SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

MechaChain

Audit

Security Assessment

11. February, 2022

For

MECHACHAIN

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	11. February 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://mechachain.io/en/

Telegram
https://t.me/mechachain

Twitter
https://twitter.com/mechachain

Discord
https://discord.gg/kMJCNaWaNz

Description

MechaChain is a 3D play-to-earn video game about robot combat and space conquest. Each robot, called “Mecha”, is a collection of NFT composed of robot parts, which can be purchased online with the game cryptocurrency called Mechanium, Ethereum, or by card. These parts once assembled give birth to a robot in a PvP fighting video game.

The player earns Mechanium by winning battles, and can trade and buy new parts to become the best MechaChain pilot.

Project Engagement

During the 9th of February 2022, **MechaChain 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

MECHACHAIN

Contract Link

v1.0

- Github
  - https://github.com/thibautvdu/MechaChain-Smart-Contracts/tree/develop/contracts
  - Commit: 678acde6e82a5c049c8e0707cb8095cfa8c3218e

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:

Dependency / Import Path	Count
@openzeppelin/contracts/access/AccessControl.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	5
@openzeppelin/contracts/utils/Counters.sol	4
@openzeppelin/contracts/utils/math/SafeMath.sol	3

Tested Contract Files

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

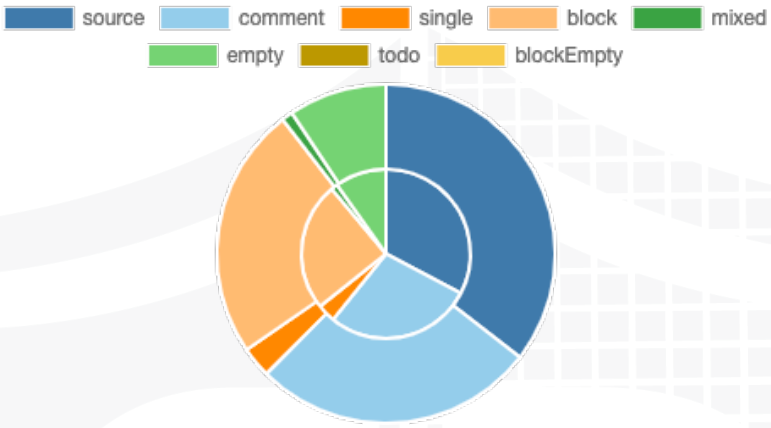
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

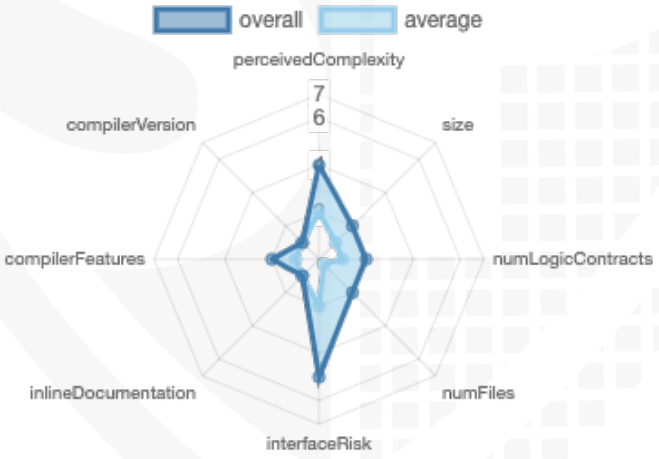
File Name	SHA-1 Hash
contracts/MechaniumTeamDistribution.sol	90cb4cc185119a3d9697215287c0fa9f415697c3
contracts/MechaniumDevDistribution.sol	c8be3324b825d6d8320de4aca4599a31510d21f7
contracts/MechaniumAdvisorsDistribution.sol	d6b6bc15af8fd71c03c8e29a39c29ab2ca90c3c1
contracts/IMechaniumVesting.sol	dcfa96b40c919cf702816e56b8f0a86fe9c46755
contracts/MechaniumPresaleDistribution.sol	595a7a404b3cbebade7d18cb4c1a09937df50926
contracts/MechaniumFoundersDistribution.sol	c5c566826f7dfc9a139d9c7c50b93064c5d63765
contracts/MechaniumVesting.sol	905c4b57c97c5705df1a8c209ad4b87ce407929b

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	0	1	1

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	65	0

Version	External	Internal	Private	Pure	View
1.0	21	28	0	0	49

State Variables

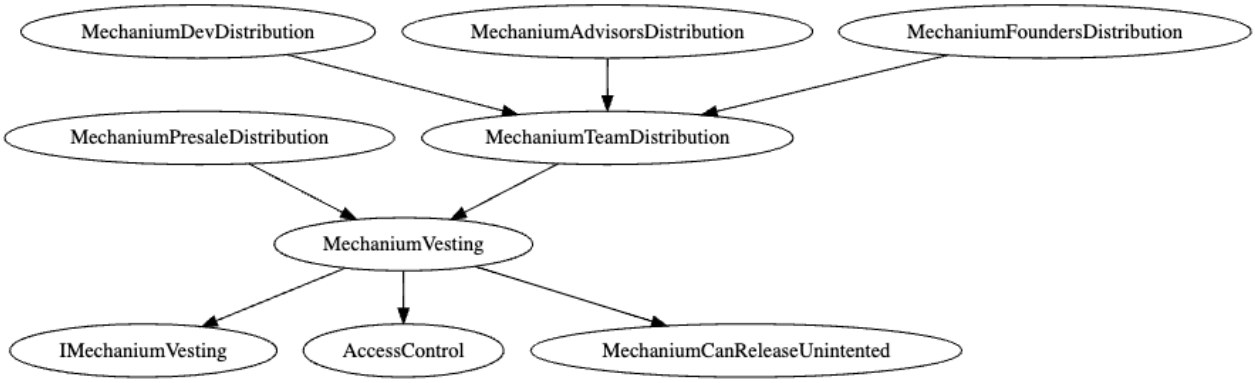
Version	Total	Public
1.0	22	1

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.2				

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2
1.0				yes		

Inheritance Graph
v1.0



CallGraph

v1.0

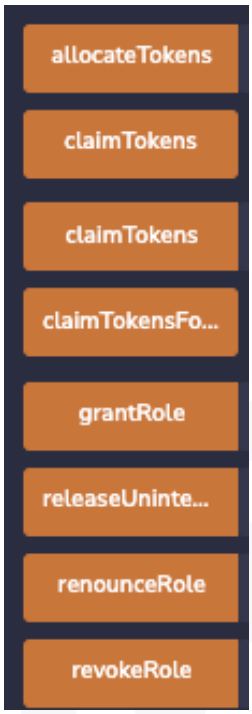
Scope of Work/Verify Claims

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).

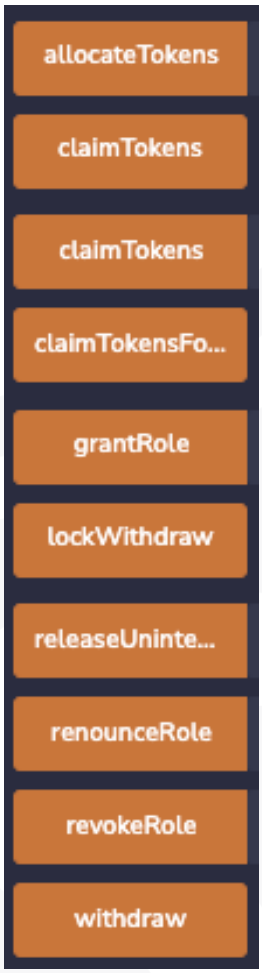
We will verify the following claims:

1. Overall checkup (Smart Contract Security)

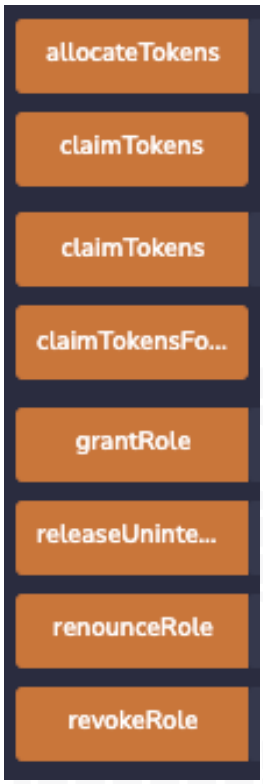
Write functions of contract v1.0



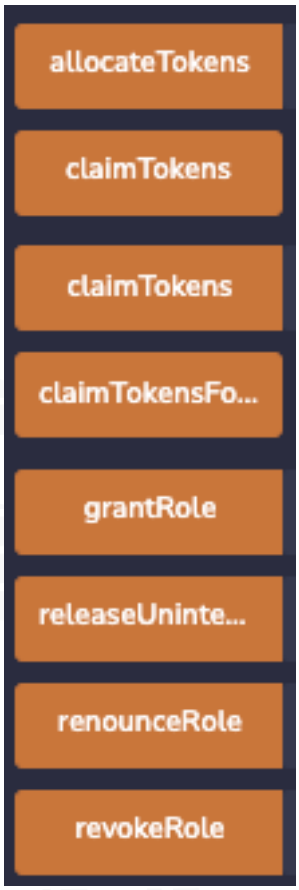
MechaniumVesting



MechaniumFounderDistribution



MechaniumDevDistribution



MechaniumAdvisorsDistribution

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers and public functions

v1.0

- allocateTokens
  - onlyRole
  - tokensAvailable
- allocateTokens
- claimTokens
- claimTokensForAll
  - onlyRole
- grantRole
  - onlyRole
- revokeRole
  - onlyRole
- renounceRole
- releaseUnintented
  - onlyRole

- allocateTokens
  - onlyRole
  - tokensAvailable
- startVesting
  - vestingNotStarted
  - onlyRole
- setStakingPool
  - onlyRole
- setStakingTransferTimeLimit
  - onlyRole
- setMinimumStakingTime
  - onlyRole
- transferToStakingPool
- allocateTokens
- claimTokens
- claimTokensForAll
  - onlyRole
- lockWithdraw
  - onlyRole
- withdraw
  - onlyRole

Comments

- MechaniumFoundersDistribution
  - If _lockWithdraw is set to true with lockWithdraw function (MechaniumFoundersDistributions L67), you cannot change back to false

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

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/MechaniumTeamDistribution.sol	1	————	251	214	112	74	84	————
📝	contracts/MechaniumDevDistribution.sol	1	————	28	28	14	14	4	————
📝	contracts/MechaniumAdvisorsDistribution.sol	1	————	28	28	14	14	4	————
🔍	contracts/IMechaniumVesting.sol	————	1	100	14	3	61	37	————
📝	contracts/MechaniumPresaleDistribution.sol	1	————	361	315	150	119	96	————
📝	contracts/MechaniumFoundersDistribution.sol	1	————	124	120	52	53	33	————
🎨	contracts/MechaniumVesting.sol	1	————	389	331	150	154	121	🧮
📝 🔍 🎨	**Totals**	**6**	**1**	**1281**	**1050**	**495**	**489**	**379**	🧮

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized 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

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	IMechaniumCanReleaseUnintented	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#2	IMechaniumVesting	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#3	MechaniumAdvisorsDistribution	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#4	MechaniumCanReleaseUnintented	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.

#5	MechaniumDevDistribution	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#6	MechaniumFoundersDistribution	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#7	MechaniumPresaleDistribution	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#8	MechaniumTeamDistribution	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#9	MechaniumVesting	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.2””.
#10	MechaniumPresaleDistribution	Missing Zero Address Validation (missing-zero-check)	201	Check that the address is not zero

Informational issues

Issue	File	Type	Line	Description
#1	IMechaniumCanReleaseUninented	MIsspelling	See description	Change following: - Mechanim to Mechanium L5 - IMechaniumCanReleaseUnintented to IMechaniumCanReleaseUnintended L10 - unintented to unintended L12, L12, L13 - releaseUnintented to releaseUnintended L18 Change variables/functions/interfaces/imports etc. everywhere else

#2	IMechaniumVesting	MIsspelling	See description	Change following: - Mechanim to Mechanium L5 - amont to amount L42 Change variables/functions/interfaces etc. everywhere else
#3	MechaniumCanReleaseUnintented	MIsspelling	See description	Change following: - IMechaniumCanReleaseUnintented to IMechaniumCanReleaseUnintended L6, L14 - ReleaseUintentedTokens to ReleaseUnintendedTokens L23, L83 - unintented to unintended L21, L47 - releaseUnintented to releaseUnintended L52 Change variables/functions/interfaces etc. everywhere else
#4	MechaniumFoundersDistribution	MIsspelling	See description	Change following: - whitdraw to withdraw L10, L72, L84 Change variables/functions/interfaces etc. everywhere else
#5	MechaniumPresaleDistribution	MIsspelling	See description	Change following: - tokens to tokens L59 - getStrakingTransferTimeLimit to getStakingTransferTimeLimit L351 Change variables/functions/interfaces etc. everywhere else

#6	MechaniumVesting	MIsspelling	See description	Change following: - MechaniumCanReleaseUnintented to MechaniumCanReleaseUnintended L8, L19 - transfered to transferred L87, L96 Change variables/functions/interfaces etc. everywhere else

Testing Protocol

MechaniumFoundersDistribution

✓ Smart contract should be deployed (139ms)
✓ Admin should set allocator role
✓ Allocator should allocate to user
1) Allocator should not be able to allocate tokens to smart contract address
> No events were emitted
✓ User balance should encrease when allocated tokens
✓ Total unlockable tokens must be 20% of the first allocation (1 year after the first allocation)
✓ Total unlockable tokens must be 20% of the two allocations (1 year after the seconds allocation)
✓ Total unlockable tokens must be 40% of the first allocation + 20% of the seconds (1 year + 6 month)
✓ Total unlockable tokens must be 100% of all allocation
✓ User should be able to claim unlockable tokens
✓ Allocator should allocate new tokens to user
✓ Total unlockable tokens must be 20% of the new allocation (1 year after the new allocation)
✓ Admin should withdraw (39ms)
✓ User should not have unlockable tokens anymore
2) User should not be able to lock withdraw
> No events were emitted
✓ Admin should lock withdraw
3) Admin should not be able to relock withdraw
> No events were emitted
4) Admin should not be able to withdraw if already locked
> No events were emitted
5) User should not be able to withdraw
> No events were emitted
✓ Admin can add new tokens supply

✓ Allocator should allocate to user
✓ User balance should encrease when allocated tokens
✓ Total unlockable tokens must be 20% of the first allocation (1 year after the first allocation)
✓ Total unlockable tokens must be 20% of the two allocations (1 year after the seconds allocation)
✓ Total unlockable tokens must be 40% of the first allocation + 20% of the seconds (1 year + 6 month)
✓ Someone should claim user's unlockable tokens

MechaniumPresaleDistribution

✓ Smart contract should be deployed (171ms)
✓ Allocator account should not have ALLOCATOR_ROLE
✓ Admin should be able to set ALLOCATOR_ROLE
14) Allocator should not be able to allocate amount superior to balance
> No events were emitted
15) User should not be able to allocate tokens
> No events were emitted
✓ Contract should have 10M of balance
✓ Vesting should not be started if starting time has not arrived
✓ Allocator should be able to allocate tokens to multiple users (62ms)
✓ Allocator should be able to allocate tokens to user
16) Allocator should not be able to allocate tokens to smart contract address
> No events were emitted
✓ User balance should increase after allocation
✓ User balance should be locked if vesting time not started
✓ Admin should be able to change vesting start time
17) Admin should not be able to set vesting start time after max start time
> No events were emitted
18) Admin should not be able to claim user tokens if vesting has not started
> No events were emitted
✓ Admin should be able to start the vesting immediatly
✓ Unlockable amount must be 20% of total user balance (first month)
✓ Unlockable amount must be 20% of total user balance (first month)
✓ Pending tokens must be calculated per seconds
✓ Admin should claim user's unlockable tokens (20%)
19) Admin should not be able to claim user's tokens in same period
> No events were emitted
20) User shoud not be able to transfer to staking pool if the staking pool is not set
> No events were emitted
✓ Admin shoud be able to set staking pool address

21) User shoud not be able to transfer if staking time lower than minimum staking time
> No events were emitted
22) User shoud not be able to transfer if staking time greater than maximum staking time
> No events were emitted
23) User shoud not be able to transfer an amount of 0
> No events were emitted
24) User shoud not be able to transfer an amount superior to his allocated tokens
> No events were emitted
25) User shoud not be able to set staking pool address
> No events were emitted
✓ User shoud be able to transfer an amount of unlocked tokens to staking pool
✓ Admin can refill the contract
✓ Allocator should be able to allocate tokens a month after the vesting has started (after 30 days)
✓ The new late beneficiary can claim 40% of his allocation
✓ Unlockable amount must be 60% of total allocated tokens (after 60 days)
✓ Admin should claim user's unlockable tokens (60%)
✓ Unlockable amount must be 100% of total user balance (after 5 months)
✓ Admin should claim user's unlockable tokens (100%)
26) Admin should not be able to claim user's tokens after they where all claimed
> No events were emitted
✓ Allocator should be able to allocate tokens after the end of the vesting period
✓ The new late beneficiary can claim 100% of his allocation
27) Admin should claim all users tokens

Audit Comments

11. February 2022:

- Read whole report for more information

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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY