SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Zoombies

Audit

Security Assessment

18. April, 2023

For

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	14. April 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	18. April 2023	• Reaudit

Network
Moonbeam

Website
https://www.zoombies.info/
https://zoombies.world/moonbeam

Twitter
https://twitter.com/cryptoznft

Medium
https://cryptoz-cards.medium.com/

Discord
https://discord.gg/eDXvJKUZgQ

Youtube
https://www.youtube.com/@zoombies

Description

Zoombies is an infinite world of NFT collectible trading cards built on Rarity, Scarcity and Community.

The Zoombies World brings all the value and excitement of blockchain NFTs into a tidy free-to-play mix of predictable rules that result in unpredictable outcomes. Zoombies has something for everyone, and when luck strikes, it can take you out of this world! Zoombies is a full cycle token economy which allows players to mint, trade and burn NFTs alongside the ZOOM ERC-20 standard token for highly liquid value transfer.

Project Engagement

During the 12th of April 2023, **Zoombies 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 Link

v1.0

- Zoom ERC20 Moonbeam
  - https://moonbeam.moonscan.io/token/0xc46c5cb32a72663c0db3205f6b444f9c34e216d1
- Zoombies ERC721 Moonbeam
  - https://moonbeam.moonscan.io/token/0xd6e8a1d5e0675168af078391ec3abd983eb18ea0

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/Ownable.sol	2
@openzeppelin/contracts/security/Pausable.sol	2
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	1
@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol	1
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	1
@openzeppelin/contracts/utils/Counters.sol	1

Tested Contract Files

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

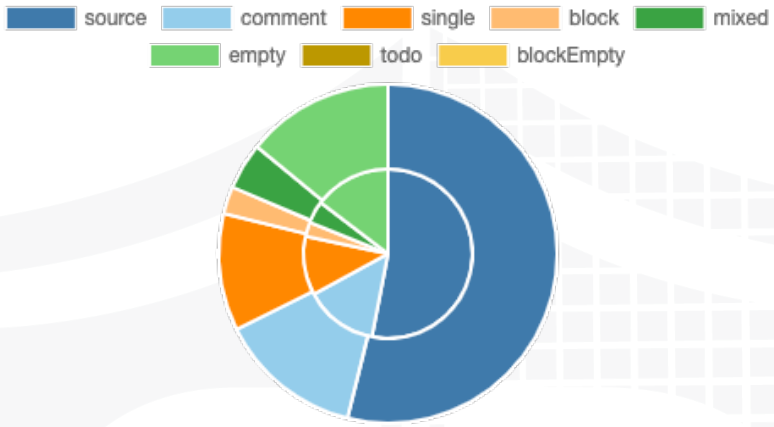
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

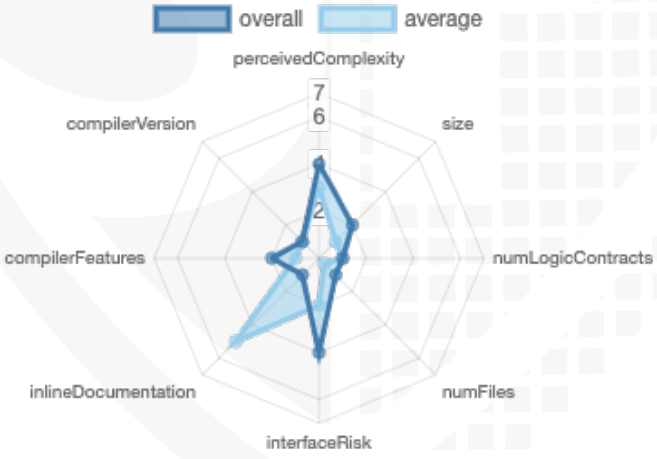
File Name	SHA-1 Hash
contracts/ZoombiesUniverse.sol	4b97d51e58ce0ab7ef60ef940919b6af44bb7d7c
contracts/Zoombies.sol	d1791be84dea9b0cda7875a1af1c2ce0252d03d5
contracts/ZoomToken.sol	df7365043fed9325551f7f02339ad37ded9d44b2

