SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Binance Wealth Matrix

Audit

Security Assessment
02. May, 2023

For

SolidProof_io

@solidproof_io

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	19. February 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1,1	27. February 2023	• Reaudit
1.2	02. May 2023	• Reaudit Climb token

Network
Binance

Website
www.binancewealthmatrix.com

Telegram
https://t.me/BinanceWealthMatrix

Twitter
https://twitter.com/BinanceWM

Description

Each CLIMB token utilizes a built-in contract exchange system that renounces the need for a traditional Liquidity Pool. Rather than a Liquidity Pool pairing of the backing asset to the token using a traditional market maker method for exchange and price calculation, both assets are stored within the contract itself. To purchase CLIMB tokens, each investor interacts directly with the contract via our dApp using BNB (BEP20). Investors can interact with the contract using BNB or USDT

Project Engagement

During the Date of 19 February 2023, **Binance Wealth Matrix 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://github.com/s69hub/BinanceWealthMatrix-contracts
- Commit: d1a98903c78ec55928bd27f0cecf1138c4991c06

v1.1

- https://github.com/s69hub/BinanceWealthMatrix-contracts
- Commit: 09317f59f320cea6ca9ccd2a520045561c25dd7c

v1.2

- https://github.com/jmanywhere/climb-token/blob/main/contracts/ClimbV2.sol
- Commit: https://github.com/jmanywhere/climb-token/commit/31975829e7c7d93b0dad825d552ca732c0e0815c

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.

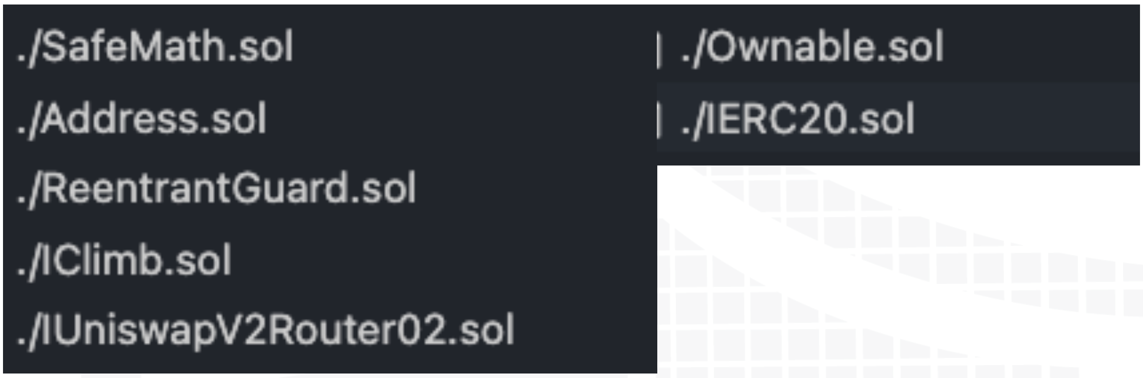
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program 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.

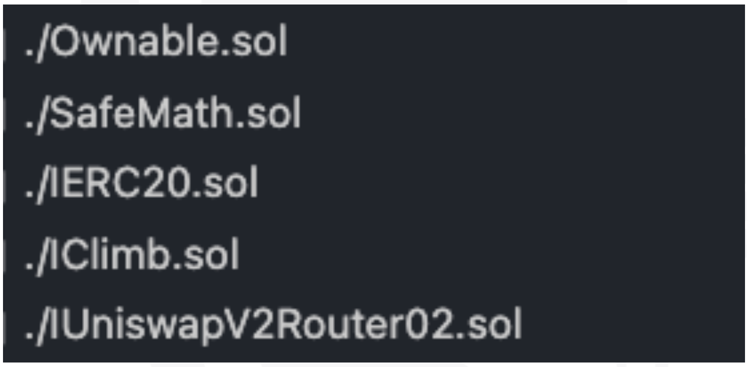
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:



```
./SafeMath.sol
./Address.sol
./ReentrantGuard.sol
./IClimb.sol
./IUniswapV2Router02.sol
./Ownable.sol
./IERC20.sol
```



```
./Ownable.sol
./SafeMath.sol
./IERC20.sol
./IClimb.sol
./IUniswapV2Router02.sol
```

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

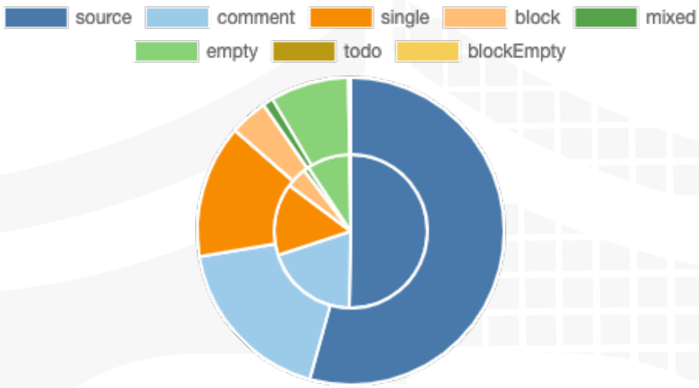
File Name	SHA-1 Hash
contracts/FeeReceiver.sol	f7e536503e1d964cb6bc397bbc60c1d8011460a4
contracts/Context.sol	6a0b5b8e1b849d1ea73eabcfb1c9cd7e0cdbc91b
contracts/IClimb.sol	1fb036ce8f980483f47e6481ed5d88cafabf31dd
contracts/Address.sol	2627336e3d80494975461b9e231cdd6ceaa420a2
contracts/SafeMath.sol	6005a330295839b0c2f0a1c73aa592ef262ba031
contracts/Matrix.sol	ccd98c619c00e87ffadc0bbda00123536ffb50f3
contracts/Ownable.sol	802cd4dd8338a4a3251ac6e50e14d85e79a03d4e
contracts/IUniswapV2Router02.sol	c4e0e2d2fd72fdcbc83eb01b646291bbeebfda0d
contracts/ReentrantGuard.sol	a0cb0f6c9feabfffe8e1b0fda8f8470ed4586ad3
contracts/ClimbToken.sol	fbfc3b598fceaf85fdf7bc730d0ad33a4a6ff9a2
contracts/IERC20.sol	1fce5436a768e8783f72b1bbdfcbcb6b9373c701

v1.2

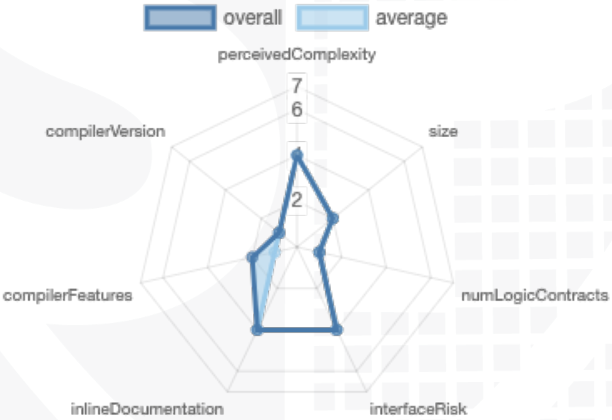
File Name	SHA-1 Hash
contracts/ClimbV2.sol	128e70154da6b8566e6071600b057ac866e04c55

Metrics

Source Lines
v1.2

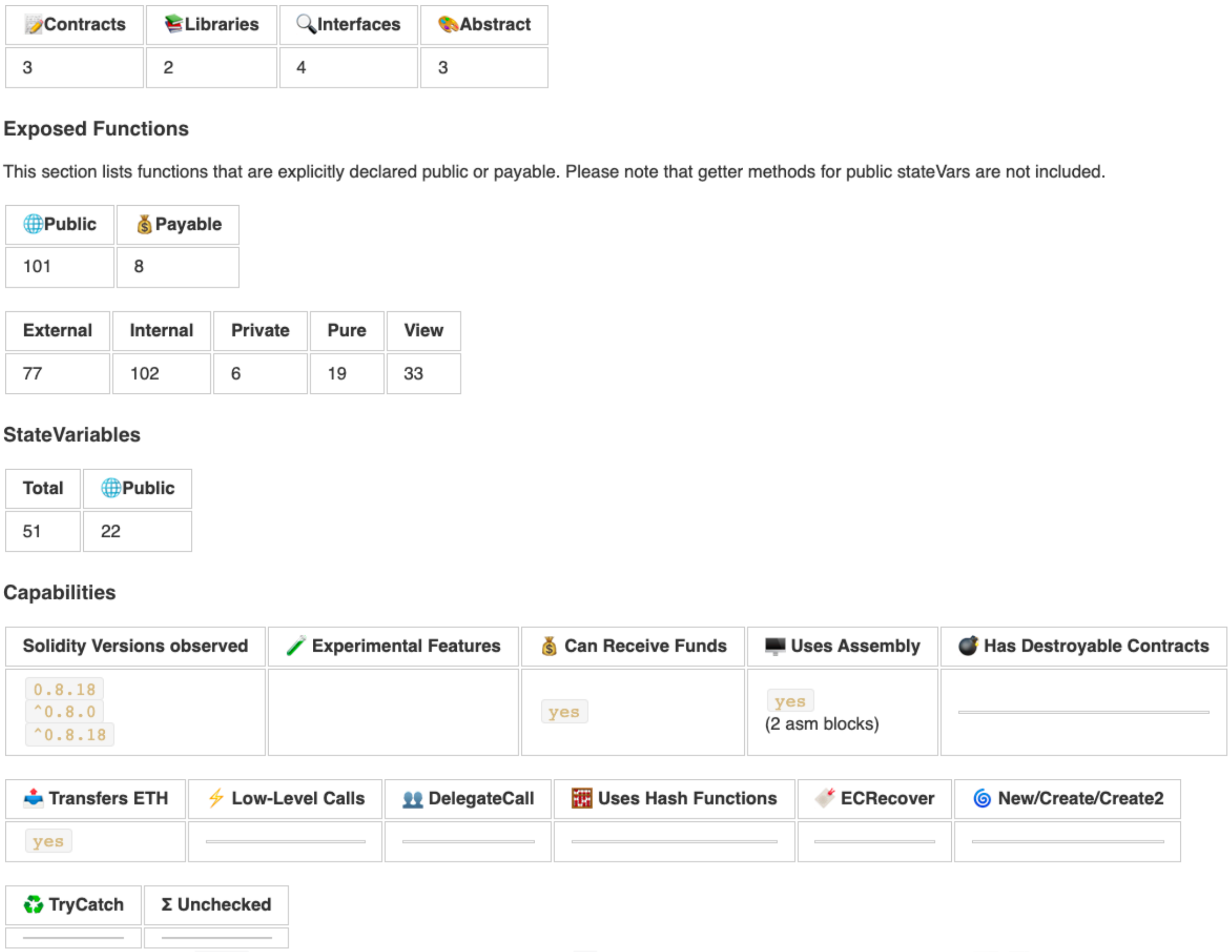


Risk Level
v1.2



Capabilities

Components



📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
3	2	4	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.

🌐Public	💰Payable
101	8

External	Internal	Private	Pure	View
77	102	6	19	33

StateVariables

Total	🌐Public
51	22

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts

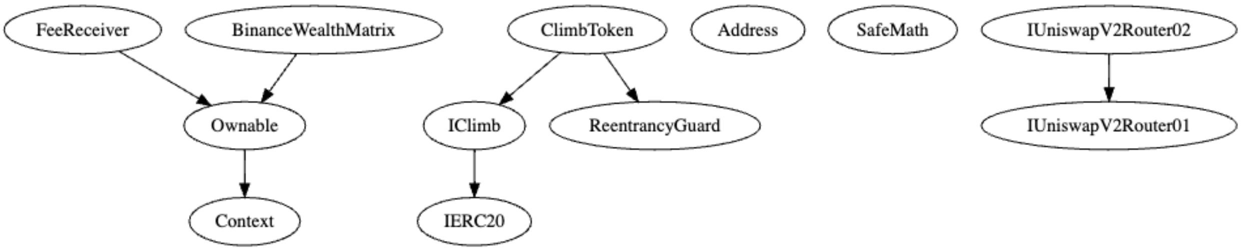
`0.8.18`
`^0.8.0`
`^0.8.18`
		`yes`	`yes`
			(2 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`					

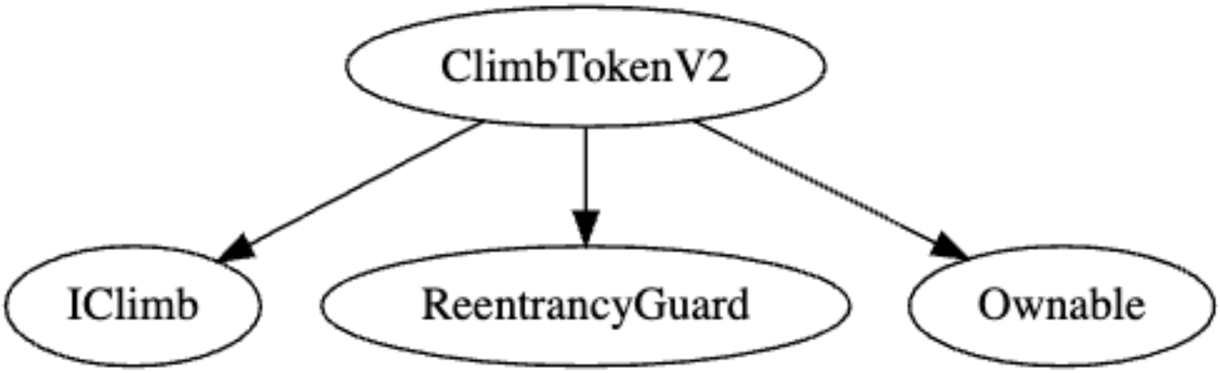
♻️ TryCatch	Σ Unchecked

Inheritance Graph

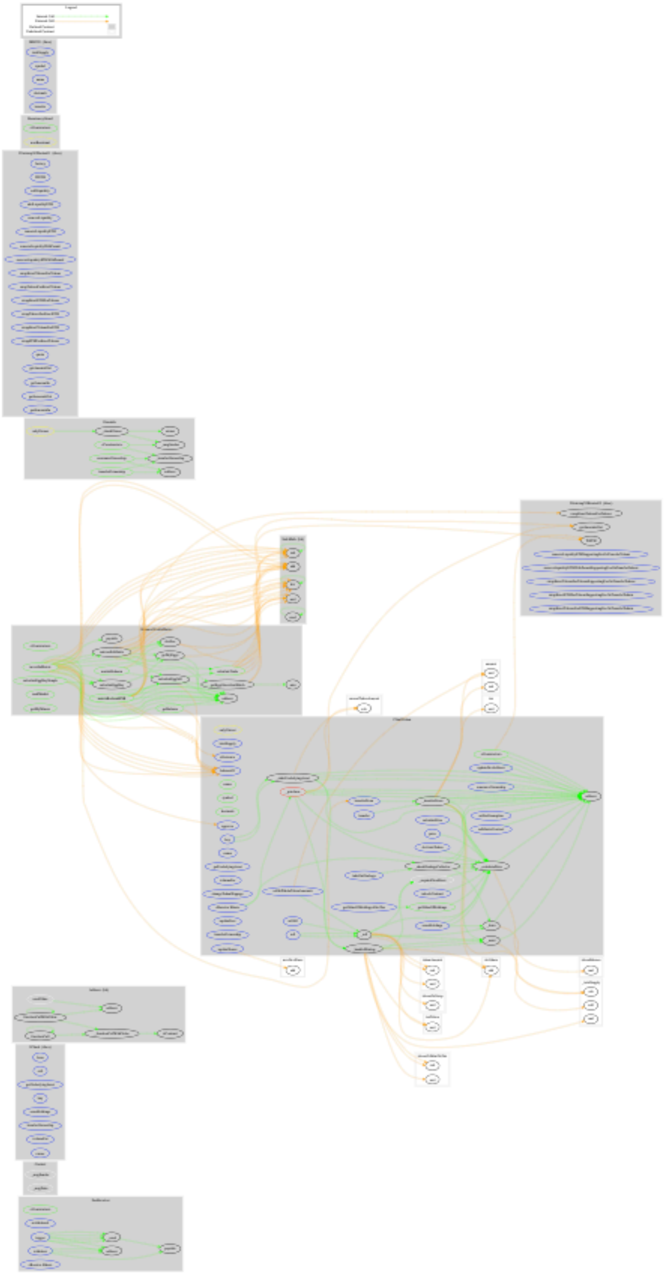
v1.0



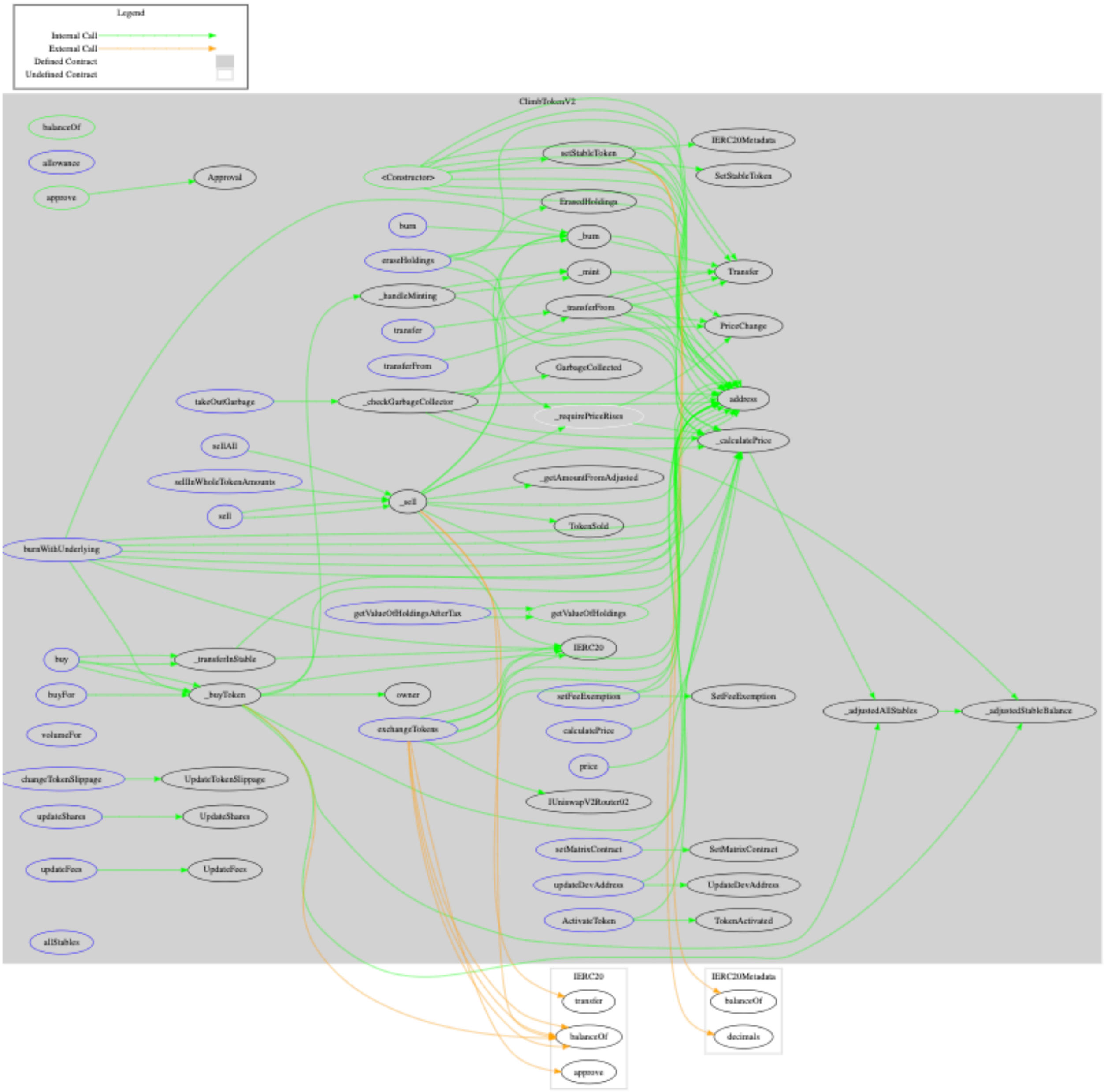
v1.2



CallGraph
v1.0



v1.2



Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ClimbTokenV2
balanceOf
allowance
approve
Approval
<Constructor>
setStableToken
IERC20Metadata
SetStableToken
ErasedHoldings
burn
_burn
eraseHoldings
_mint
Transfer
_handleMinting
_transferFrom
transfer
PriceChange
transferFrom
GarbageCollected
takeOutGarbage
_checkGarbageCollector
address
_requirePriceRises
_calculatePrice
sellAll
_getAmountFromAdjusted
sellInWholeTokenAmounts
_sell
sell
TokenSold
burnWithUnderlying
getValueOfHoldingsAfterTax
getValueOfHoldings
IERC20
buy
_transferInStable
buyFor
_buyToken
owner
setFeeExemption
SetFeeExemption
_adjustedAllStables
_adjustedStableBalance
volumeFor
exchangeTokens
calculatePrice
price
changeTokenSlippage
UpdateTokenSlippage
IUniswapV2Router02
updateShares
UpdateShares
setMatrixContract
SetMatrixContract
updateFees
UpdateFees
updateDevAddress
UpdateDevAddress
ActivateToken
TokenActivated
allStables
IERC20
transfer
balanceOf
approve
IERC20Metadata
balanceOf
decimals

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. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

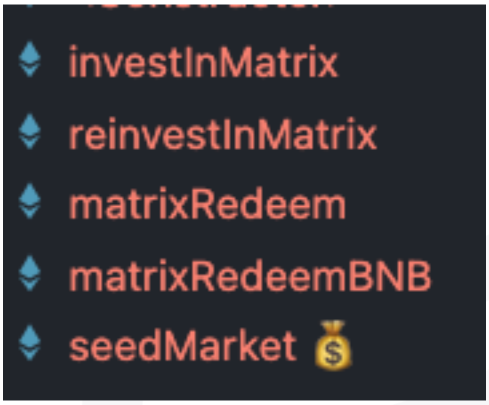
Name	
Is contract an upgradeable?	No

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	✓	✓	✓

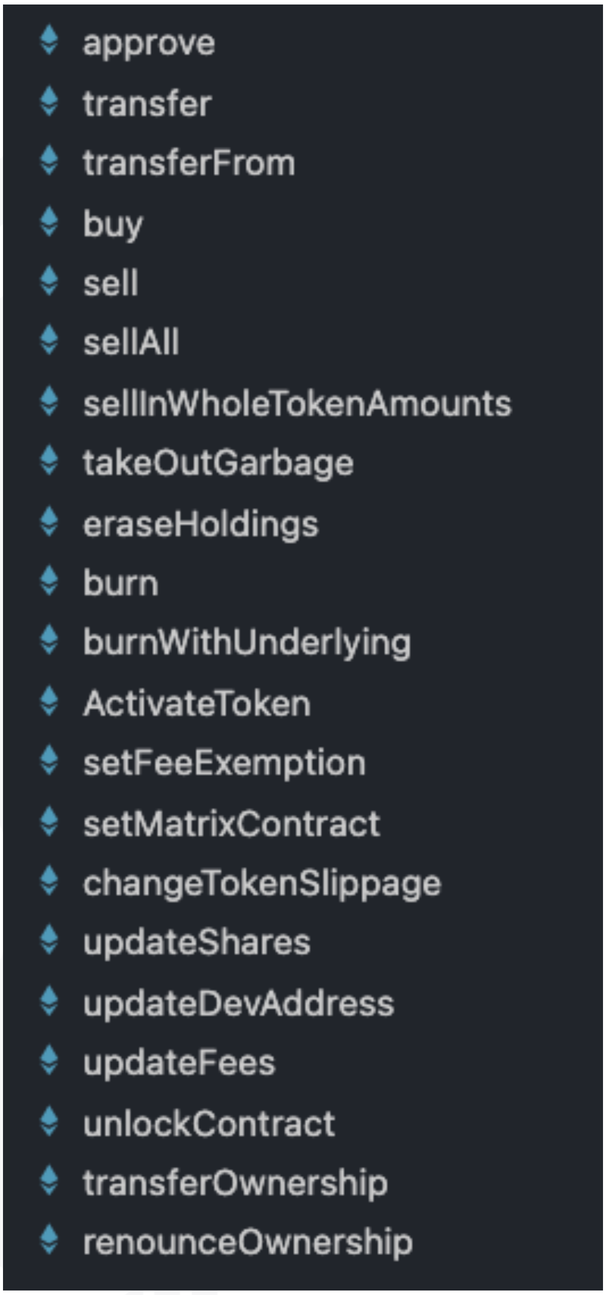
Write functions of contract v1.0

Matrix.sol



- investInMatrix
- reinvestInMatrix
- matrixRedeem
- matrixRedeemBNB
- seedMarket 💰

ClimbToken.sol



- approve
- transfer
- transferFrom
- buy
- sell
- sellAll
- sellInWholeTokenAmounts
- takeOutGarbage
- eraseHoldings
- burn
- burnWithUnderlying
- ActivateToken
- setFeeExemption
- setMatrixContract
- changeTokenSlippage
- updateShares
- updateDevAddress
- updateFees
- unlockContract
- transferOwnership
- renounceOwnership

Deployer cannot mint any new tokens

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

Comments:

v1.0

- Tokens will be minted automatically when the token is bought, sell or staked with BNB, or USDT

Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Tokens can be burned by msg.sender

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

v1.0

- The fees cannot exceed 5%

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	–	–	–

Overall checkup (Smart Contract Security)

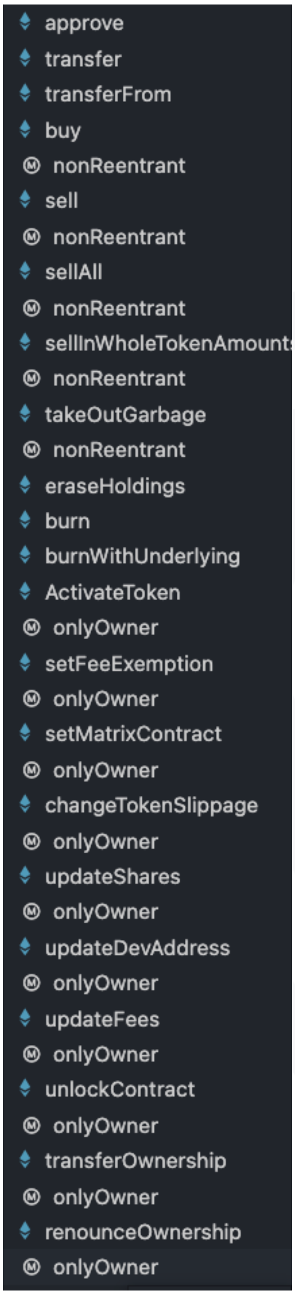
Tested	Verified
✓	✓

Legend

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

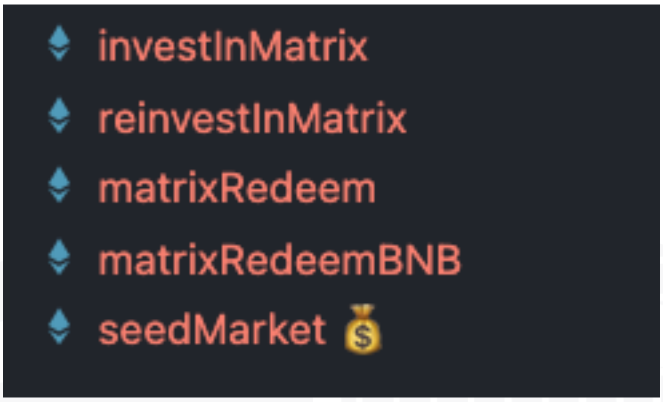
Modifiers and public functions
v1.0

ClimbToken.sol



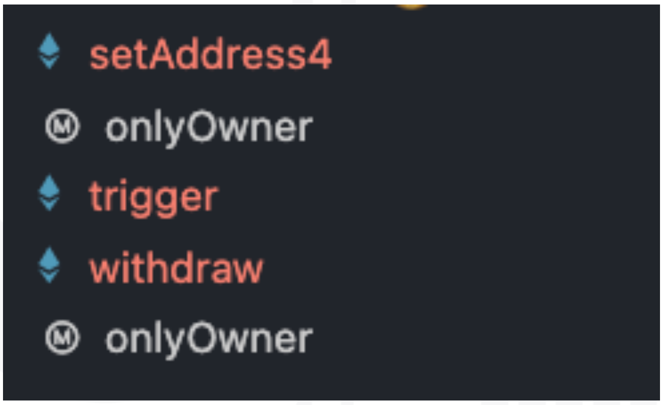
approve
transfer
transferFrom
buy
nonReentrant
sell
nonReentrant
sellAll
nonReentrant
sellInWholeTokenAmount
nonReentrant
takeOutGarbage
nonReentrant
eraseHoldings
burn
burnWithUnderlying
ActivateToken
onlyOwner
setFeeExemption
onlyOwner
setMatrixContract
onlyOwner
changeTokenSlippage
onlyOwner
updateShares
onlyOwner
updateDevAddress
onlyOwner
updateFees
onlyOwner
unlockContract
onlyOwner
transferOwnership
onlyOwner
renounceOwnership
onlyOwner

Matrix.sol



investInMatrix
reinvestInMatrix
matrixRedeem
matrixRedeemBNB
seedMarket 💰

FeeReceiver.sol



setAddress4
onlyOwner
trigger
withdraw
onlyOwner

Ownership Privileges:

- Activate token but cannot deactivate it
- Include/Exclude wallets from fees
- Set the matrix contract. Aware of this because if the matrix contract is updated by the owner then new contract may bring some new security flaws.
- Update slippage
- Update dev address

- Unlock contract but cannot lock it again
- Owner can withdraw the balance of the FeeReceiver contract
- While staking the underlying asset in the climb token contract, there is no slippage in the function on line 316

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

v1.2

- There is missing state visibility for the variable *_volumeFor*. By default it is set to private in L69. The same appears to the *Token_Activated* variable in L75
  Recommendation
  It is recommended to set the state visibility explicitly.
  Status: Resolved

- Some require statements missing the error message. If the contract reverts without an error message it is hard to understand what is happened.
  Recommendation
  Add an error message to every require statements to inform the investors if the called functions reverts and what happened.
  Status: Resolved

- Wrong comment or logic is missing. Ensure that either the logic is correct or the comment is adjusted. Router was not exempted.
  Status: Resolved

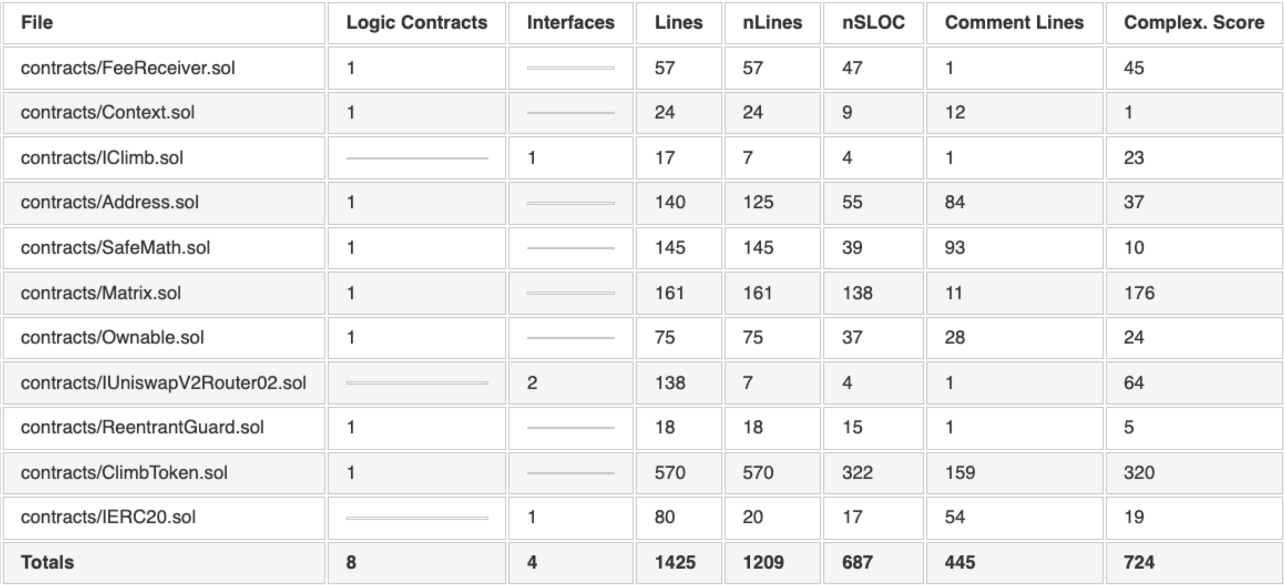
- *_tokenSlippage* variable L72 has no functionality in the contract.
  Recommendation
  Use it or remove it from the contract.
  Status: Resolved

- To optimize the code move the L652 *IERC20Metadata stableToken = IERC20Metadata(_stable);* in the *else if* condition because there is the only place where it was used.
  Status: Resolved

- Unuse local variable in the *exchangeTokens* function.
  Recommendation
  Remove the unused local variable or use it in the function.
  Status: Resolved

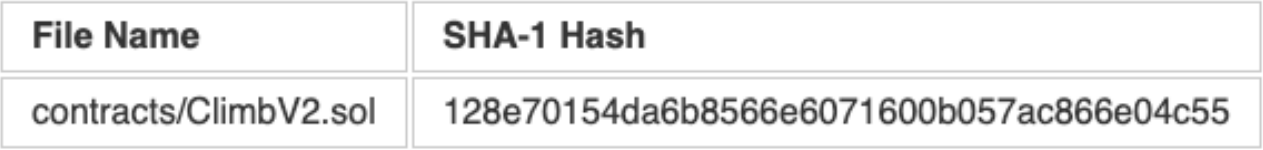
Source Units in Scope

v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/FeeReceiver.sol	1		57	57	47	1	45
contracts/Context.sol	1		24	24	9	12	1
contracts/IClimb.sol		1	17	7	4	1	23
contracts/Address.sol	1		140	125	55	84	37
contracts/SafeMath.sol	1		145	145	39	93	10
contracts/Matrix.sol	1		161	161	138	11	176
contracts/Ownable.sol	1		75	75	37	28	24
contracts/IUniswapV2Router02.sol		2	138	7	4	1	64
contracts/ReentrantGuard.sol	1		18	18	15	1	5
contracts/ClimbToken.sol	1		570	570	322	159	320
contracts/IERC20.sol		1	80	20	17	54	19
Totals	**8**	**4**	**1425**	**1209**	**687**	**445**	**724**

v1.2



File Name	SHA-1 Hash
contracts/ClimbV2.sol	128e70154da6b8566e6071600b057ac866e04c55

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

No low issues

Informational issues

Issue	File	Type	Line	Description
#4	Matrix.sol	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/latest/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. February 2023:

- There is still an owner (Owner still has not renounced ownership)
- In the climb token contract, the tax tokens will be minted into the dev address, and the tax amount will be burned.
- The owner will be able to stake tokens even before the token is activated.
- The price of the token will be decided by dividing the underlying balance of the contract by total supply.

- Minting fee will be charged every time the tokens are bought or purchased.
- Selling of tokens can only take place when the price is risen
- Read whole report and modifiers section for more information

02. May 2023:

- In the version 1.2 only the Climb Token was audited. Everything regarding to the ClimbV2 token is liste under v1.2 sections

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	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	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

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY