SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

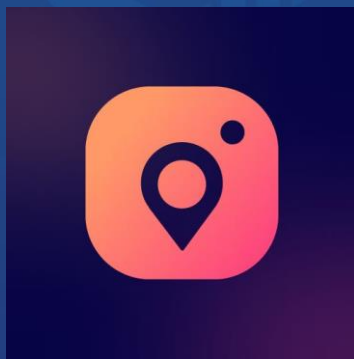MADE IN GERMANY

Global Guide

Audit

Security Assessment
31.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	29.July,2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Binance (BSC)

Website
https://shelbyswap.io/

Twitter
https://twitter.com/globalguideapp

Telegram
https://t.me/globalguideofficial

Discord
https://discord.gg/6ZZdUaHw7U

Instagram
https://www.instagram.com/globalguideapp/

Description

Global-Guide is the first web 3.0 Dapp that implements the Socialize-To-Earn concept in crypto.
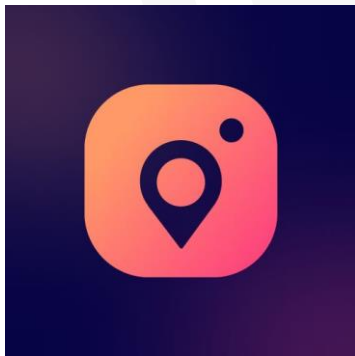
Global Guide is aimed to become a global community of explorers who write reviews, share photos, share their locations, add or edit places on our Web 3.0 platform. We refer these activities as socializing.

The concept behind Global-Guide is one that is highly scalable and promising and also incredibly approachable to everyone. Users of Global-Guide can socialize to earn $GGT tokens which can either be used in-game, or cashed out for profit. Meanwhile millions of people such as tourists can rely on their contributions to decide where to go and what to do. In order to socialize, users must own a camera NFT which is specifically designed to use for different scenes such as humans, foods, natures, events, architectures or seasons.

Project Engagement

During the 29th of July 2022, **Global Guide** 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/0xf8715373504eC3007E5A8E4Fd18b8278ae323c61#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.
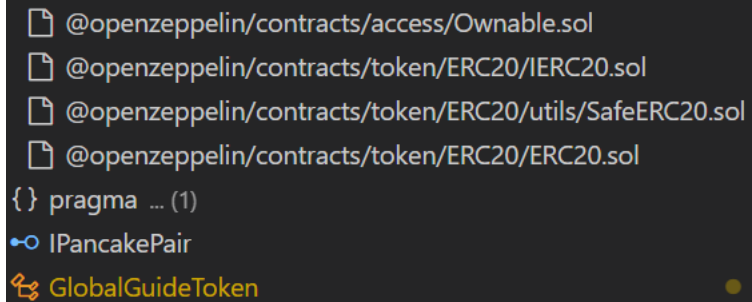
Methodology

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:


```
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol
{ } pragma ... (1)
IPancakePair
GlobalGuideToken
```

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/GlobalGuideToken.sol	a79e7c63caa273d37022851d69bf9710eafc7835

Metrics

Source Lines
v1.0

Risk Level
v1.0
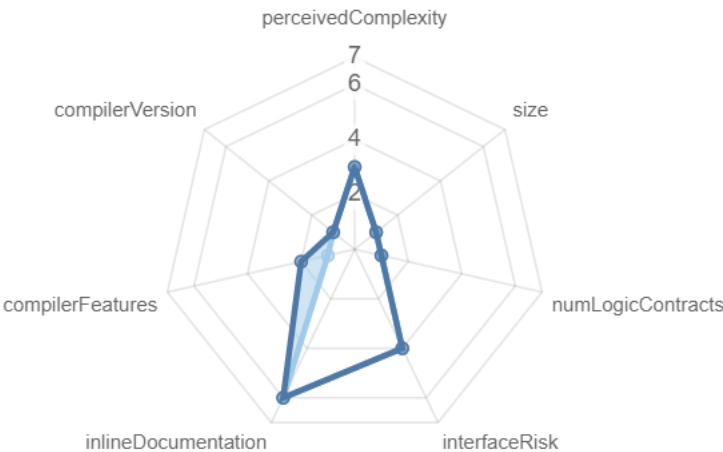
Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	0	1	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	34	0