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	0	0	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	31	5

Version	External	Internal	Private	Pure	View
1.0	21	46	4	1	5

State Variables

Version	Total	Public
1.0	29	25

Capabilities

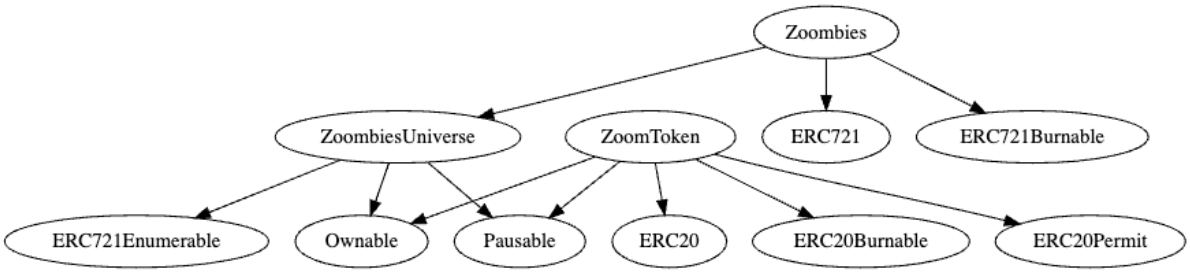
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.9`		`yes`		

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2

1.0	yes			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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

Correct implementation of Token standard

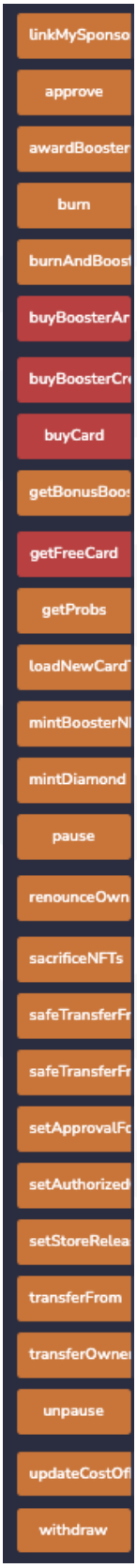
ERC20				
Function	**Description**	**Exist**	**Tested**	**Verified**
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

ERC721				
Function	**Description**	**Exist**	**Tested**	**Verified**
BalanceOf	Count all NFTs assigned to an owner	✓	✓	✓
OwnerOf	Find the owner of an NFT	✓	✓	✓
SafeTransferFrom	Transfers the ownership of an NFT from one address to another address	✓	✓	✓
SafeTransferFrom	See above - Difference is that this function has an extra data parameter	✓	✓	✓
TransferFrom	Transfer ownership of an NFT	✓	✓	✓
Approve	Change or reaffirm the approved address for an NFT	✓	✓	✓
SetApprovalForAll	Enable or disable approval for a third party ("operator") to manage all of `msg.sender`'s assets	✓	✓	✓
GetApproved	Get the approved address for a single NFT	✓	✓	✓
IsApprovedForAll	Query if an address is an authorized operator for another address	✓	✓	✓
SupportsInterface	Query if a contract implements an interface	✓	✓	✓
Name	Provides information about the name	✓	✓	✓
Symbol	Provides information about the symbol	✓	✓	✓
TokenURI	Provides information about the TokenUri	✓	✓	✓

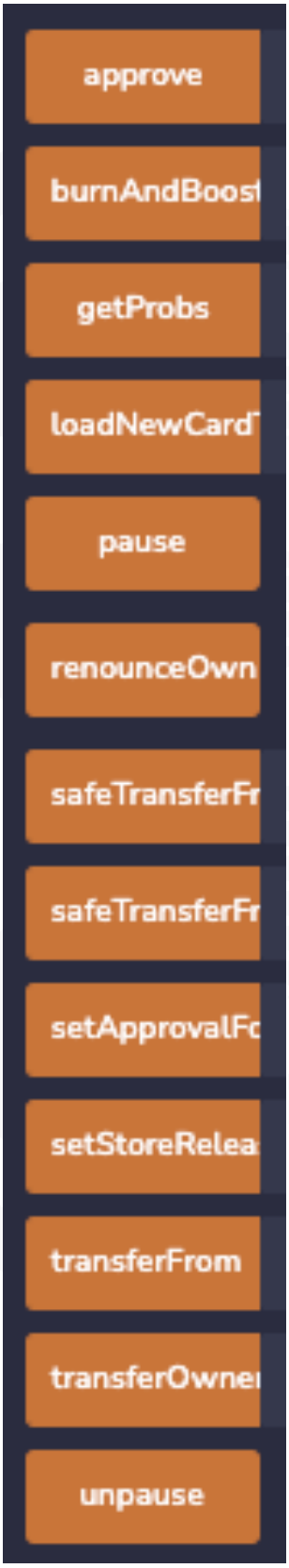
Write functions of contract v1.0

ZoomToken

Zoombies



ZoombiesUniverse



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

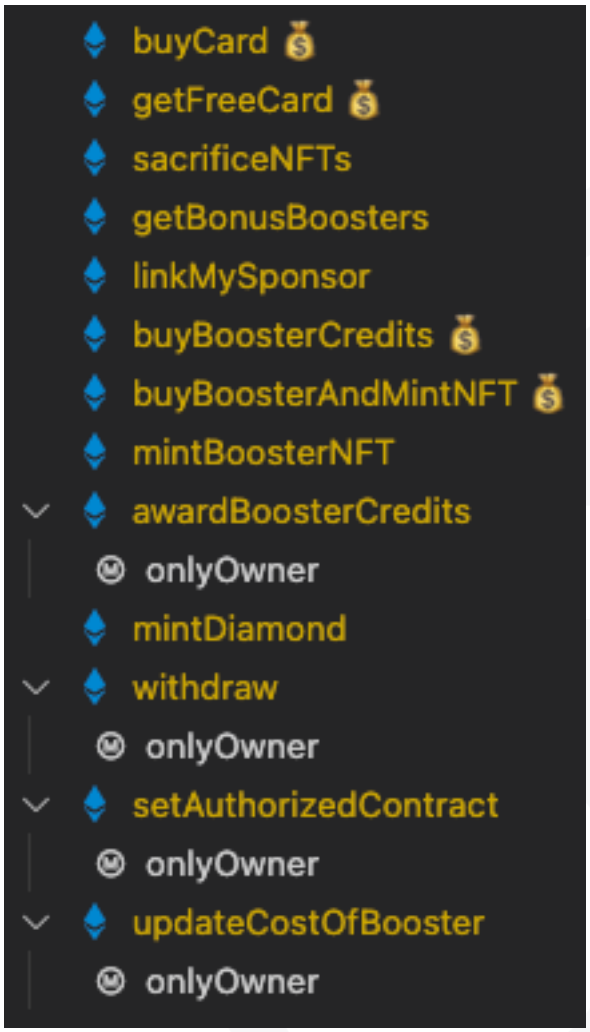
Legend

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

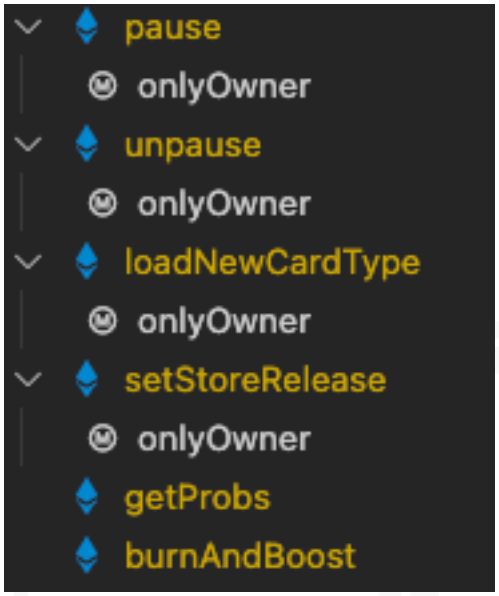
Modifiers and public functions

v1.0

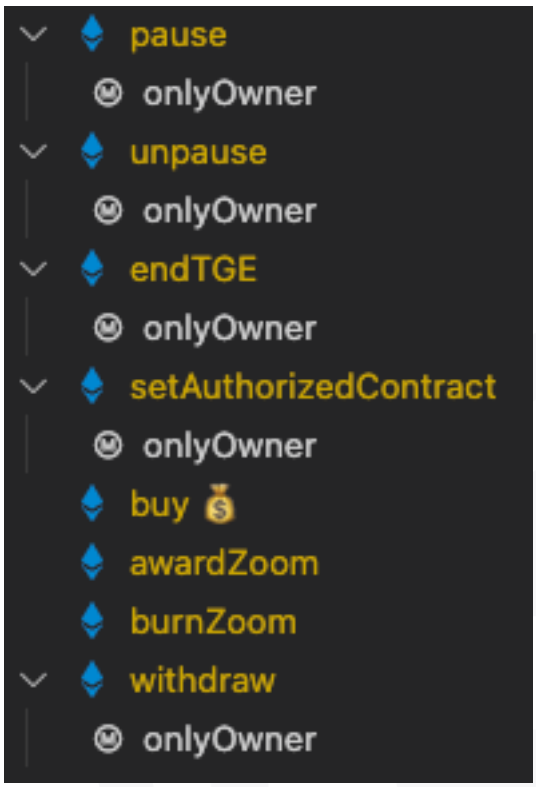
Zoombies



ZoombiesUniverse



ZoomToken



Note: Imported libraries were not listed down below

Comments

- ZoomToken
  - Only owner is able to
    - Pause/unpause
    - End TGE only once
    - Set authorized contract
    - Withdraw tokens to his own wallet
  - Only authorized contract is able to
    - Mint new zoom tokens without limitations. Be aware of it
    - Burn zoom
      - Without any allowances of an arbitrary address
- Zoombies
  - Only owner is able to
    - Set award booster credits without any limitations
    - Set authorized contract address
    - Set wei cost of boost without any limitations
    - Pause the contract to lock funds
  - Only authorized contract is able to
    - Mint diamonds
- ZoombiesUniverse
  - Only owner is able to
    - Pause / unpause

- Load new card type
- Set store release

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/ZoombiesUniverse.sol	1	———	246	233	156	40	92	🧱
📝	contracts/Zoombies.sol	1	———	358	349	246	64	205	💰💠
📝	contracts/ZoomToken.sol	1	———	130	126	83	27	89	💰💠
📝🦠	**Totals**	**3**	———	**734**	**708**	**485**	**131**	**386**	💰💠🧱

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	ZoombiesUniverse	Randomness	178	It is recommended to use an external service as chainlink to get a random number instead of generating it out of the contract because of frontrunning.

Informational issues

Issue	File	Type	Line	Description
#1	Zoombies	Error message is missing	See description	Provide an error message for require statement. Look for every “require” statements and add an error message to them.
#2	All	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.

#3	ZoombiesUniverse	Gas saving	See description	To save gas while using the contract it is recommended to order the structs by the type of them. For more informations read the following: https://dev.to/javier123454321/solidity-gas-optimizations-pt-3-packing-structs-23f4
#4	ZoomToken	Static numbers	64, 128	It is recommend not to set static numbers in the contract. Use instead state variables that the owner is able to update because of some reasons. If you are going to implement a function for it it is also recommended to add a range while setting the variable.
#5	ZoombiesUniverse	Tautology or contradiction	122	Fix the incorrect comparison by changing the value type or the comparison. The "_hoursFromNow" variable is an uint type.

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.

18. April 2023:

- Zoomtokens authorized contracts/EOA are able to mint new tokens without any limitations. Additionally to it they are also able to burn without any allowances. Be aware of it.
- Read whole report and modifiers section 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	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

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY