SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
20. October, 2021

For

HyperBoost

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	20. October 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://hyperboostworld.com/

Telegram
https://t.me/HyperBoostOfficial

Twitter
https://twitter.com/HyperBoostBSC

Facebook
https://www.facebook.com/HyperBoostBSC/

Discord
https://discord.gg/tZnpcavR9H

Reddit
https://www.reddit.com/r/HyperBoost_World/

Description

HyperBoost seeks to touch the world by showing off his tricks, including its one of a kind tokenomics and a true utility use case project like no other project has done before.

Project Engagement

During the 18th of October 2021, **HyperBoost 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://bscscan.com/address/
0xC1F5cF212e8d2774A9756a9a38f05dEC1ED4afcA#contracts

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

<u>Auditing Strategy and Techniques Applied</u>

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:
- OpenZeppelin
  - Context
  - IERC20
- Uniswap
  - UniswapV2Factory
  - UniswapV2Pair
  - UniswapV2Router01
  - UniswapV2Router02

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.
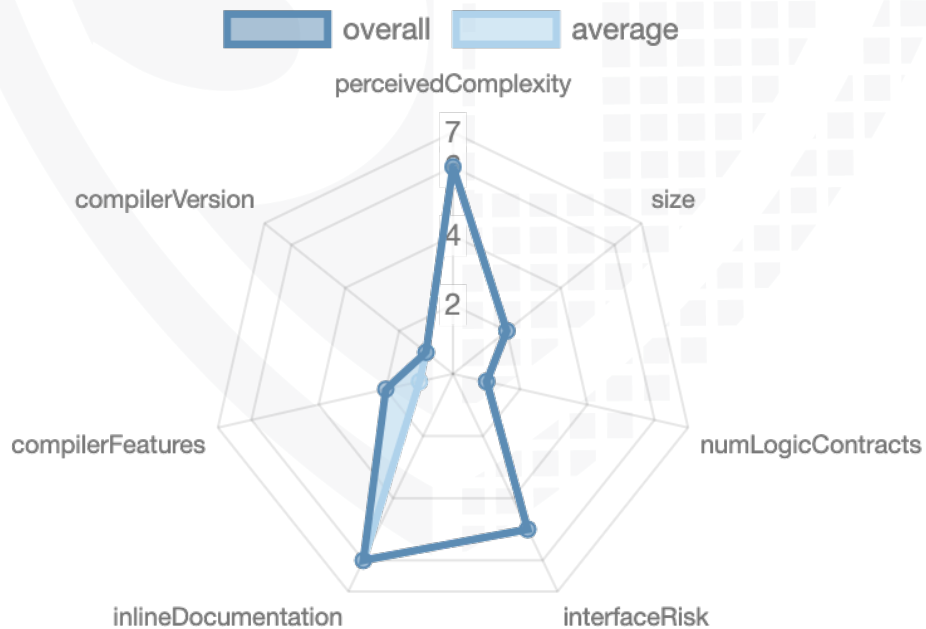
v1.0

File Name	SHA-1 Hash
contracts/hyperboost.sol	4a1a25b0d04f352f10b8ca27f9bc30865d27cd76

Metrics

Source Lines
v1.0
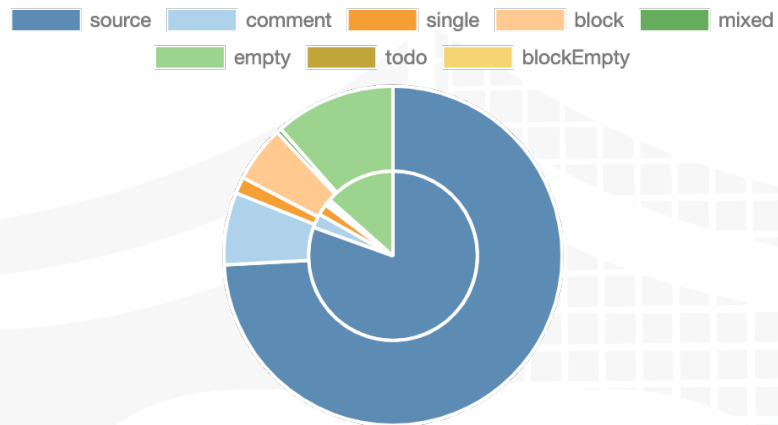
Risk Level
v1.0

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	0	5	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	114	6

Version	External	Internal	Private	Pure	View
1.0	96	80	10	10	43

State Variables

Version	Total	Public
1.0	105	50

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`>=0.6.0 <0.9.0`		yes	`****` (0 asm blocks)	

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

Scope of Work

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. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph
v1.0
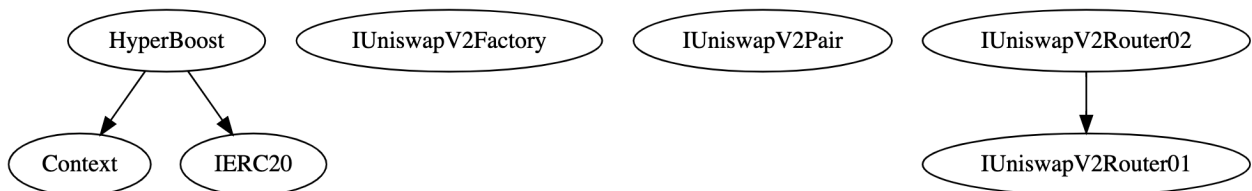
Verify Claims
Correct implementation of Token standard

Tested	Verified
✓	✓

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	✓	✓	✓

Optional implementations

Function	Description	Exist	Tested	Verified
renounceOwnership	Owner renounce ownership for more trust	✓	✓	✗

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified	File
Deployer cannot mint	✓	✓	✓	Main
Comment	Line: -			

Max / Total Supply: 1.000.000.000.000.000

Deployer cannot burn or lock user funds

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

Comments:
v1.0

- Deployer can lock
    - When gasLimitActive is true the tx.gasprice should be lower/ equal to gasPriceLimit

```
if (gasLimitActive) {
    require(tx.gasprice <= gasPriceLimit, "Gas price exceeds limit.");
}
```

    - If _hasLimits is true

```
function _hasLimits(address from, address to) private view returns (bool) {
    return from != owner()
        && to != owner()
        && !_liquidityHolders[to]
        && !_liquidityHolders[from]
        && to != DEAD
        && to != address(0)
        && from != address(this);
}
```

    - And trading is not enabled
        - Transaction will be reverted

```
if (sameBlockActive) {
    if (lpPairs[from]){
        require(lastTrade[to] != block.number);
        lastTrade[to] = block.number;
    } else {
        require(lastTrade[from] != block.number);
        lastTrade[from] = block.number;
    }
}
```

- If sameblockActive is true
  - lastTrade from to/from should not be the same as block.number

```
require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
```

- If lpPairs[to] is true and sellLimitsEnabled

```
if(sellLimitsEnabled) {
    if (block.timestamp - sellValues[from].userSellTime >= maxSellTimeLimit) {
        sellValues[from].userSoldAmt = amount;
        sellValues[from].userSoldAmtLimit = (balanceOf(from) * maxSellAmtLimit) / maxSellAmtLimitDivisor;
        sellValues[from].userSellTime = block.timestamp;
    } else {
        require(sellValues[from].userSoldAmt + amount <= sellValues[from].userSoldAmtLimit, "Selling too much within the time limit!");
        sellValues[from].userSoldAmt += amount;
    }
}
```

- block.timestamp - timestamp of userSellTime must be higher/equal to maxSellTimeLimit otherwise its required that the userSoldAmt from the seder must be lower/equal to userSoldAmtLimit

```
if(to != _routerAddress && !lpPairs[to]) {
    require(balanceOf(to) + amount <= _maxWalletSize, "Transfer amount exceeds the maxWalletSize.");
}
```

- If balance of recipient plus the amount that sent is higher than _maxWalletSize transaction will be reverted
- If sniperProtection is true

```
if (isSniper(from) || isSniper(to)) {
    revert("Sniper rejected.");
}
```

- And the sender or recipient is sniper
  - Transaction will be reverted

```
_checkLiquidityAdd(from, to);
if (!_hasLiqBeenAdded && _hasLimits(from, to)) {
    revert("Only owner can transfer at this time.");
}
```

- If hasLimits and _liqAddStatus is 0 or _liqAddStatus is not startingSupply/20
  - Transaction will be reverted
-

```
if (_hasLimits(from↑, to↑)){
    if (_liqAddStatus == 0 || _liqAddStatus != startingSupply / 20) {
        revert();
    }
}
```

```
if(lpPairs[from↑] && block.timestamp <= _liqAddStamp + 5 minutes) {
    revert("Buying not enabled yet!");
}
```

1. approve	→
2. approveMax	→
3. decreaseAllowance	→
4. enableTrading	→
5. excludePresaleAddresses	→
6. increaseAllowance	→
7. isProtected	→
8. removeSniper	→
9. renounceOwnership	→
10. setBuybackEnabled	→
11. setBuybackSettings	→
12. setExcludedFromFees	→
13. setExcludedFromReward	→
14. setGasPriceLimit	→
15. setLpPair	→
16. setMaxTxPercent	→
17. setMaxWalletSize	→
18. setNewRouter	→
19. setProtectionSettings	→
20. setRatios	→
21. setSellLimits	→
22. setSellLimitsEnabled	→
23. setStartingProtections	→
24. setSwapAndLiquifyEnabled	→
25. setSwapSettings	→
26. setTaxesBuy	→
27. setTaxesSell	→
28. setTaxesTransfer	→
29. setWallets	→
30. transfer	→
31. transferFrom	→
32. transferOwner	→

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

1. approve →
2. approveMax →
3. decreaseAllowance →
4. enableTrading →
5. excludePresaleAddresses →
6. increaseAllowance →
7. isProtected →
8. removeSniper →
9. renounceOwnership →
10. setBuybackEnabled →
11. setBuybackSettings →
12. setExcludedFromFees →
13. setExcludedFromReward →
14. setGasPriceLimit →
15. setLpPair →
16. setMaxTxPercent →
17. setMaxWalletSize →
18. setNewRouter →
19. setProtectionSettings →
20. setRatios →
21. setSellLimits →
22. setSellLimitsEnabled →
23. setStartingProtections →
24. setSwapAndLiquifyEnabled →
25. setSwapSettings →
26. setTaxesBuy →
27. setTaxesSell →
28. setTaxesTransfer →
29. setWallets →
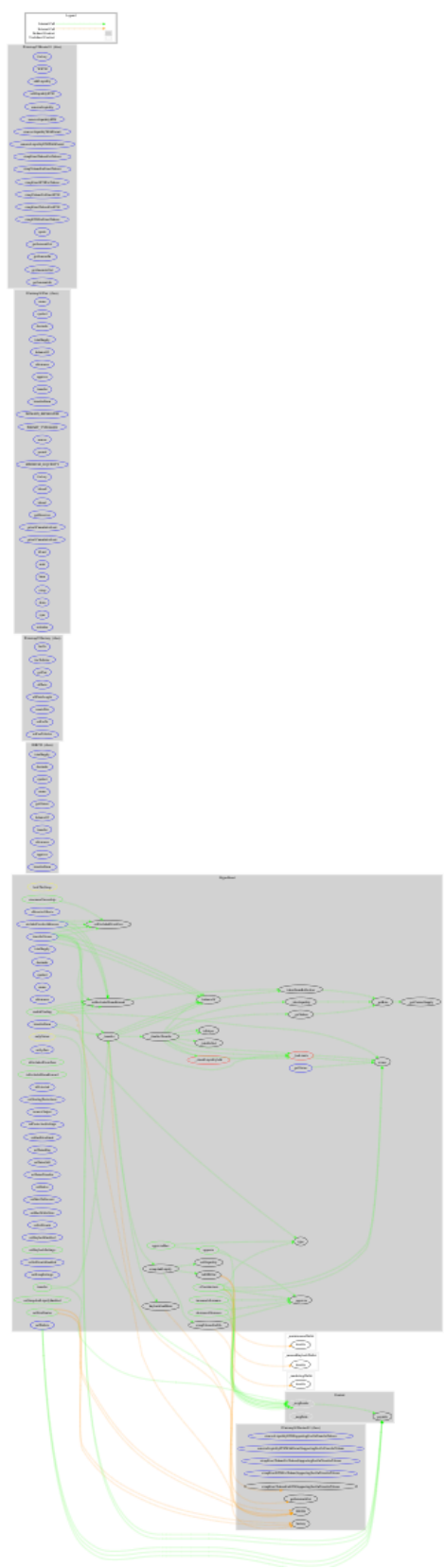30. transfer →
31. transferFrom →
32. transferOwner →

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

CallGraph

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍🐞	contracts/hyperboost.sol	2	5	1172	919	770	88	678	💰📥
📝🔍🐞	**Totals**	**2**	**5**	**1172**	**919**	**770**	**88**	**678**	💰📥

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 found -

High issues
- no high issues found -

Medium issues
- no medium issues found -

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	Main	A floating pragma is set	6	The current pragma Solidity directive is „">=0.6.0 <0.9.0 "".
#3	Main	Missing Zero Address Validation (missing-zero-check)	806	Check that the address is not zero
#4	Main	State variable visibility is not set	306, 388, 427	It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description
#1	Main	State variables that could be declared constant (constable-states)	382, 383, 370, 323, 380, 324, 367, 392, 391, 398, 397, 321	Add the `constant` attributes to state variables that never change

#2	Main	Unused state variables (unused-state)	394, 400	Remove unused state variables

Audit Comments
20. October 2021:

- There is still an owner (Owner still has not renounced ownership)
- Look at Deployer cannot burn or lock user funds section

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	**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