Version	External	Internal	Private	Pure	View
1.0	32	21	0	5	13

State Variables

Version	Total	Public
1.0	8	4

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.0`			Yes	

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

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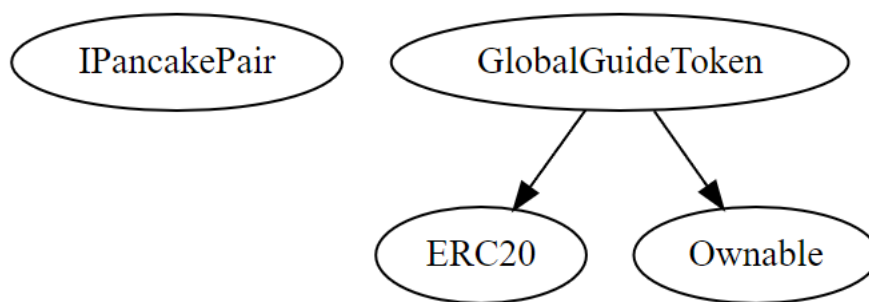
Call Graph
v1.0
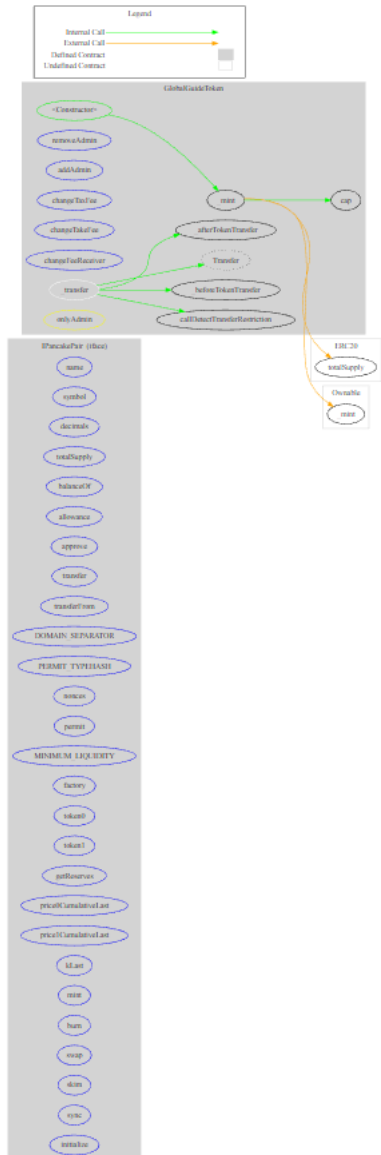
Scope of Work/Verify Claims

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)
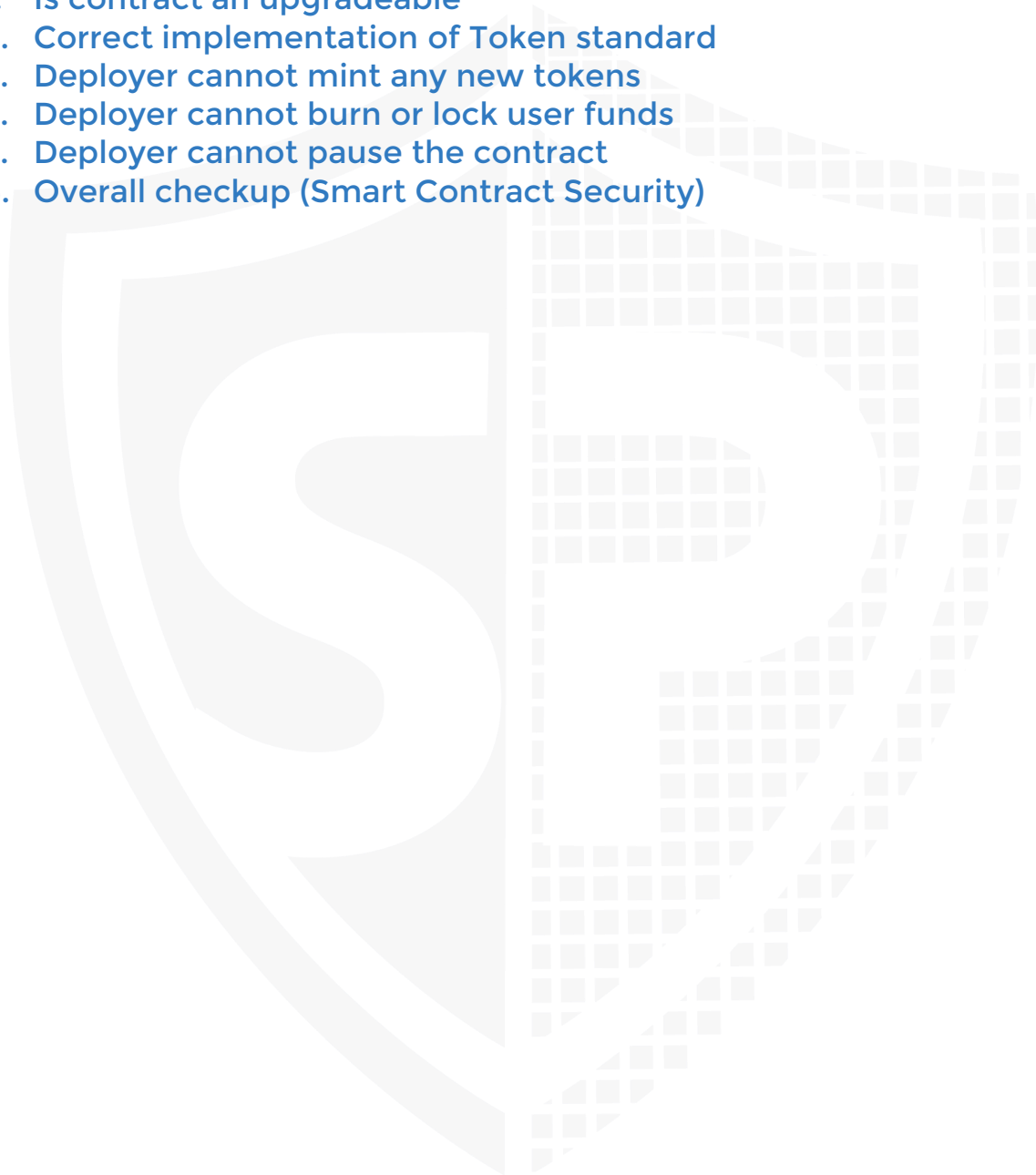
Is contract an upgradeable

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. addAdmin

2. approve

3. callDetectTransferRestriction

4. changeFeeReceiver

5. changeTakeFee

6. changeTaxFee

7. decreaseAllowance

8. increaseAllowance

9. removeAdmin

10. renounceOwnership

11. transfer

12. transferFrom

13. transferOwnership

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint			
Max / Total Supply	N/A		

Comments:

- The tokens will be only once while deployment of the contract but the total supply will also be set by the deployer at the time of deployment

18

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock			
Deployer cannot burn			

Comments:

- Please keep in mind that all the individuals can burn tokens from their own account or from another account if they have an allowance.

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause			

Overall checkup (Smart Contract Security)
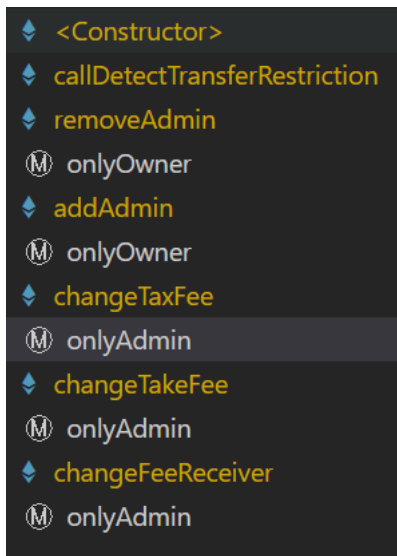
Tested	Verified

Legend

Attribute	Symbol
Verified / Checked	
Partly Verified	
Unverified / Not checked	
Not available	

Modifiers and public functions
v1.0

```
♦ <Constructor>
♦ callDetectTransferRestriction
♦ removeAdmin
Ⓜ onlyOwner
♦ addAdmin
Ⓜ onlyOwner
♦ changeTaxFee
Ⓜ onlyAdmin
♦ changeTakeFee
Ⓜ onlyAdmin
♦ changeFeeReceiver
Ⓜ onlyAdmin
```

Comments:

- Some of the functions has a modifier "onlyAdmin" which allows multiple authorities to do certain actions in the contract and we have also noticed that the fee of the transaction is being transferred to the "feeReceiver".

- Moreover, there can be multiple wallets who can act as Admin in the contract because there is not a max number defined for Admins.

- The owner of the "onlyOwner" can add as many "Admins" as they want.

- The authorities in control of the "onlyAdmin" other than the ones who controls the "owner" account can also modify the fees charge by the contract whenever they please.

- In conclusion, there can be "n" number of admins on this contract and all of those admins will have the permission to change the following:

  - The fee status of the contract
  - The fee receiver address of the contract
  - The amount of fees charged by the contract

22

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/GlobalGuideToken.sol	1	1	265	157	121	16	146	📋🔲Σ
	Totals	**1**	**1**	**265**	**157**	**121**	**16**	**146**	📋🔲Σ

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, ...)

<u>Audit Results</u>

<div style="background-color:#7CFC00; text-align:center;">

AUDIT PASSED

</div>

Critical issues

<div style="background-color:#7CFC00; text-align:center; color:green;">No critical issues</div>

High issues

<div style="background-color:#7CFC00; text-align:center; color:green;">No high issues</div>

Medium issues

Issue	File	Type	Line	Description
#1	Main	Fee can be 100%	304	The Admins of this contract can set the fee percent as 100 or more and the transfer "amount*taxFee" will be credited to the fee receiver's address rather than "to" address. This is a medium issue because there can be 'n' number admins with control of both feeReceiver and fee percent.

Low issues

Issue	File	Type	Line	Description
#1	Main	A floating pragma is set	3	The current pragma Solidity directive is „"^0.8.0".
#2	Main	Missing Events	182,188,194, 200,206	Emit an event for critical parameter changes.
#3	Main	Missing zero check	138,188,206	Check that the address is not zero
#4	Main	Local variables shadowing	135,136	Rename the local variables that shadow another component. In this case, "name" and "symbol" is shadowing the variable with the ERC20 contract's variables

Informational issues

Issue	File	Type	Line	Description
#1	Main	Unused state variables	127	Make sure to remove all the unused state variables
#2	Main	Uninitialized local variables	228	Make sure to initialize all the variables
#3	Main	Missing check for existing entry.	188	Ensure that the address doesn't already exist in the mapping before adding it
#4	Main	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.

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.

31.July,2022:

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

[1 2 4](#)	Storage Location		
[S W C - 1 2 3](#)	Requirement Violation	[CWE-573: Improper Following of Specification by Caller](#)	**PASSED**
[S W C - 1 2 2](#)	Lack of Proper Signature Verification	[CWE-345: Insufficient Verification of Data Authenticity](#)	**PASSED**
[S W C - 1 2 1](#)	Missing Protection against Signature Replay Attacks	[CWE-347: Improper Verification of Cryptographic Signature](#)	**PASSED**
[S W C - 1 2 0](#)	Weak Sources of Randomness from Chain Attributes	[CWE-330: Use of Insufficiently Random Values](#)	**PASSED**
[S W C - 1 1 9](#)	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	**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**

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY