArbiDexRouter02
- IArbiDexFactory
- SafeMath
- IArbiDexPair
- ArbiDexLibrary
- IERC20
- IWETH

MasterChefv2

- SafeMath
- IBEP20
- Address
- SafeBEP20
- Context
- Ownable
- BEP20
- ArxToken

ARXFlexiblePool

- Context
- Ownable
- IERC20
- Address
- SafeERC20
- Pausable
- IARXPool

SmartChefFactory

- IERC20
- IERC20Metadata
- Context
- Ownable
- ReentrancyGuard
- Address
- SafeERC20
- SmartChefInitializable

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

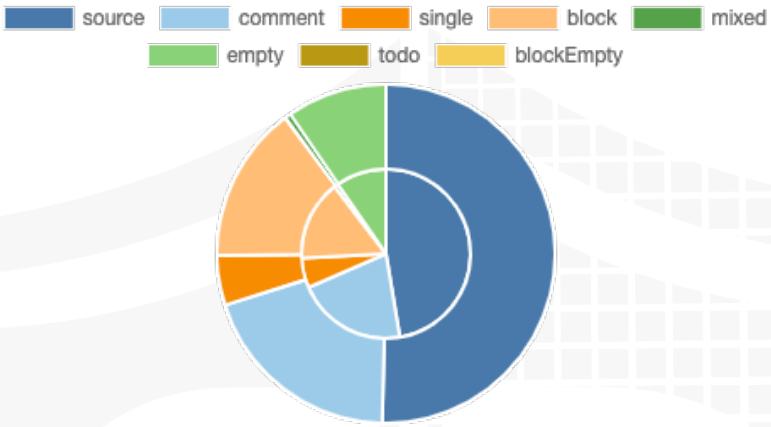
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

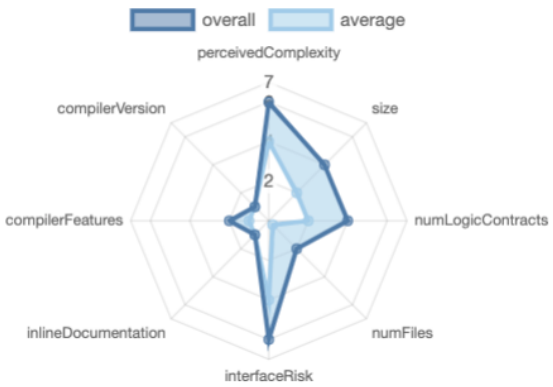
File Name	SHA-1 Hash
contracts/MasterChefv2.sol	0d7e1bff5b2ab3ac65d226d59e592db4cb7cf8f6
contracts/Router.sol	71211276b89cc53bb9d1cafbdbbd1d91901831f8
contracts/ARXFlexiblePool.sol	bd5ae332467a2591d7b208b0520d648812d1b3b2
contracts/ARXPool.sol	15b252a171eb88709c36c26a0152bad6b03ca4a9
contracts/SmartChefFactory.sol	69a545a842209395aefef55ad943a1a63f7cfa11
contracts/ARXToken.sol	f77e2d2fda1913d5fa28163ad883d0eeb63768f6
contracts/Factory.sol	53403cd923dfb92ad027e28952e4820d3ddbcd1a

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
17	18	19	9

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
360	10

External	Internal	Private	Pure	View
271	490	14	66	158

StateVariables

Total	🌐Public
144	112

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts

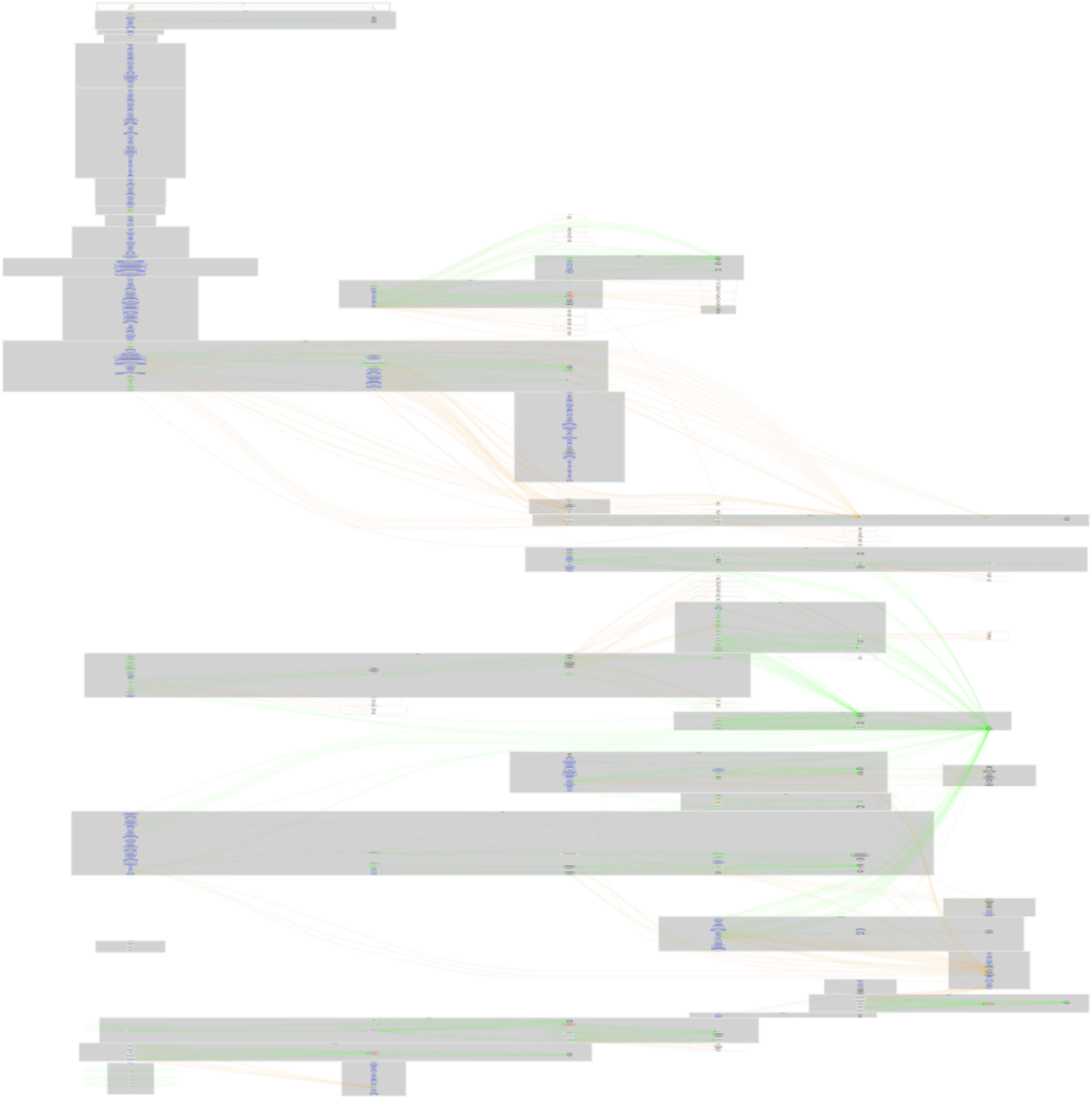
`0.6.12`
`>=0.6.0`
`>=0.6.2`
`>=0.5.0`
`=0.6.6`
`^0.8.0`
`=0.5.16`
		`yes`	`yes`
			(14 asm blocks)
				—

💸 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	✒️ ECRecover	🌀 New/Create/Create2
`yes`	—	`yes`	`yes`	`yes`	`yes` → `AssemblyCall:Name:create2`

♻️ TryCatch	Σ Unchecked
—	`yes`

Inheritance Graph
v1.0

CallGraph
v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Overall checkup (Smart Contract Security)

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers and public functions
v1.0

MasterChefv2.sol

- setTreasury
  - onlyOwner
- updateArxPerSec
  - onlyOwner
- updateWETHPerSec
  - onlyOwner
- updateMultiplier
  - onlyOwner
- add
  - onlyOwner
- set
  - onlyOwner
- massUpdatePools
- updatePool
- deposit
- withdraw
- emergencyWithdraw
- setStartTime
  - onlyOwner

SmartChefFactory.sol

- initialize
- deposit
  - nonReentrant
- withdraw
  - nonReentrant
- emergencyWithdraw
  - nonReentrant
- emergencyRewardWithdraw
  - onlyOwner
- recoverToken
  - onlyOwner
- setTreasury
  - onlyOwner
- stopReward
  - onlyOwner
- updateRewardPerBlock
  - onlyOwner
- updateStartAndEndBlocks
  - onlyOwner

ARXPool.sol

- unlock
  - onlyOwner
  - whenNotPaused
- deposit
  - whenNotPaused
- withdrawByAmount
  - whenNotPaused
- withdraw
  - whenNotPaused
- withdrawAll
- setTreasury
  - onlyOwner
- setFreePerformanceFeeUser
  - onlyOwner
- setOverdueFeeUser
  - onlyOwner
- setWithdrawFeeUser
  - onlyOwner
- setPerformanceFee
  - onlyOwner
- setPerformanceFeeContract
  - onlyOwner
- setWithdrawFee
  - onlyOwner
- setOverdueFee
  - onlyOwner
- setWithdrawFeeContract
  - onlyOwner
- setWithdrawFeePeriod
  - onlyOwner
- setMaxLockDuration
  - onlyOwner
- setDurationFactor
  - onlyOwner
- setDurationFactorOverdue
  - onlyOwner
- setUnlockFreeDuration
  - onlyOwner
- inCaseTokensGetStuck
  - onlyOwner
- pause
  - onlyOwner
  - whenNotPaused
- unpause

ARXFlexiblePool

- deposit
  - whenNotPaused
- withdraw
- withdrawAll
- setTreasury
  - onlyOwner
- setPerformanceFee
  - onlyOwner
- setWithdrawFee
  - onlyOwner
- setWithdrawFeePeriod
  - onlyOwner
- setWithdrawAmountBooster
  - onlyOwner
- emergencyWithdraw
  - onlyOwner
- inCaseTokensGetStuck
  - onlyOwner
- pause
  - onlyOwner
  - whenNotPaused
- unpause

Note:

- General fork from PancakeSwap
- PancakeSwap
  - Contracts inside are the same as the pancake-smart-contracts directory
    - https://github.com/pancakeswap/pancake-smart-contracts/tree/master/projects
    - Differences between ArbiDEX and PancakeSwap contracts are the following:
      - SmartChefFactoryInitializable has the same logic except the "pool limit per user functionality" has been removed.
      - MasterChefv2 contract does not have the staking functionality as it does in the Pancake swap
      - Boost Weight functionality of CakePool has been removed from ARXPool
      - Mint Liquidity has been changed in the factory contract to 20/25, and a swap fee of 0.25% will be charged in the ArbiDexPair contract
      - ArbiDexRouter has no modified functionalities

Ownership Privileges

- *MasterChefv2.sol* -
  - Set Treasury address
  - Update ArxPerSec, WETHPerSec, and Multiplier to any arbitrary value.
  - Add new Lp to the pool
  - Set allocation point for a given ARX
  - Set Start Time

- *SmartChefFactory.sol* -
  - Withdraw reward tokens (Beware of it)
  - Withdraw other tokens that are accidentally sent to the contract, but cannot withdraw staked token by any means.
  - Set treasury address
  - Update reward block
  - Stop reward
  - Update start and end blocks but not after the pool has started

- *ARXPool.sol* -
  - Unlock user ARX funds only when the contract is not paused
  - Pause/Unpause Deposits and Withdraws
  - Set treasury address, fee address, overdue fee address, and withdraw fee address

    - Set performance fee but cannot be set more than 20%
    - Set withdraw fee but cannot be set more than 5%
    - The overdue fees can be set up to 100%. This fee will only be levied based on users overdue duration
    - Set withdraw fee contract and period
    - Set max lock duration but not more than 1000 days, but keep in mind that it could be set to zero.
    - Set duration factor, duration factor overdue, and unlock free duration
    - Withdraw unexpected tokens from the contact, but not the staked one

- *ARXFlexiblePool.sol*
    - Pause/Unpause Deposits
    - Set treasury address
    - Set performance, and withdraw fee
    - Set withdraw fee period and amount booster
    - Owner can withdraw their shares while the staking is true. Once it it done no more staking could be done in the contract.
    - Withdraw unexpected tokens from the contract but not the deposit tokens

- *ARXToken.sol*
    - Owner can Mint tokens but not more than max supply which is 20 Million

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope
v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/MasterChefv2.sol	8	1	1520	1336	668	627	487
contracts/Router.sol	4	6	824	427	355	40	579
contracts/ARXFlexiblePool.sol	6	2	947	799	421	407	288
contracts/ARXPool.sol	6	2	1309	1170	657	525	410
contracts/SmartChefFactory.sol	7	2	915	754	370	374	292
contracts/ARXToken.sol	7	1	1206	1022	434	571	300
contracts/Factory.sol	6	5	498	404	325	48	435
Totals	**44**	**19**	**7219**	**5912**	**3230**	**2592**	**2791**

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	All	Multiple pragma is set	—	Some of the contracts contain different pragma versions which is not recommended for deployment. We recommend to have the same pragma in all contracts and also to update the old pragma versions to the new ones.
#2	MasterChefv2.sol	Missing Zero Address Validation (missing-zero-check)	1301	Check that the address is not zero
#3	MasterChefv2.sol	Missing Events Arithmetic	1301-1313, 1323, 1343	Emit an event for critical parameter changes
#4	Router.sol	Old Compiler Version	388	The contract uses a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities

#5	Factory.sol	Old Compiler Version	5	The contract uses a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities
#6	MasterChefv2.sol	Old Compiler Version	7	The contract uses a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities

Informational issues

Issue	File	Type	Line	Description
#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	—	We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

05. April 2023:

- This project consists of the following forks
  - Pancake Swap
  - Uniswap
- Read whole report and modifiers section for more information
- The low issues that exist in the PancakeSwap codebase still exist in the forked code.
- We recommend using a multisig wallet for the owner address to prevent any risk of the loss of private key
- Do your own research here

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

SolidProof_io

@solidproof_io

SolidProofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY