



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Avion Finance Audit

Security Assessment
07. July, 2022

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	23
Source Units in Scope	27
Critical issues	28
High issues	28
Medium issues	28
Low issues	28
Informational issues	29
Alleviation	30
Audit Comments	30
SWC Attacks	31

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	07. July 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://www.avion.finance/>

Twitter

<https://twitter.com/AvionFinance>

Medium

<https://medium.com/@AvionFinance>

Discord

<https://discord.gg/UNQx77DBXk>



Description

Avion Finance is a DaaS and NFT protocol that comes from the amalgamation of hundreds of hours of work from the core team. We strive to create not only one of the best protocols in DeFi right now but also the most vibrant and welcoming communities. From our initial planning for the project, it became clear to us the problems in the DeFi and rebase token space right now, there is an abundant lack of sustainability, transparency, innovation, and equitability. We're an evolution of the rebase token and not your everyday Olympus or Titano fork. We aim to be the new defacto standard in the rebase token space and we look forward to having you aboard.

Project Engagement

During the 6th of July 2022, **AvionFinance 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

- Github
 - <https://github.com/avionfinance/avion>
 - Commit: 957bc549a7b3baca8a9f7cf35f863a2b428bb183

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
@chainlink/contracts/src/v0.8/interfaces/VRFCoordinatorV2Interface.sol	1
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721EnumerableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/utils/cryptography/MerkleProof.sol	1
hardhat/console.sol	1

Tested Contract Files

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

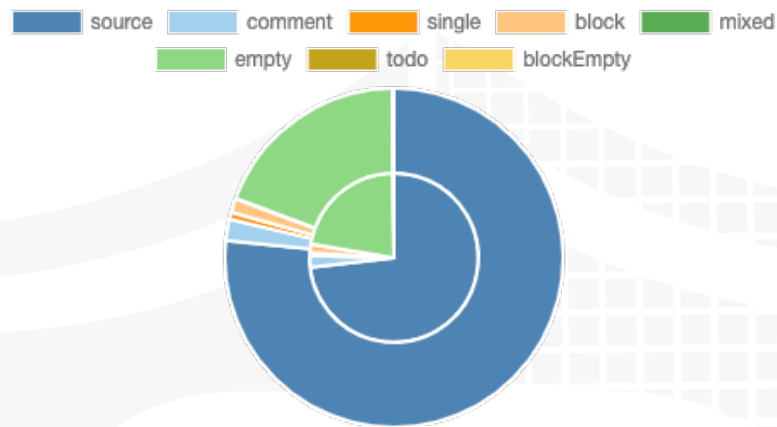
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

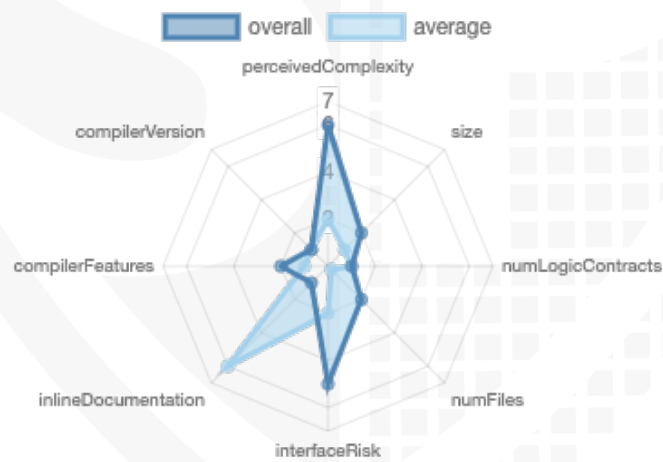
File Name	SHA-1 Hash
contracts/interfaces/IAvionToken.sol	d86c63a5e95f78afaecd0541516c9498f34a9d73
contracts/interfaces/IAvionCollection.sol	efe5489e34e67b5385a95f97dc21316cf4a528eb
contracts/interfaces/IPancakePair.sol	5991e45cf0e286e7b769e7b578a8e618d86d4b1a
contracts/interfaces/IPancakeRouter02.sol	fa9ccaf888732e2665476d63227c5f3512cd4769
contracts/interfaces/IPancakeRouter01.sol	e2178bdd770848643e277d895e0a6c16dfda2d95
contracts/interfaces/IPancakeFactory.sol	bea514529d2ac1a181b8e93b1396def433ad201f
contracts/interfaces/LinkTokenInterface.sol	890f80bb9b58bfdbf66e1858e7255cfc204b95ee
contracts/AvionERC721.sol	e5f4b12a40f52ffc4adb1c455273bda7b95bdd4f
contracts/AvionERC20.sol	2d05b55a14157b482b83e0e3e68eb280302d1459
contracts/FakeUSDC.sol	cbd1eeba1e8af3384dc6e7445a6a48538de7a309

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	0	7	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	137	6

Version	External	Internal	Private	Pure	View
1.0	116	110	4	10	46

State Variables

Version	Total	Public
1.0	75	50

Capabilities

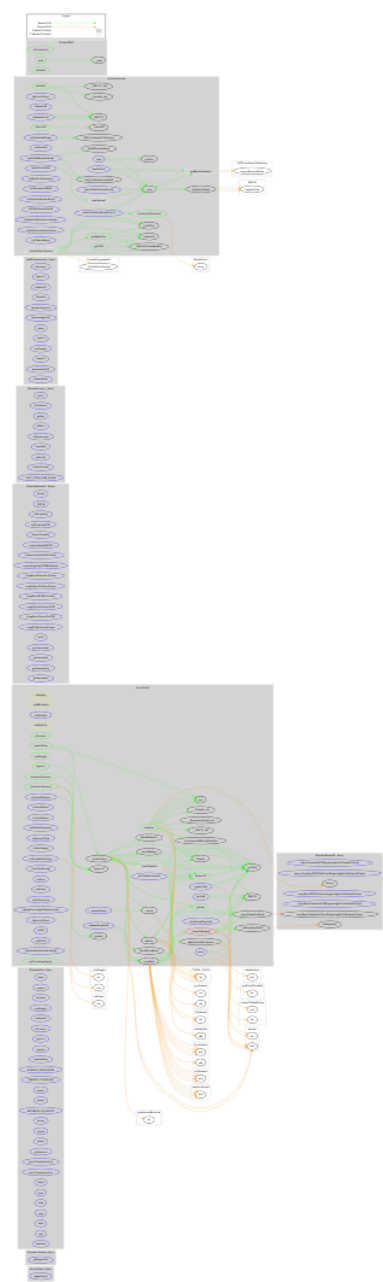
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.9 ≥0.5.0 ≥0.6.2 ^0.8.0		yes		

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. 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

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

AvionCollection

AvionToken

approve
changePercent...
clearStuckBala...
decreaseAllow...
getLink
IncreaseAllow...
initialize
manualRebase
manualSwapB...
mint
renounceOwn...
setAutomated...
setAutoRebase
setBlacklist
setBUSD
setFeeExempt
setFeeReceivers
setFees
setInitialDistri...
setNextRebase
setNft
setRebaseFreq...
setRewardYield
setRouter
setSwapBackS...
transfer
transferFrom
transferOwner...
updateYield

approve
initialize
mint
mintAtPublicP...
mintAtWhitel...
mintForTest
mintToDevWa...
rawFulfillRand...
renounceOwn...
safeTransferFr...
safeTransferFr...
setAllowanceT...
setApprovalFo...
setBaseURI
setChainLinkP...
setMintFeeDis...
setPublicPres...
setPublicPres...
setTokenAddr...
setWhitelist
setWhitelistPr...
setWhitelistPr...
transferFrom
transferOwner...
withdrawErc20

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗

Comments:

v1.0

- Owner can mint new tokens
- Remove “mintForTest” if it is not necessary anymore in L174, AvionERC721

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- Owner can lock user funds by
 - blacklisting addresses
 - initialDistributionFinished variable which could be set to false
- Tokens
 - will be burned while tx

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	—	—	—



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

- initialize
 - initializer
- setBlacklist
 - onlyOwner
- transfer
- transferFrom
 - validRecipient
- mint
 - onlyOwner
- decreaseAllowance
- increaseAllowance
- approve
- setNextRebase
 - onlyOwner
- updateYield
- setAutoRebase
 - onlyOwner
- manualRebase
 - nonReentrant
- setRebaseFrequency
 - onlyOwner
- setRewardYield
 - onlyOwner
- setFeeExempt
 - onlyOwner
- setSwapBackSettings
 - onlyOwner
- setFees
 - onlyOwner
- setRouter
 - onlyOwner
- setFeeReceivers
 - onlyOwner
- manualSwapBack
 - onlyOwner
- changePercentageForSaleAmo...
 - onlyOwner
- <Constructor> 💰
- setNft
 - onlyOwner
- setBUSD
 - onlyOwner
- setAutomatedMarketMakerPair
 - onlyOwner
- setInitialDistributionFinished
 - onlyOwner
- clearStuckBalance
 - onlyOwner
- getLink
 - onlyOwner

- initialize
 - initializer
- <Constructor> 💰
- setBaseURI
 - onlyOwner
- setChainLinkParam
 - onlyOwner
- rawFulfillRandomWords
- setWhitelist
 - onlyOwner
- mint
- mintForTest
 - onlyOwner
- mintAtPublicPresalePeriod
- mintAtWhitelistPresalePeriod
- mintToDevWallet
 - onlyOwner
- setMintFeeDistribution
 - onlyOwner
- setAllowanceToMint
 - onlyOwner
- setWhitelistPresalePeriod
 - onlyOwner
- setPublicPresalePeriod
 - onlyOwner
- setWhitelistPresalePeriodLimit
 - onlyOwner
- setPublicPresalePeriodLimit
 - onlyOwner
- setTokenAddress
 - onlyOwner
- withdrawErc20
 - onlyOwner

Note: Not listed functions was imported by libraries

Comments

- Deployer can set following state variables without any limitations
 - AvionERC20
 - percentageForLessThanSevenDays
 - Should not be 0 otherwise it causes that the `userInfo.amountAvailable` will be 0 as well
 - percentageForMoreThanSevenDays
 - Should not be 0 otherwise it causes that the `userInfo.amountAvailable` will be 0 as well
 - buyFees
 - sellFees
 - totalTransferFee
 - feeDenominator
 - Should not be 0 because you cannot divide by 0 (see L509)
 - gonSwapThreshold
 - rewardYields
 - rewardYieldDenominator
 - Should not be 0 because you cannot divide by 0 (see L380)
 - rebaseFrequency
 - nextRebase
 - AvionERC721
 - whitelistPresalePeriodLimit
 - publicPresalePeriodLimit
 - s_subscriptionId
 - numWords
 - requestConfirmations
 - callbackGasLimit
- Deployer can enable/disable following state variables
 - AvionERC20
 - blacklist
 - initialDistributionFinished
 - automatedMarketMakerPairs
 - swapEnabled
 - _isFeeExempt
 - autoRebase
 - AvionERC721
 - publicPresalePeriod
 - whitelistPresalePeriod
























- allowedToMint
- Deployer can set following addresses
 - AvionERC20
 - pair
 - busdToken
 - avionNFT
 - liquidityReceiver
 - treasuryReceiver
 - teamReceiver
 - burnReceiver
 - router
 - AvionERC721
 - token
 - whitelist
 - s_owner
 - keyHash
 - vrfCoordinator
 - BASE_URI
- Existing Modifiers
 - AvionERC20
 - swapping
 - validRecipient
 - noBlacklist
- AvionERC20
 - Function
 - getLink
 - We recommend you to rename the function because the name does not math for the functionality of function.
 - Also don't hardcode addresses into contracts
 - clearStuckBalance
 - Owner can get out contract balance
 - The burnReceiver can be a private Wallet. That means, that the “burned tokens” can be sold from that account also. We recommend you to set the burnReceiver als address(0) which gets the tokens when it is burned
- AvionERC721
 - Owner is able to withdraw erc20 tokens. If this function is used to withdraw foreign tokens, you should prevent to pass the address of avionErc20 contract

- AvionERC721
 - Owner is able to lock
 - “mintAtPublicPresalePeriod” function by setting publicPresalePeriodLimit to 0
 - “mintAtWhitelistPresalePeriod” function by setting whitelistPresalePeriodLimit to 0
 - We recommend you to check for zero value otherwise don't call the transferFrom function for the BUSD
 - L223
 - L226
 - Owner can mint for dev wallet, himself

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/interfaces/IAvionToken.sol	————	1	8	7	4	1	5	————
	contracts/interfaces/IAvionCollection.sol	————	1	8	7	4	1	5	————
	contracts/interfaces/IPancakePair.sol	————	1	52	7	5	————	55	————
	contracts/interfaces/IPancakeRouter02.sol	————	1	44	6	4	————	16	
	contracts/interfaces/IPancakeRouter01.sol	————	1	95	4	3	————	48	
	contracts/interfaces/IPancakeFactory.sol	————	1	19	6	4	————	19	————
	contracts/interfaces/LinkTokenInterface.sol	————	1	36	5	3	1	25	
	contracts/AvionERC721.sol	1	————	370	348	270	13	230	  
	contracts/AvionERC20.sol	1	————	799	731	546	13	410	 
	contracts/FakeUSDC.sol	1	————	16	16	11	1	9	————
	Totals	3	7	1447	1137	854	30	822	   

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	AvionER C721	Missing Zero Address Validation (missing- zero-check)	72 71 108 110 364	Check that the address is not zero
#2	AvionER C20	Missing Zero Address Validation (missing- zero-check)	760 124 107 747 739 539 536 538 537 735 111-114	Check that the address is not zero
#3	AvionER C20	State variable visibility is not set	28, 66	It is best practice to set the visibility of state variables explicitly

#4	AvionER C721	State variable visibility is not set	12, 14, 25, 30, 35, 40, 41, 42	It is best practice to set the visibility of state variables explicitly
#5	AvionER C721	Local variables shadowing	102	Rename the local variables that shadow another component. Recommendation: _ownerAddr
#6	AvionER C721	Missing Events Arithmetic	104 105 106 107 337 338	Emit an event for critical parameter changes
#7	AvionER C20	Missing Events Arithmetic	469, 473, 477, 480	Emit an event for critical parameter changes after iterating
#8	AvionER C721	Wrong order or condition	229	Change the order from the highest to the lowest because 6000 is also >5000 and it will move into the first if cases. That means that the other “else if” are not reachable.
#9	AvionER C20	Unusual function in no upgradeable contracts	See description	We recommend you to use constructor instead of initialize function when the contract is not an upgradeable. Everybody can call initialize function.

Informational issues

Issue	File	Type	Line	Description
#1	AvionER C721	Unused state variables	27, 17, 28	Remove unused state variables
#2	AvionER C20	Unused state variables	35	Remove unused state variables
#3	Main	Use functions instead of duplicating	403	You can use shouldRebase function in manualRebase function

#4	Pancake interfaces	SPDX Missing	See description	Provide a SPDX License to source code
----	--------------------	--------------	-----------------	---------------------------------------

Alleviation Medium Issue

Issue	File	Type	Line	Description
#1	Main	Owner can take out tokens from AvionErc20	See description	We recommend you to prevent passing AvionErc20 token to the withdraw function.

The team have to take out every ERC20 token. DYOR here.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/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.

07. July 2022:

- 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	NOT 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	NOT 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	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


MADE IN GERMANY