SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Staker Protocol

Audit

Security Assessment

31. May, 2022

For

STAKER PROTOCOL

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	27. May 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Harmony ONE

Website
https://stakerprotocol.com/

Telegram
https://t.me/StakerProtocolOfficial

Twitter
https://twitter.com/StakerProtocol

Description

STAKER PROTOCOL TOKEN can be utilized for its savings, while it can also be used for rewards and incentives with many community development partners, with each partner having its own rules on redemption. This makes the token unique and focuses on our core impact on community development as well as wealth building.

Project Engagement

During the 25th of May 2022, **Staker Protocol 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

- https://explorer.harmony.one/address/0xd8576e97de2e2d3a826b7e0e4507144b2a2b7f2b?activeTab=7
- https://explorer.harmony.one/address/0x79e244b545a0e0ad51e52016276728610f7ee368?activeTab=7
- https://explorer.harmony.one/address/0x9291e06137676a01667149472f43d5ba1d026aa8?activeTab=7
- https://explorer.harmony.one/address/0x89e23428d648e7583f7fe297665176e2d8818bed?activeTab=7

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:

Ownable
IToken
ISwap
Reservoir
📚 SafeMath

IAccessControl
Context
📚 Strings
IERC165
ERC165
AccessControl
ISwap
IToken
ITokenMint
IStakerVault
StakeLogic
📚 SafeMath

Ownable
Whitelist
📚 SafeMath
BEP20Basic
BasicToken
BEP20
StandardToken
MintableToken
Staker

Ownable
Whitelist
BEP20
📚 SafeMath
IToken
Swap

Tested Contract Files

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

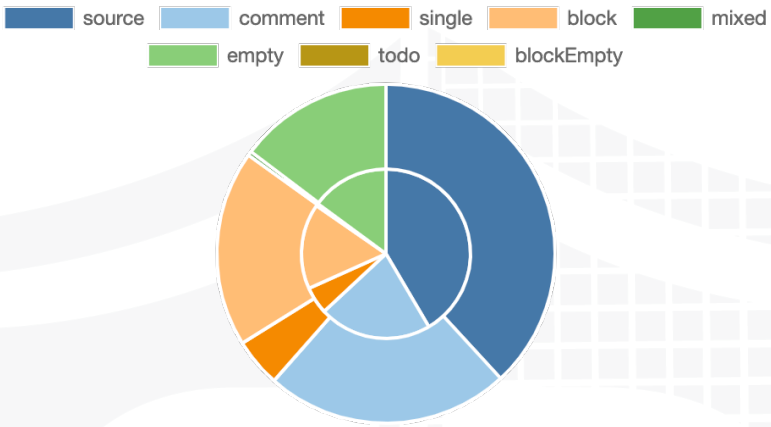
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

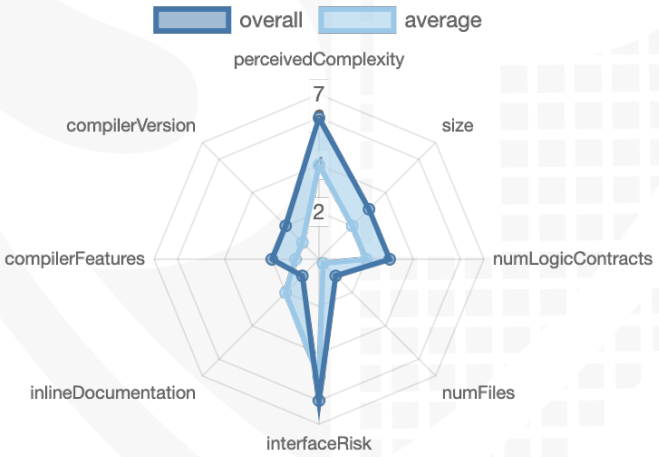
File Name	SHA-1 Hash
contracts/StakeLogic.sol	df9487a6d86ff06dbb3c0da361dd1b380c257b2d
contracts/Swap.sol	faca4e91302a3fc3912e0eae81a5c9b13bb84c74
contracts/Reservoir.sol	0786bac88f735fb61a20b42c3d6554db04260bf7
contracts/StakerToken.sol	a63849d3caa0e99ddb326d04a7a3fb1a94b16a38

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	14	5	10	3

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	228	13

Version	External	Internal	Private	Pure	View
1.0	81	201	7	28	99

State Variables

Version	Total	Public
1.0	106	84

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.4` `^0.6.12` `^0.4.25`		`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. Overall checkup (Smart Contract Security)

Write functions of contract v1.0

1. approve
2. removeAddressesFromWhitelist
3. removeAddressFromWhitelist
4. decreaseApproval
5. addAddressToWhitelist
6. increaseApproval
7. addAddressesToWhitelist
8. transferOwnership
9. setVaultAddress
10. setTaxDefault
11. mint
12. finishMinting
13. transferFrom
14. transfer
15. setAccountCustomTax
16. removeAccountCustomTax
17. excludeAccount
18. includeAccount

1. grantRole
2. renounceRole
3. revokeRole
4. updateRefBalances
5. updateTokenMint
6. updateOwner
7. updatePennyToken
8. updateStakerToken
9. updateStakerVault
10. updatePayoutRate
11. updateRefDepth
12. updateDepositRefBonus
13. updateClaimRefBonus
14. updateRefPayOutActivation
15. updateInitialDeposit
16. updateMinimumAmountDeposit
17. updateCompoundTax
18. updateExitTax
19. updateDepositBracketSize

20. updateShareFee
21. updateMaxPayoutCap
22. updateDeposit_bracket_max
23. updateHoldRequirements
24. checkin
25. deposit
26. claim
27. roll
28. batchAirdrop
29. airdrop

1. addAddressToWhitelist

2. addAddressesToWhitelist

3. approve

4. decreaseAllowance

5. increaseAllowance

6. removeAddressFromWhitelist

7. removeAddressesFromWhitelist

8. transfer

9. transferFrom

10. transferOwnership

11. unpause

12. pause

13. setToken

14. setMaxTokensPeriodicSell

15. setPeriodSellLimitation

16. setTrackingInterval

17. bnbToTokenSwapInput

18. bnbToTokenSwapOutput

19. tokenToBnbSwapInput

20. tokenToBnbSwapOutput

21. addLiquidity

22. removeLiquidity

1. transferOwnership

2. updateEntryFee

3. updateExitFee

4. updateStakerFee

5. updateInstantFee

6. updatePayoutRate

7. updateMagnitude

8. updateMinBuy

9. updateBalanceInterval

10. updateDistributionInterval

11. updateSwapAddress

12. updateCollateralAddress

13. buy

14. reinvest

15. withdraw

16. sell

17. sweep

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

- updateEntryFee
  - onlyOwner
- updateExitFee
  - onlyOwner
- updateStakerFee
  - onlyOwner
- updateInstantFee
  - onlyOwner
- updatePayoutRate
  - onlyOwner
- updateMagnitude
  - onlyOwner
- updateMinBuy
  - onlyOwner
- updateBalanceInterval
  - onlyOwner
- updateDistributionInterval
  - onlyOwner
- updateSwapAddress
  - onlyOwner
- updateCollateralAddress
- buy 💰
- <Fallback> 💰
- reinvest
  - onlyStronghands
- withdraw
  - onlyStronghands
- sell
  - onlyStronghands
- sweep

- transferOwnership
  - onlyOwner

- updateRefBalances
  - OnlyGovernance
- updateTokenMint
  - OnlyGovernance
- updateOwner
  - OnlyGovernance
- updatePennyToken
  - OnlyGovernance
- updateStakerToken
  - OnlyGovernance
- updateStakerVault
  - OnlyGovernance
- updatePayoutRate
  - OnlyGovernance
- updateRefDepth
  - OnlyGovernance
- updateDepositRefBonus
  - OnlyGovernance
- updateClaimRefBonus
  - OnlyGovernance
- updateRefPayOutActivation
  - OnlyGovernance
- updateInitialDeposit
  - OnlyGovernance
- updateMinimumAmountDeposit
  - OnlyGovernance
- updateCompoundTax
  - OnlyGovernance
- updateExitTax
  - OnlyGovernance
- updateDepositBracketSize
  - OnlyGovernance
- updateShareFee
  - OnlyGovernance
- updateMaxPayoutCap
  - OnlyGovernance
- updateDeposit_bracket_max
  - OnlyGovernance
- updateHoldRequirements
  - OnlyGovernance
- checkin
- deposit
- claim
- roll
- batchAirdrop
- airdrop

- grantRole
  - onlyRole
- revokeRole
  - onlyRole
- renounceRole

- setVaultAddress
  - onlyOwner
- setTaxDefault
  - onlyOwner
- mint
- finishMinting
  - onlyOwner
  - canMint
- transferFrom
- transfer
- setAccountCustomTax
  - onlyOwner
- removeAccountCustomTax
  - onlyOwner
- excludeAccount
  - onlyOwner
- includeAccount
  - onlyOwner
- mint
  - onlyWhitelisted
  - canMint
- finishMinting
  - onlyWhitelisted
  - canMint
- transferFrom
- approve
- increaseApproval
- decreaseApproval

```
addAddressToWhitelist
    onlyOwner
addAddressesToWhitelist
    onlyOwner
removeAddressFromWhitelist
    onlyOwner
removeAddressesFromWhitelist
    onlyOwner
```

```
unpause
    onlyOwner
pause
    onlyOwner
setToken
    onlyOwner
setMaxTokensPeriodicSell
    onlyOwner
setPeriodSellLimitation
    onlyOwner
setTrackingInterval
    onlyOwner
bnbToTokenSwapInput 💰
    isNotPaused
bnbToTokenSwapOutput 💰
    isNotPaused
tokenToBnbSwapInput
    isNotPaused
tokenToBnbSwapOutput
    isNotPaused
addLiquidity 💰
    isNotPaused
removeLiquidity
    onlyWhitelisted
```

Comments

- *Deployer can set following state variables without any limitations*
  - Max 2^8-1
    - entryFee_
    - exitFee_
    - stakerFee
    - instantFee
    - payoutRate_
    - taxDefault
  - magnitude
  - minBuy
  - balanceInterval
  - distributionInterval
  - ref_balances
  - payoutRate
  - ref_depth
  - deposit_ref_bonus
  - claim_ref_bonus
  - minimumInitial
  - minimumAmount
  - CompoundTax
  - ExitTax
  - deposit_bracket_size
  - shareFee
  - max_payout_cap
  - deposit_bracket_max
  - ref_balances
  - maxTokensPeriodicSell
  - periodSellLimitation
  - trackingInterval_

- *Deployer can enable/disable following state variables*
  - refPayOutIsActive
  - _excluded
  - _isExcluded
  - _hasCustomTax
  - _hasCustomTax
    - If it is set manually you have to set it every time for that address, so it will not use the taxDefault anymore
  - isPaused

- *Deployer can set following addresses*
  - swapToken
  - swap

    - swapAddress
    - cToken
    - collateralAddress
    - tokenMint
    - owner
    - pennyToken
    - stakerToken
    - stakerVaultAddress
    - stakerVault
    - vaultAddress
    - token

- *Existing Modifiers*
    - onlyOwner
    - onlyBagholders
    - onlyStronghands
    - onlyRole
    - OnlyGovernance
    - onlyWhitelisted
    - canMint
    - isNotPaused

- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
    - Be aware of this

- Every whitelisted address is able to mint tokens
- Owner can pause swap contract

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/StakeLogic.sol	6	6	1509	1269	670	497	515	💰📩🧮🔆
📝📚🔍	contracts/Swap.sol	5	1	812	801	411	267	384	💰📩
📝📚🔍	contracts/Reservoir.sol	3	2	896	768	441	231	335	💰📩
📝📚🔍	contracts/StakerToken.sol	8	1	588	560	298	178	246	📩🔆
📝📚🔍🎨	**Totals**	**22**	**10**	**3805**	**3398**	**1820**	**1173**	**1480**	💰📩🧮🔆

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

Issue	File	Type	Line	Description
#1	All except StakeLogic	Raw mathematical operations	See description	Use SafeMath library instead of raw mathematical operations because of the overflow/underflow issue. It is mixed up in the contracts with raw and safemath operations. Since pragma version 0.8.x safemath is implemented by default.

Low issues

Issue	File	Type	Line	Description
#1	All	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	All	A floating pragma is set	At the top of source code	The current pragma Solidity directive is...(look at the top with the "^" sign)

#3	Reservoir	Missing Zero Address Validation (missing-zero-check)	346, 409, 403	Check that the address is not zero
#4	StakeLogic	Missing Zero Address Validation (missing-zero-check)	793, 845, 857	Check that the address is not zero
#5	StakerToken	Missing Zero Address Validation (missing-zero-check)	404	Check that the address is not zero
#6	Reservoir	State variable visibility is not set	307	It is best practice to set the visibility of state variables explicitly
#7	Swap	Local variables shadowing	728	Rename the local variables that shadow another component
#8	Reservoir	Missing Events Arithmetic	396 400 368 372 380 388 384 376	Emit an event for critical parameter changes
#9	StakerToken	Missing Events Arithmetic	409	Emit an event for critical parameter changes
#10	Swap	Missing Events Arithmetic	456	Emit an event for critical parameter changes
#11	StakeLogic	Uninitialized state variable	760	State variable is not initialized but it is used in the code. The value will never change.
#12	StakeLogic	Uninitialized local variable	953	Local variable is not initialized
#13	StakeLogic	Tautology or contradiction	558	Fix the incorrect comparison by changing the value type or the comparison.

Informational issues

Issue	File	Type	Line	Description

#1	StakeLogic	State variables that could be declared constant (constable-states)	741	Add the `constant` attributes to state variables that never change
#2	StakeLogic	Functions that are not used	1153	Remove unused functions
#3	Main	Misspelling	See description	Change following words: - Make sure to change it everywhere else as well.
#4	All	Error message is missing	See description	Provide an error message for require statement Look for "require(" and add an error message everywhere where a message is missing
#5	All	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#6	Reservoir	Naming convention	Look into the file and compare visibility with variable name	Usually private/internal variables starts with an underscore. If you are going to change it, make sure to change it everywhere else also
#7	StakeLogic	Naming convention	Look into the file and compare visibility with variable name	Usually external/public variables starts with a lower case letter. If you are going to change it, make sure to change it everywhere else also
#8	Reservoir	Visibility order	546	Visibility modifier "public/ external/etc." first and then the others.
#9	StakerToken	Visibility order	64, 78, 92, 106, 347, 360, 451,	Visibility modifier "public/ external/etc." first and then the others.
#10	StakerToken	Function can be pure	536, 455	Function can be restricted to pure

#11	Swap	Function can be pure	485, 500	Function can be restricted to pure
#12	Swap	Unused local variable	599, 563	Remove unused local variable
#13	StakeLogic	Unused local variable	938, 1202, 1295, 1392, 1410,	Remove unused local variable

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.

27. May 2022:

- We recommend to
  - update the contracts at least to 0.8.x pragma version
  - Use decentralised oracles for calculating prices
- 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	NOT 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	NOT 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	NOT 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