SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
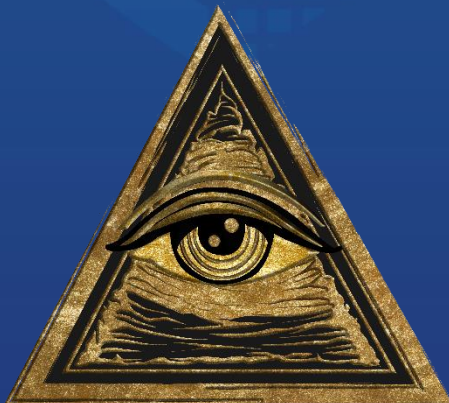
MADE IN GERMANY

Illuminati Finance

Audit

Security Assessment
18.July,2022

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	15.July,2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Binance (BSC)

Website

https://illuminati.financial/

Twitter

https://twitter.com/illuminatifinan

Telegram

https://t.me/illuminatifinan

Discord

https://discord.com/invite/9rwAyVmdmW

Medium

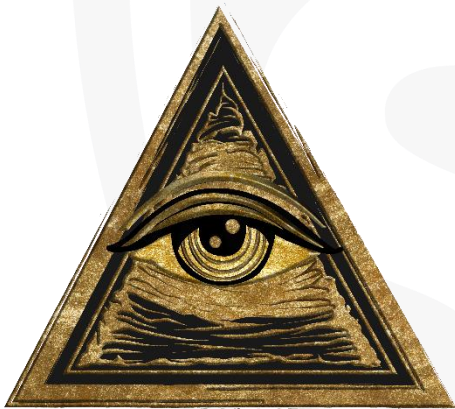https://medium.com/@illuminati_financial

Description

Illuminati provides a decentralized financial asset which rewards users with a sustainable fixed compound interest model through use of its unique Unlimited NFT Farming and Illuminati Auto-Staking Protocol (IAP for short). The IAP is a new financial protocol that makes staking easier, more efficient and awards $ILLU token holders the highest stable returns in crypto.

Project Engagement

During the 15th of July 2022, **Illuminati Finance** team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Links

v1.0
https://bscscan.com/address/0xb9E4FD33591ebAE6764694B541E72A2a803cE8F4#code

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.
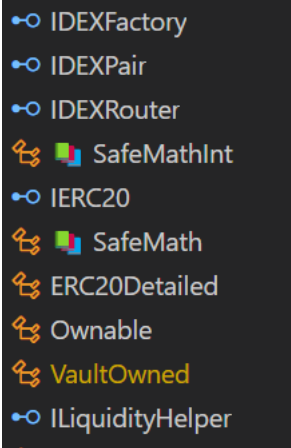
Methodology

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 analyzing 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:

```
•o IDEXFactory
•o IDEXPair
•o IDEXRouter
   SafeMathInt
•o IERC20
   SafeMath
ERC20Detailed
Ownable
VaultOwned
•o ILiquidityHelper
```

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/IlluminatiToken.sol	cc59b200a0e0a3abc16314d21f3780348d0995c2

Metrics

Source Lines
v1.0
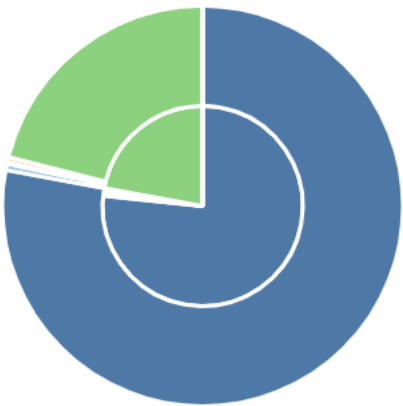
Risk Level
v1.0
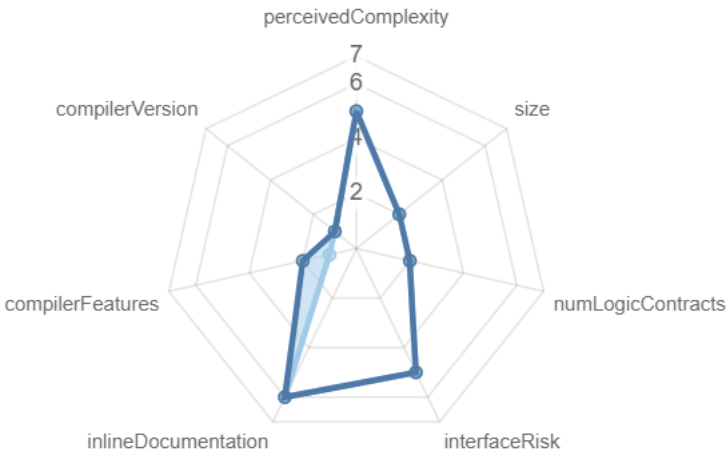
Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	2	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	62	3

Version	External	Internal	Private	Pure	View
1.0	46	66	3	15	24

State Variables

Version	Total	Public
1.0	45	20

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.7.5`		Yes		

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

Inheritance Graph
v1.0
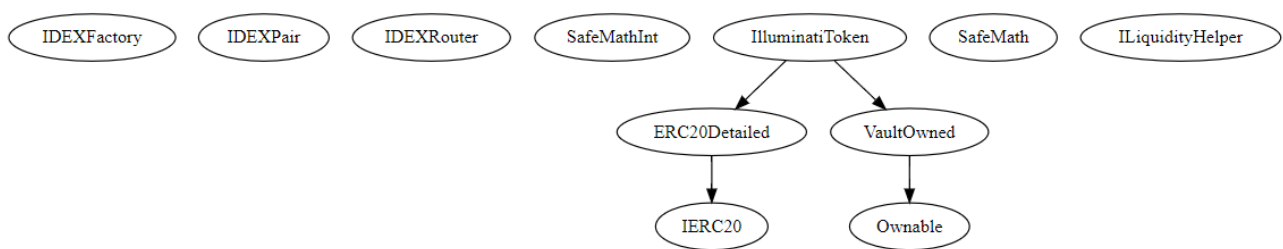
Call Graph
v1.0
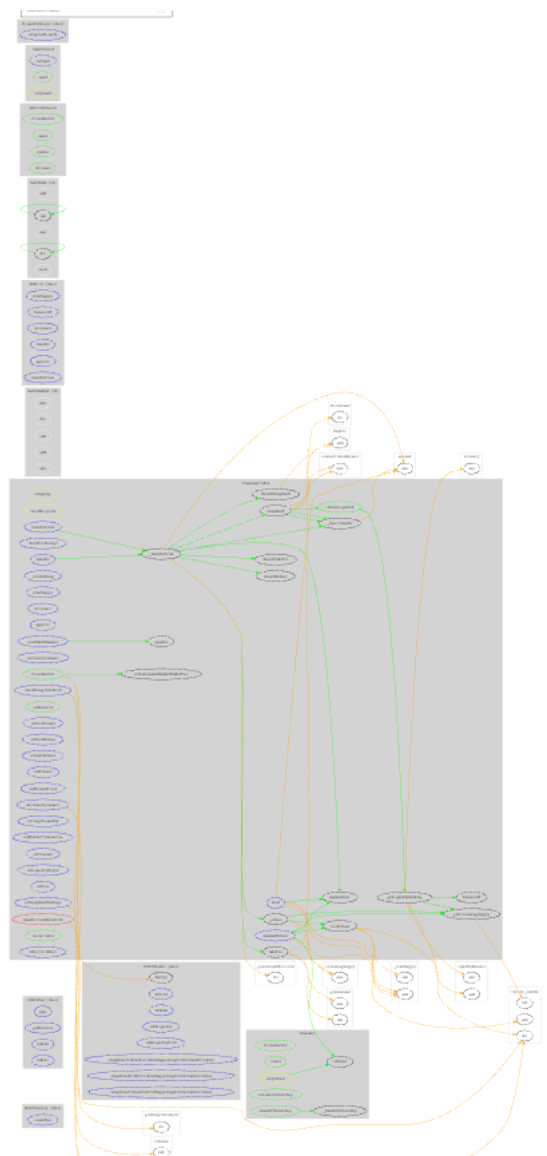
Scope of Work/Verify Claims

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)
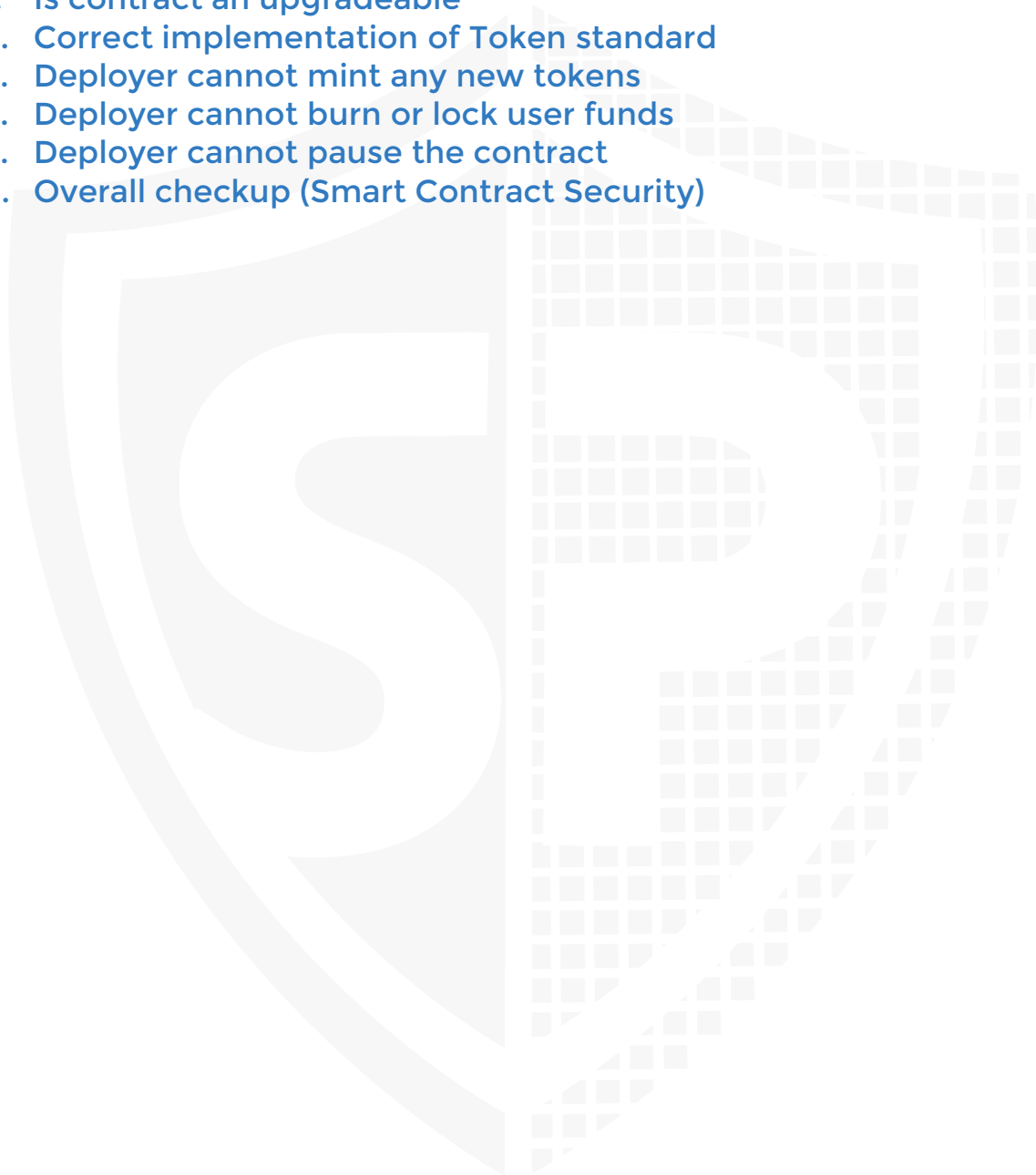
Is contract an upgradeable

Name	
Is contract an upgradeable?	**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			

Write functions of contracts

v1.0

1. approve

2. clearStuckBalance

3. decreaseAllowance

4. increaseAllowance

5. manualRebase

6. manualSync

7. mint

8. renounceOwnership

9. rescueToken

10. setAutoRebase

11. setAutomatedMarketMakerPair

12. setBlacklist

13. setFeeExempt

14. setFees

15. setLiquidityHelper

16. setMaxSellTransaction

17. setNextRebase

18. setRebaser

19. setRewardYield

20. setSwapBackSettings

21. setTargetLiquidity

22. setTreasury

23. setVault

24. transfer

25. transferFrom

26. transferOwnership

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint			
Max / Total Supply	10.000.000		

Comments:

Although, the deployer can't mint any new tokens more than the maximum supply which is set as constant but the deployer will still have the rights to mint the tokens till the maximum supply is reached. Hence, they can increase the "total supply"

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock			
Deployer cannot burn			

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause			

Comments:

The deployer can halt the selling of tokens to the AMM indirectly if they want

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

⬥ \<Constructor\> 💰
⬥ approve
⬥ decreaseAllowance
⬥ increaseAllowance
⬥ mint
Ⓜ onlyVault
⬥ transfer
Ⓜ validRecipient
⬥ transferFrom
Ⓜ validRecipient
⬥ setBlacklist
Ⓜ onlyOwner
⬥ setFeeExempt
Ⓜ onlyOwner
⬥ setNextRebase
Ⓜ onlyOwner
⬥ setAutoRebase
Ⓜ onlyOwner
⬥ setRebaser

⬥ setRewardYield
Ⓜ onlyOwner
⬥ setSwapBackSettings
Ⓜ onlyOwner
⬥ setTargetLiquidity
Ⓜ onlyOwner
⬥ setMaxSellTransaction
Ⓜ onlyOwner
⬥ setTreasury
Ⓜ onlyOwner
⬥ setLiquidityHelper
Ⓜ onlyOwner
⬥ setFees
Ⓜ onlyOwner
⬥ setAutomatedMarketMakerPair
Ⓜ onlyOwner
⬥ manualRebase
⬥ manualSync

⬥ clearStuckBalance
Ⓜ onlyOwner
⬥ rescueToken
Ⓜ onlyOwner

Comments:
- Some of the functions has a modifier "onlyOwner" which allows an authorities to do certain operations whenever they want. Some of the operations they can do are listed below:
    - The owner can set the maximum transaction amount to zero and stop everyone from selling of tokens to the AMM
    - The owner can transfer the contract's balance to any arbitrary wallet of their choosing.
    - The owner can blacklist/whitelist any wallet address
    - The owner can include or exclude anyone from the fee.

Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/IlluminatiToken.sol	6	5	765	647	499	4	432
Totals	**6**	**5**	**765**	**647**	**499**	**4**	**432**

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

<div style="background-color: #7FFF3C; text-align: center;">

AUDIT PASSED

</div>

Critical issues

<div style="background-color: #7FFF3C; text-align: center; color: green;">

No critical issues

</div>

High issues

<div style="background-color: #7FFF3C; text-align: center; color: green;">

No high issues

</div>

Medium issues

Issue	File	Type	Line	Description
#1	Main	Own contract tokens are transferrable	756	The owner address can transfer the whole balance of the contract to any address because there is no prevention against it. We recommend to prevent passing own contract address in the function.

Low issues

Issue	File	Type	Line	Description
#1	Main	A floating pragma is set	6	The current pragma Solidity directive is „"^0.7.5"".
#2	Main	Missing Events	540,544,549,5535 58,563,568,573,5 78,582,588,594	Emit an event for critical parameter changes. In this case, minting, burning of tokens, etc.
#3	Main	Missing zero check	280,751,616,751	Check that the address is not zero
#4	Main	Halting Transactions	578	The owner can set the maximum transaction amount to zero and stop everyone from selling of

				tokens to the AMM because there is no prevention against it.
#5	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

Informational issues

Issue	File	Type	Line	Description
#1	Main	Dead Code	747	This function is never used in the contract and should be removed.
#2	Main	Unused return values	402	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	Main	NatSpec documentation missing	–	If you started to comment your code, also comment all other functions, variables etc.
#4	Main	Use of old compiler version	6	Using old compiler versions for deployment is not recommended as they contain known security bugs.

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.

18.July,2022:

• There is still an owner (Owner still has not renounced ownership)

- Owner can Enable/Disable the swapping functionality
- Owner can transfer the contract's balance to some other account.
- Read the 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	**NOT 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-	Presence of unused variables	CWE-1164: Irrelevant Code	**NOT PASSED**

[1 3 1](#)			
[S W C - 1 3 0](#)	Right-To-Left-Override control character (U+202E)	[CWE-451: User Interface (UI) Misrepresentation of Critical Information](#)	**PASSED**
[S W C - 1 2 9](#)	Typographical Error	[CWE-480: Use of Incorrect Operator](#)	**PASSED**
[S W C - 1 2 8](#)	DoS With Block Gas Limit	[CWE-400: Uncontrolled Resource Consumption](#)	**PASSED**
[S W C - 1 2 7](#)	Arbitrary Jump with Function Type Variable	[CWE-695: Use of Low-Level Functionality](#)	**PASSED**
[S W C - 1 2 5](#)	Incorrect Inheritance Order	[CWE-696: Incorrect Behavior Order](#)	**PASSED**
[S W C -](#)	Write to Arbitrary	[CWE-123: Write-what-where Condition](#)	**PASSED**

[124](#)	Storage Location		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	**NOT PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**NOT 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**

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY