



SOLIDProof

Bring trust into your projects

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

MADE IN GERMANY

MyNFTScore

Audit

Security Assessment
05. April, 2023

For



SolidProof_io



@solidproof_io

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	18
Source Units in Scope	21
Critical issues	22
High issues	22
Medium issues	22
Low issues	22
Informational issues	23
Audit Comments	24
SWC Attacks	25

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 Uniswap, 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	05. April, 2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://www.mynftscore.io/>

Telegram

<https://t.me/+U70ib07PndZjMjFi>

Twitter

<https://mobile.twitter.com/MyNFTScore>

Instagram

<https://instagram.com/mynftscore?igshid=YmMyMTA2M2Y=>

Youtube

<https://youtube.com/@MyNFTScore>

Description

First decentralized Gamified Earning Protocol DEFI Platform.

Project Engagement

During the 22nd of March 2023, **MyNFTScore 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

- MyNFTScore721
 - <https://bscscan.com/address/0x3Fd553c7270AB9cDC922E1Bfb9E3222aCBCf9641>
- MyNFTScoreECDSA
 - <https://bscscan.com/address/0x81E2ffB7F3043bCA328338AA64e9DC25F17262B9>
- MyNFTScoreCycle
 - <https://bscscan.com/address/0x81E2ffB7F3043bCA328338AA64e9DC25F17262B9>
- MyNFTScore Proxy
 - <https://bscscan.com/address/0x85c9cDB201537916FE561205b2D9BE46b0DACECb>
- MyNFTScore implementation
 - <https://bscscan.com/address/0xEC169E32710Cf5CdB139321f01e1100244A81e2C>

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 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 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-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol	1
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol	1
@openzeppelin/contracts/utils/Base64.sol	1
@openzeppelin/contracts/utils/Strings.sol	1
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	1

Tested Contract Files

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

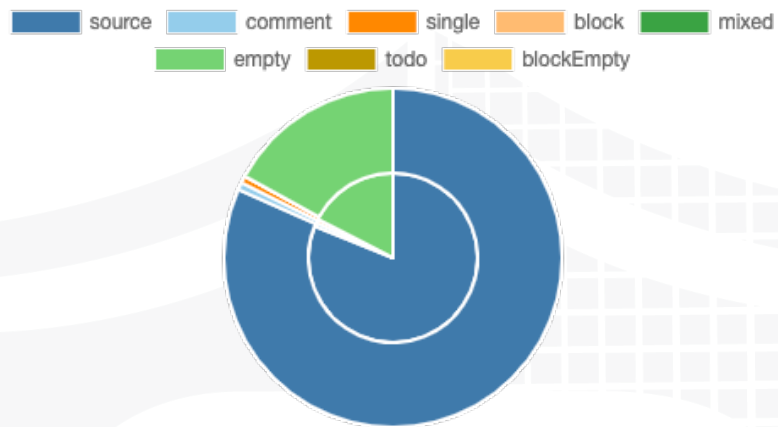
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

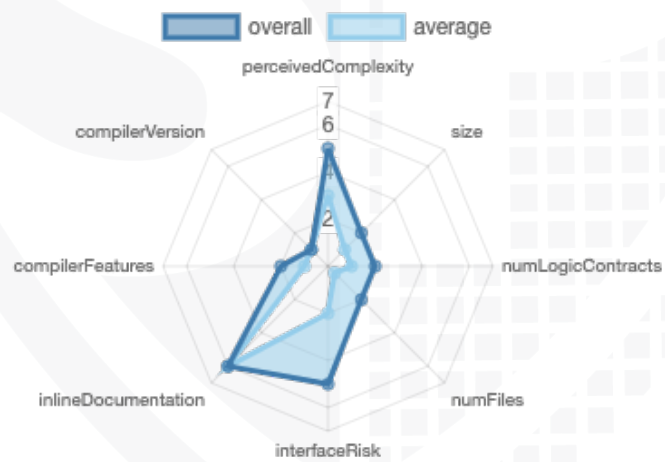
File Name	SHA-1 Hash
contracts/tokens/MyNFTScore721.sol	bd18185171705fdedd5c7e508853d30fb5ca3e58
contracts/MyNFTScoreCycle.sol	50f2525370dd675494f158035dfd35fab4cb5ff9
contracts/MyNFTScore.sol	1d3017ddd0521420b102b4a10825efb4367de8ab
contracts/MyNFTScoreData.sol	c4fbc8453b174ad530839d77af9c86c76456c967
contracts/utils/MyNFTScoreECDSA.sol	159e6277c6f7551b5616fe0d868380d2e08d9992

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	1	0	0

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	57	4

Version	External	Internal	Private	Pure	View
1.0	38	63	2	6	22

State Variables

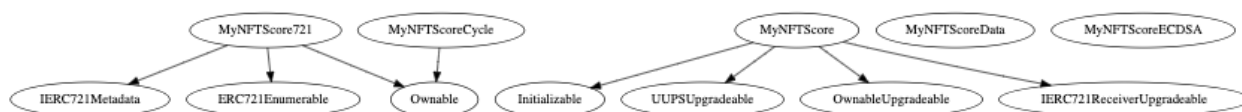
Version	Total	Public
1.0	42	11

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.17 ^0.8.17 ^0.8.4		yes		

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Overall checkup (Smart Contract Security)



Is contract an upgradeable

Name	
Is contract an upgradeable?	Yes

Comments:

v1.0

- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
 - Be aware of this and do your own research for the contract which is the contract pointing to

Correct implementation of Token standard

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	✓	✓	✓

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

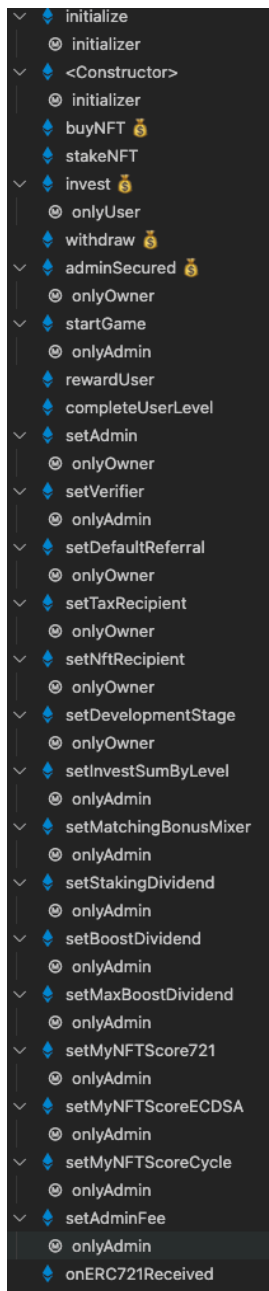
Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

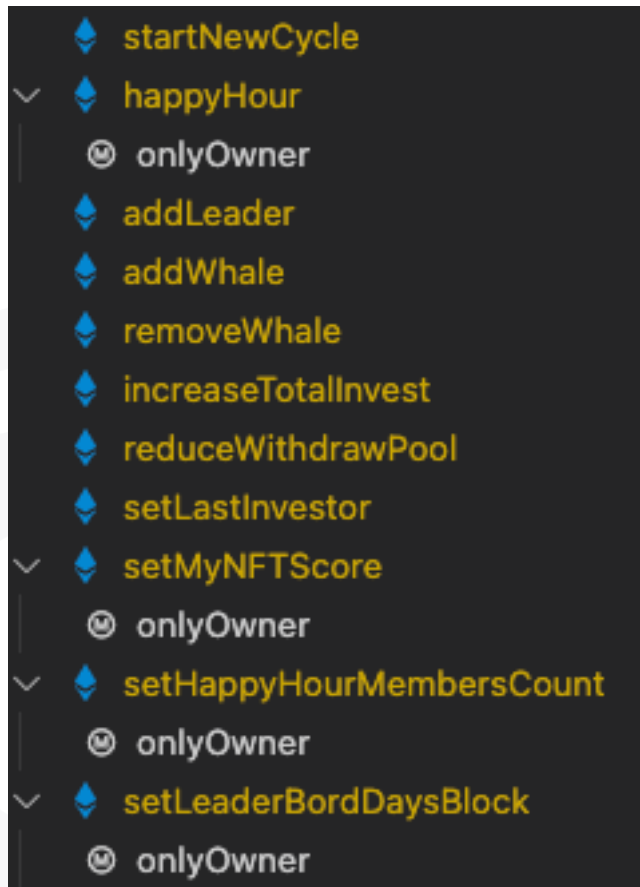
Modifiers and public functions

v1.0

MyNFTScore



MyNFTScoreCycle



Comments

- Deployer can set following state variables without any limitations
 - MyNFTScore
 - adminFee
 - boostStartDay
 - maxBoostDividend
 - boostDividend
 - stakingDividend

- matchingBonusMixer
 - incomeReferralBonusMixer
 - investSumNByLvl
 - hardStop.maxWithdrawPercent
- MyNFTScoreCycle
 - leaderBordDaysBlock
 - happyHourMembersCount
 - cycles.withdrawPoolLeft
 - cycles.totalInvest
- Deployer can enable/disable following state variables
 - MyNFTScore
 - developmentStage.withdrawStop
 - developmentStage.investStop
 - developmentStage.rewardStop
- Deployer can set following addresses
 - MyNFTScore
 - myNFTScoreCycle
 - myNFTScoreECDSA
 - myNFTScore721
 - defaultReferral
 - nftRecipient
 - taxRecipient
 - verifier
 - admin
 - MyNFTScoreCycle
 - myNFTScore
 - lastInvestor
- Existing Modifiers
 - MyNFTScore
 - onlyUser
 - onlyOwner
 - onlyAllOwners
- MyNFTScore
 - Owner is able to
 - Set fees without any limitations. Be aware of it because it can lock user funds
 - Modify rewards with an external contract
 - myNFTScoreCycle can be updated by the owner that can call the rewardUser function to modify arbitrary











addresses with a random number. There are no limitations for the setting

- Withdraw native tokens from the contract by calling the adminSecured function
- All owners are able to
 - Stop withdraw functionality
 - Stop invest functionality
 - Stop reward functionality
- Verifier is able to
 - Set a new verifier
- MyNFTScoreCycle
 - Only verifier is able to
 - Start cycle
 - Only owner is able to
 - Call the happy hour and distribute the happy hour reward to the last investor
 - Only nft score contract is able to
 - Add a new leader
 - Add/remove a whale
 - Increase the total invest
 - Reduce wutgdraw pool amount
 - Set last investor
 - Caller will be added to whale list if the user level is 14. Additionally to it if the user buy a NFT with level of 15 he will get the fast pass.
- MyNFTScore721
 - MyNFTScore address is able to
 - Burn token id
 - Mint new token
 - Owner is able to
 - Update myNftScoreAddress
 - Update base URI
 -

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/tokens/MyNFTScore721.sol	1	————	143	139	113	1	72	————
	contracts/MyNFTScoreCycle.sol	1	————	184	184	146	1	114	————
	contracts/MyNFTScore.sol	1	————	681	666	544	4	369	
	contracts/MyNFTScoreData.sol	1	————	88	88	78	1	1	————
	contracts/utlis/MyNFTScoreECDSA.sol	1	————	43	32	25	1	20	
	Totals	5	————	1139	1109	906	8	576	 

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	Main	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	MyNftS coreData	A floating pragma is set	Top of file	The current pragma Solidity directive is a floating pragma version. It is recommended to use a certain version.
#3	MyNFTS coreCycle	A floating pragma is set	Top of file	The current pragma Solidity directive is a floating pragma version. It is recommended to use a certain version.
#4	MyNFTS core	Missing Zero Address Validation (missing-zero-check)	254, 597, 605, 613, 609, 601	Check that the address is not zero
#5	MyNFTS core721	Missing Zero Address Validation (missing-zero-check)	106	Check that the address is not zero
#6	MyNFTS coreCycle	Missing Zero Address Validation (missing-zero-check)	31, 168	Check that the address is not zero

#7	MyNFTS core721	Local variables shadowing	23, 102	Rename the local variables that shadow another component
#8	MyNFTS core	Missing Events Arithmetic	643, 636, 648, 640	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	All	Error message is missing	See all require statements	Provide an error message for require statement
#2	All	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#3	MyNFTS coreCyc le	Endtime and starttime were not used in the contract	50, 53	Make sure to use start/end- time in the contract.
#4	MyNFTS core	NFT level	134	Since the buyer is able to pass the NFTLevel to the buyNFT function he can also pass a level above 15. That means that the caller can bypass the fastPass level before getting it. The condition is only checking for the nft level of 15.
#5	MyNFTS core	Staking nft	188	While staking a NFT there is no check that the passed token ID was already staked. But it is pushed into the tokenIds array.
#6	MyNFTS core	Length of invests	200	It is recommended to check that the user level - 1 is below the investSumbNByLvl array length
#7	MyNFTS core	Zero check	200	If the investSumNByLvl is 0 the currentInvestment will be also 0. That means that the caller is able to invest again which should avoided. It is recommended to check that the investSumN

#8	MyNFTS core	Withdraw/invest	See description	It is recommended to add a timelock of at least 24 hours (optional with a function that the owner can modify the timelock) for the withdrawing. Otherwise it could be possible that there are some addresses that are investing/withdrawing immediately.
#9	MyNFTS coreCycle	Verifier cannot be changed	32	It is recommended to check zero address while deploying the MyNFTScoreCycle because owner is not able to update the verifier address.
#10	MyNFTS core	Unemitted event	58	Emit or remove event

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:

- There is still an owner (Owner still has not renounced ownership)
- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
- Read whole report and modifiers section for more information

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED



SolidProof_io



@solidproof_io

*Solid
Proofed*

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


MADE IN GERMANY